Godot Engine Documentation

Versión latest

Juan Linietsky, Ariel Manzur and the Godot community

15 de noviembre de 2017

Tutoriales

1. Aprendiendo paso a paso 3
2. Motor 57
3. Tutoriales 2D 83
4. Tutoriales 3D 161
5. Redes 235
6. Plugins de Editor 241
7. Misceláneo 243
8. Conducto de Asset 293
9. Class reference 329
10. Languages 835
11. Cheat sheets 871
12. Compiling 879
13. Advanced 917
14. Contributing 955

Nota: El Motor Godot es un proyecto de código abierto desarrollado por una comunidad de voluntarios. Esto implica que el equipo de documentación siempre puede utilizar tus devoluciones y ayudas para mejorar nuestros tutoriales y referencia de clases. Por lo que si no logras entender algo, o no puedes encontrar lo que necesitas en los documentos, ayúdanos a hacer una mejor documentación y déjanos saber! Envía un problema al repositorio GitHub, o moléstanos en nuestro canal IRC #godotengine-devel!

La documentación principal de este sitio esta organizada en algunas secciones:

- *Tutoriales*
- *Referencia*
- *Comunidad*

CAPÍTULO 1

Aprendiendo paso a paso

1.1 Escenas y nodos

1.1.1 Introducción

Imagina por un segundo que ya no eres un desarrollador de juegos. En su lugar, eres un chef! Cambia tu atuendo hípster por un un gorro de cocinero y una chaqueta doblemente abotonada. Ahora, en lugar de hacer juegos, creas nuevas y deliciosas recetas para tus invitados.

Entonces, ¿Cómo crea un chef su receta? Las recetas se dividen en dos secciones, la primera son los ingredientes y en segundo lugar las instrucciones para prepararlos. De esta manera, cualquiera puede seguir la receta y saborear tu magnífica creación.

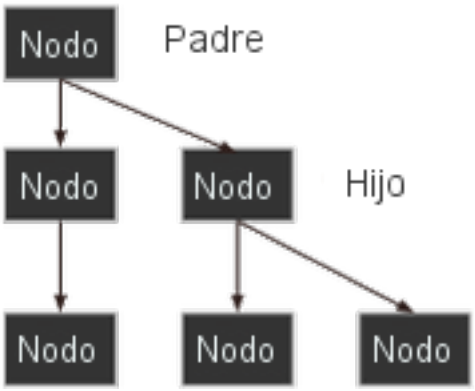
Hacer juegos en Godot se siente prácticamente igual. Usar el motor es como estar en una cocina. En esta cocina, los *nodos* son como un refrigerador lleno de ingredientes frescos con los cuales cocinar.

Hay muchos tipos de nodos, algunos muestran imágenes, otros reproducen sonido, otros muestran modelos 3D, etc. Hay docenas de ellos.

1.1.2 Nodos

Pero vayamos a lo básico. Un nodo es el elemento básico para crear un juego, tiene las siguientes características:

- Tiene un nombre.
- Tiene propiedades editables.
- Puede recibir una llamada a procesar en cada frame.
- Puede ser extendido (para tener mas funciones).
- Puede ser agregado a otros nodos como hijo.

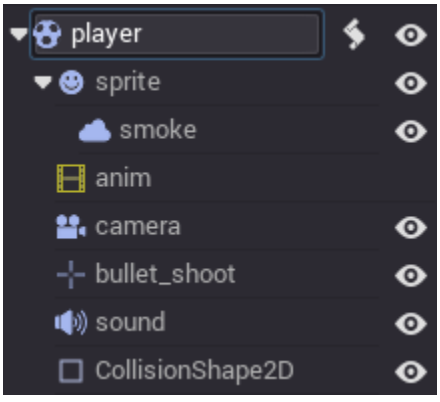


La última es muy importante. Los nodos pueden tener otros nodos como hijos. Cuando se ordenan de esta manera, los nodos se transforman en un **árbol**.

En Godot, la habilidad para ordenar nodos de esta forma crea una poderosa herramienta para organizar los proyectos. Dado que diferentes nodos tienen diferentes funciones, combinarlos permite crear funciones mas complejas.

Esto probablemente no es claro aun y tiene poco sentido, pero todo va a encajar unas secciones más adelante. El hecho mas importante a recordar por ahora es que los nodos existen y pueden ser ordenados de esa forma.

1.1.3 Escenas



Ahora que la existencia de nodos ha sido definida, el siguiente paso lógico es explicar qué es una Escena.

Una escena esta compuesta por un grupo de nodos organizados jerárquicamente (con estilo de árbol). Tiene las siguientes propiedades:

- Una escena siempre tiene un solo nodo raíz.
- Las escenas pueden ser guardadas a disco y cargadas nuevamente.
- Las escenas pueden ser *instanciadas* (mas sobre esto después).
- Correr un juego significa ejecutar una escena.
- Puede haber varias escenas en un proyecto, pero para iniciar, una de ellas debe ser seleccionada y cargada primero.

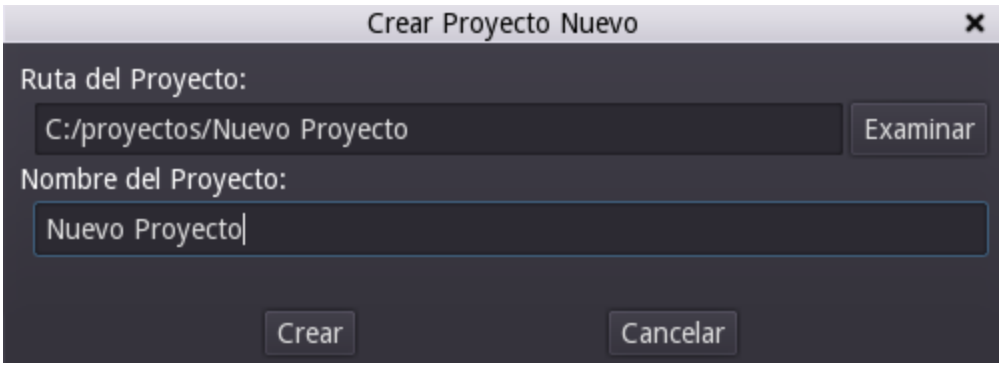
Básicamente, el motor Godot es un **editor de escenas**. Tiene más que suficientes herramientas para editar escenas 2D y 3D así como interfaces de usuario, pero el editor gira entorno al concepto de editar una escena y los nodos que la componen.

1.1.4 Creando un Nuevo Proyecto

La teoría es aburrida, así que vamos a cambiar de enfoque y ponernos prácticos. Siguiendo una larga tradición de tutoriales, el primer proyecto va a ser el "Hola Mundo!". Para esto, se usará el editor.

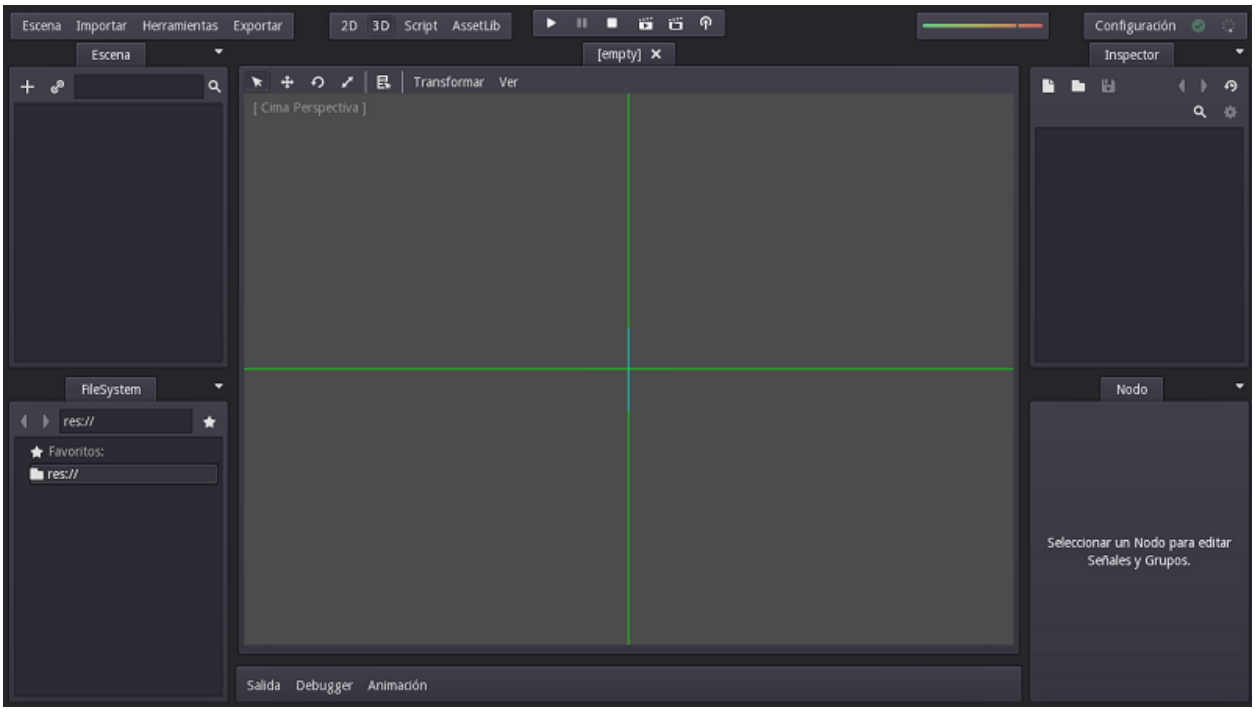
Cuando godot se ejecuta sin un proyecto, aparecerá el Gestor Proyectos. Esto ayuda a los desarrolladores a administrar sus proyectos.

Para crear un nuevo proyecto, la opción "Nuevo Proyecto" debe ser utilizada. Elije y crea una ruta para el proyecto y especificá el nombre del proyecto.

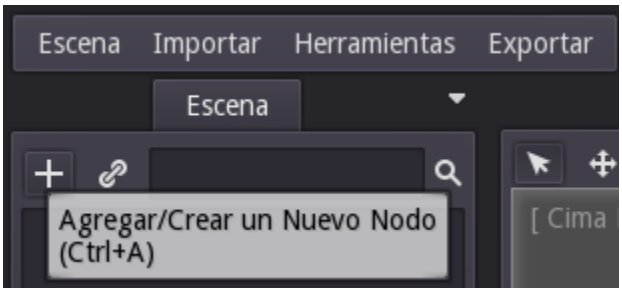


1.1.5 Editor

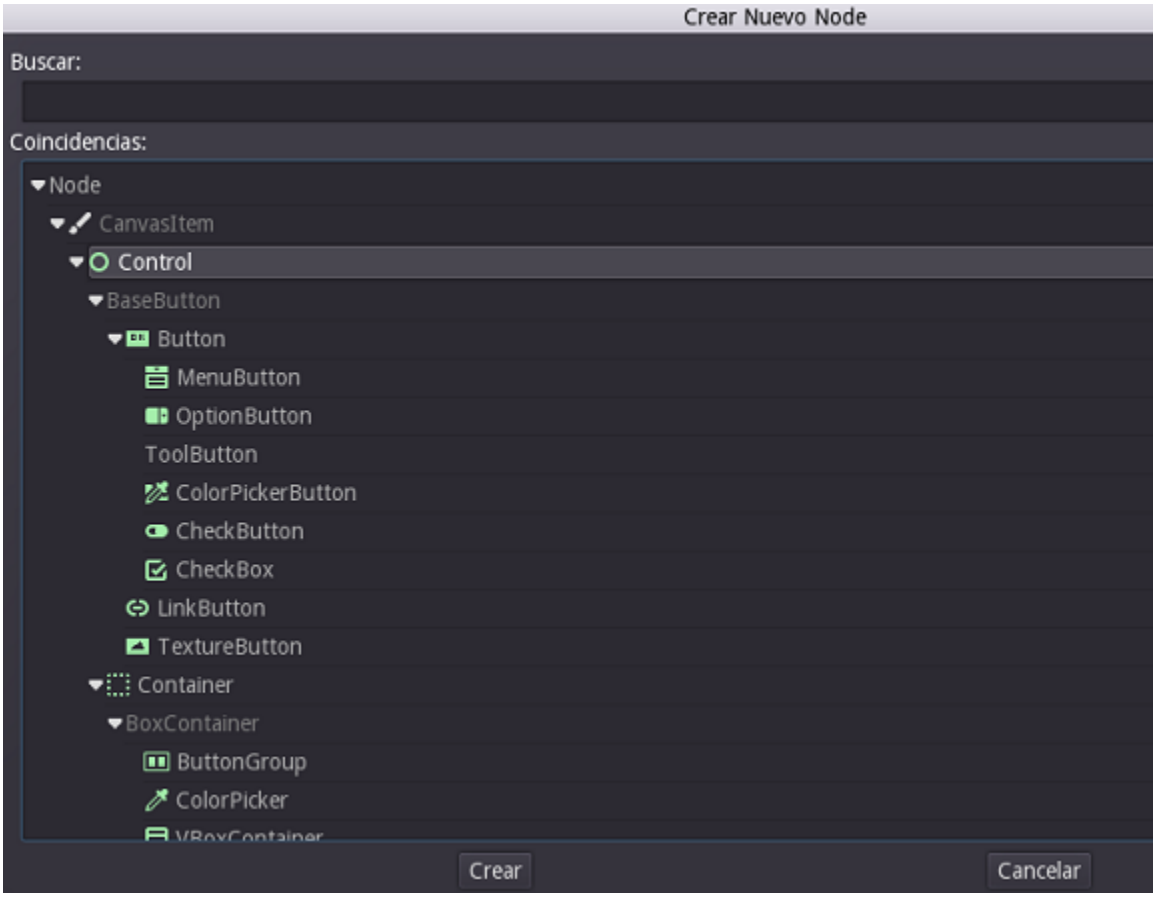
Una vez que el "Nuevo Proyecto" es creado, el siguiente paso es abrirlo. Esto abrirá el editor Godot. Así luce el editor cuando se abre:



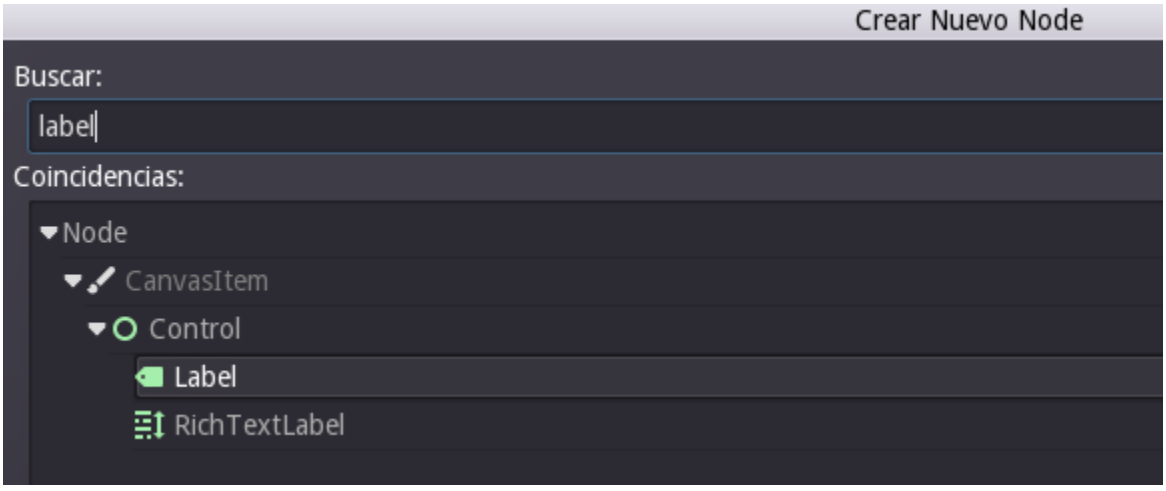
Como mencionamos antes, hacer juegos en Godot se siente como estar en una cocina, así que abramos el refrigerador y agreguemos algunos nodos frescos al proyecto. Comenzaremos con un "Hola Mundo!" Para hacer esto, el botón "Nuevo Nodo" debe ser presionado.



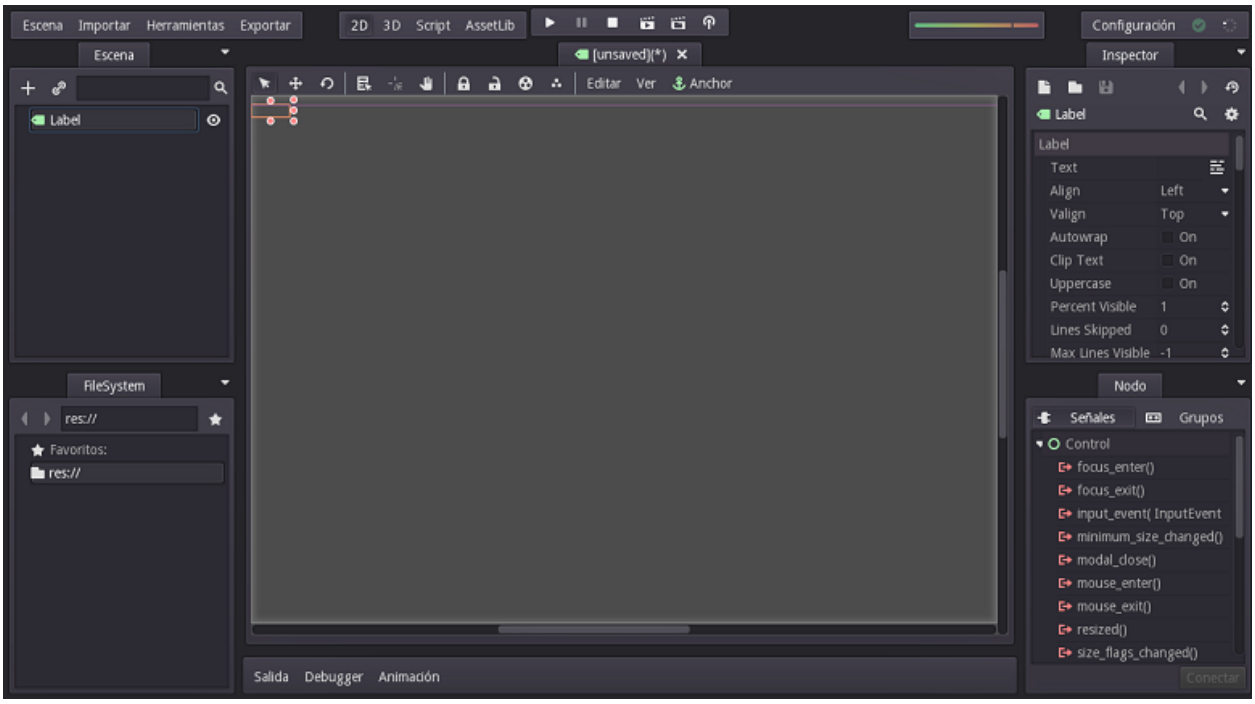
Esto abrirá el diálogo de Crear Nodo, mostrando una larga lista de nodos que pueden ser creados:



Desde allí, selecciona el nodo Label (Etiqueta) primero. Buscarlo es probablemente la forma más rápida:



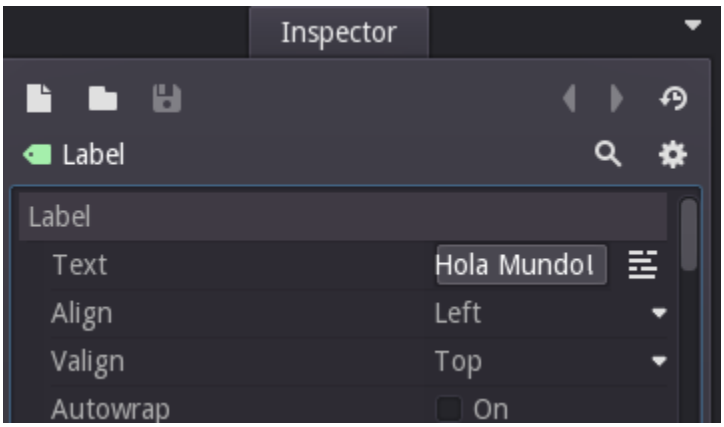
Y finalmente, crea el Label! Un montón de cosas suceden cuando Crear es presionado:



Primero que nada, la escena cambia hacia el editor 2D (porque Label es un Nodo de tipo 2D), y el Label aparece, seleccionada, en la esquina superior izquierda del viewport (ventana de visualización).

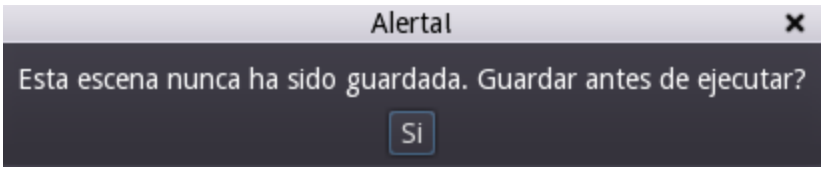
El nodo aparece en el editor de árbol de escena (caja en la esquina superior izquierda), y las propiedades de Label están en el Inspector (caja en el costado derecho)

El siguiente paso será cambiar la propiedad "Text" de la etiqueta, vamos a cambiarla a "Hola, Mundo!":

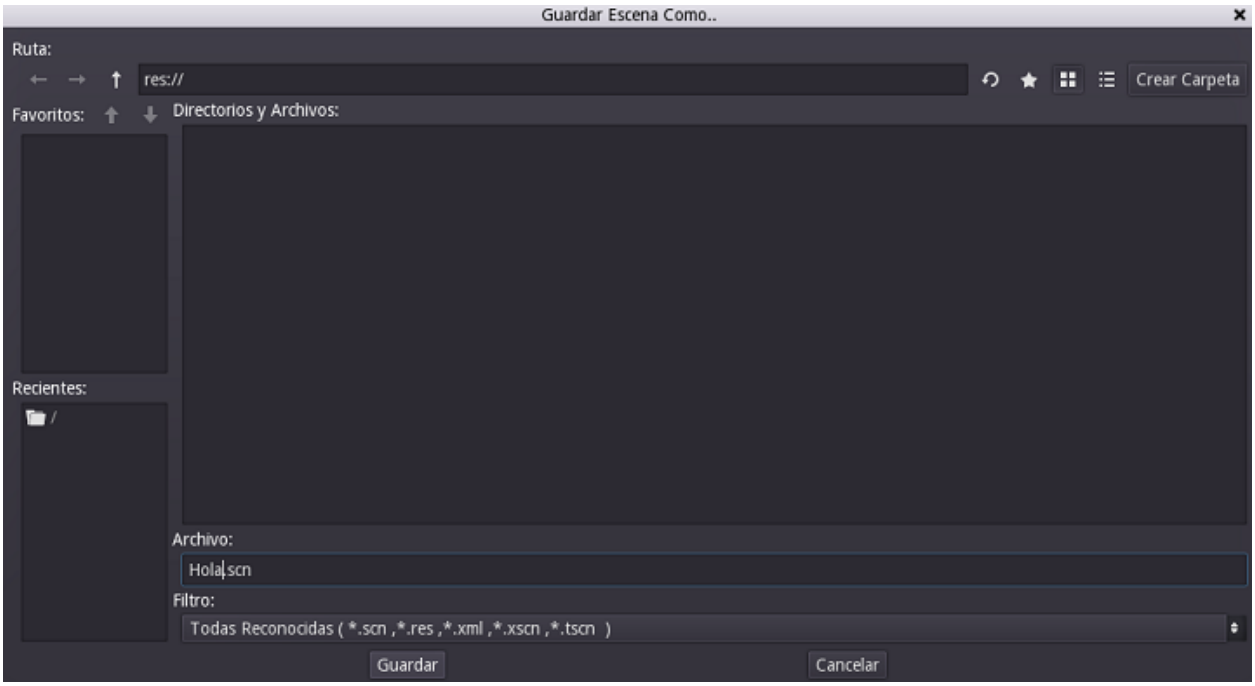


Bien, todo esta listo para correr la escena! Presiona el botón "PLAY SCENE" en la barra superior (o presiona F6):

Y... Uups.

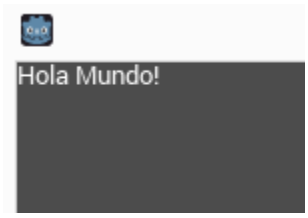


Las escenas necesitan ser salvadas para correr, por lo que guarda la escena en algo como hola.scn en Escena -> Guardar:



Y aquí es donde algo gracioso sucede. El de archivo es especial, y solo permite guardar dentro del proyecto. La raiz del proyecto es "res://" que significa "resource path" (camino de recursos). Esto significa que los archivos sólo pueden ser guardados dentro del proyecto. En el futuro, cuando hagas operaciones con archivos en Godot, recuerda que "res://" es el camino de recursos, y no importa la plataforma o lugar de instalación, es la forma de localizar donde están los archivos de recursos dentro del juego.

Luego de salvar la escena y presionar Reproducir Escena nuevamente, el demo "Hola, Mundo!" debería finalmente ejecutarse:



Éxito!

1.1.6 Configurando el proyecto

Ok, es momento de hacer algunas configuraciones en el proyecto. En este momento, la única forma de correr algo es ejecutar la escena actual. Los proyectos, sin embargo, tienen varias escenas por lo que una de ellas debe ser configurada como la escena principal. Esta escena es la que será cargada cuando el proyecto corre.

Estas configuraciones son todas guardadas en el archivo engine.cfg, que es un archivo de texto plano en el formato win.ini, para una edición fácil. Hay docenas de configuraciones que pueden ser configuradas en ese archivo para alterar como un proyecto se ejecuta, por lo que para hacer más simple el proceso, existe un cuadro de diálogo de configuración del proyecto, el cual es un tipo de interfaz para editar engine.cfg

Para acceder al cuadro de diálogo, simplemente ve a Escena -> Configuración de proyecto.

Cuando la ventana abre, la tarea será seleccionar la escena principal. Esto puede ser hecho fácilmente cambiando la propiedad application/main_scene y seleccionando 'hola.scn'

Con este cambio, presionar el botón de Play regular (o F5) va a correr el proyecto, no importa la escena que se está editando.

Yendo atrás con el diálogo de configuración de proyecto. Este diálogo permite una cantidad de opciones que pueden ser agregadas a engine.cfg y mostrar sus valores por omisión. Si el valor por defecto está bien, entonces no hay necesidad de cambiarlo.

Cuando un valor cambia, se marca un tick a la izquierda del nombre. Esto significa que la propiedad va a ser grabada al archivo engine.cfg y recordada.

Como una nota aparte, para futura referencia y un poco fuera de contexto (al fin de cuentas este es el primer tutorial!), también es posible agregar opciones de configuración personalizadas y leerlas en tiempo de ejecución usando el singleton *Globals*

1.1.7 Continuará...

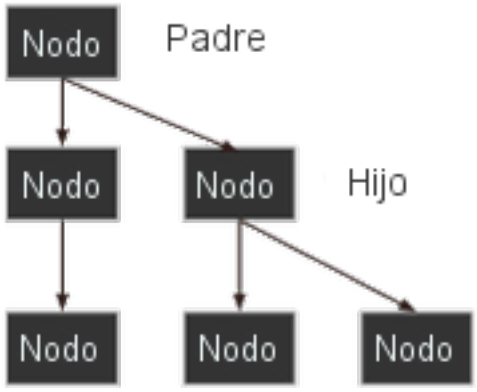
Este tutorial habla de "escenas y nodos", pero hasta ahora ha habido sólo *una* escena y *un* nodo! No te preocupes, el próximo tutorial se encargará de ello...

1.2 Instanciar

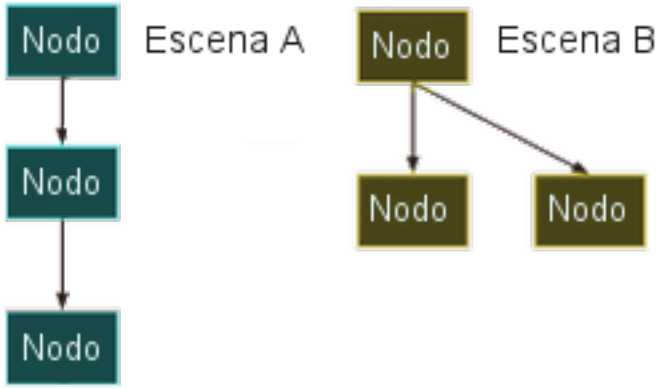
1.2.1 Fundamento

Tener una escena y tirar nodos en ella puede funcionar para proyectos pequeños, pero en la medida que el proyecto crece, más y más nodos son usados y rápidamente se puede volver inmanejable. Para resolver esto, Godot permite que un proyecto este separado en varias escenas. Esto, sin embargo, no funciona de la misma forma que en otros motores de juegos. De hecho, es bastante diferente, por lo que por favor no saltes este tutorial!

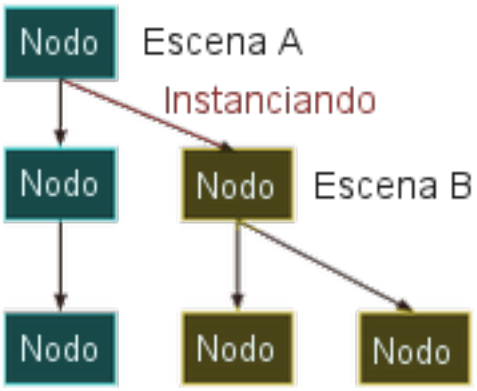
Para resumir: Una escena es una colección de nodos organizados como un árbol, donde soló pueden tener un nodo particular como nodo raíz.



En Godot, una escena puede ser creada y salvada a disco. Se pueden crear y guardar tantas escenas como se desee.



Luego, mientras editas una escena existente o creas una nueva, otras escenas pueden ser instanciadas como parte de está:

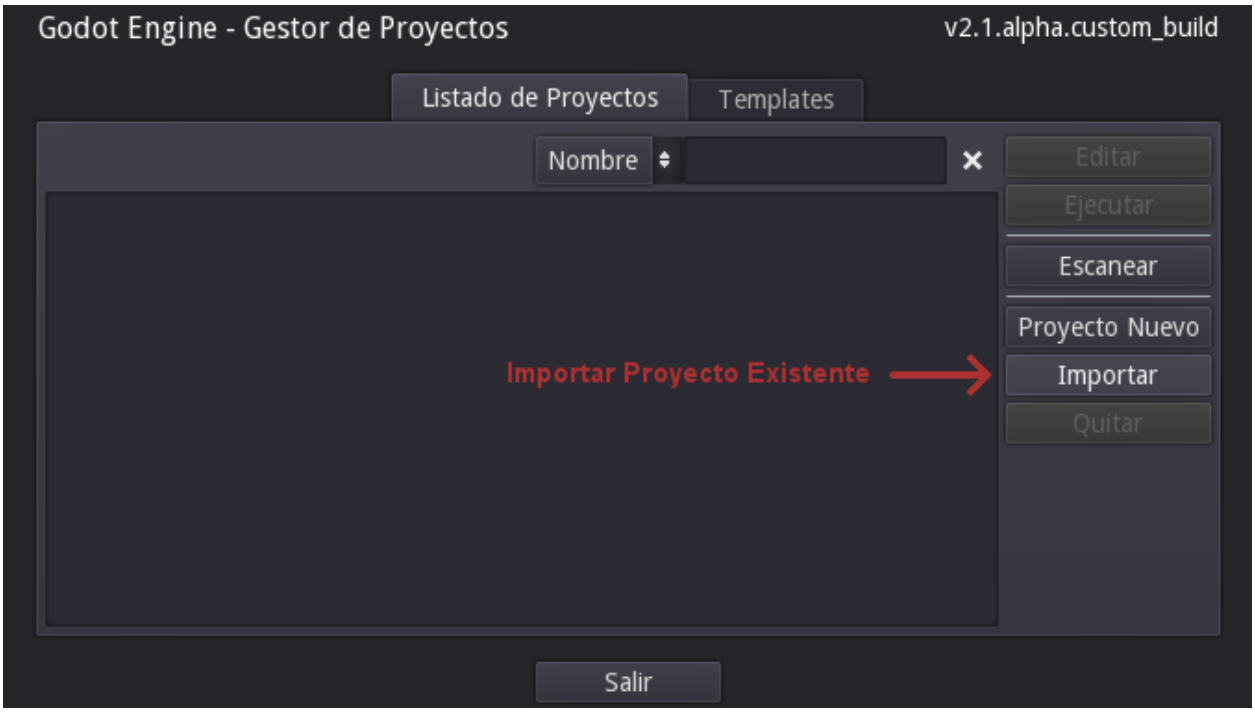


En la imagen anterior, la escena B fue agregada a la escena A como una instancia. Puede parecer extraño al principio, pero al final de este tutorial va a tener completo sentido!

1.2.2 Instanciar, paso a paso

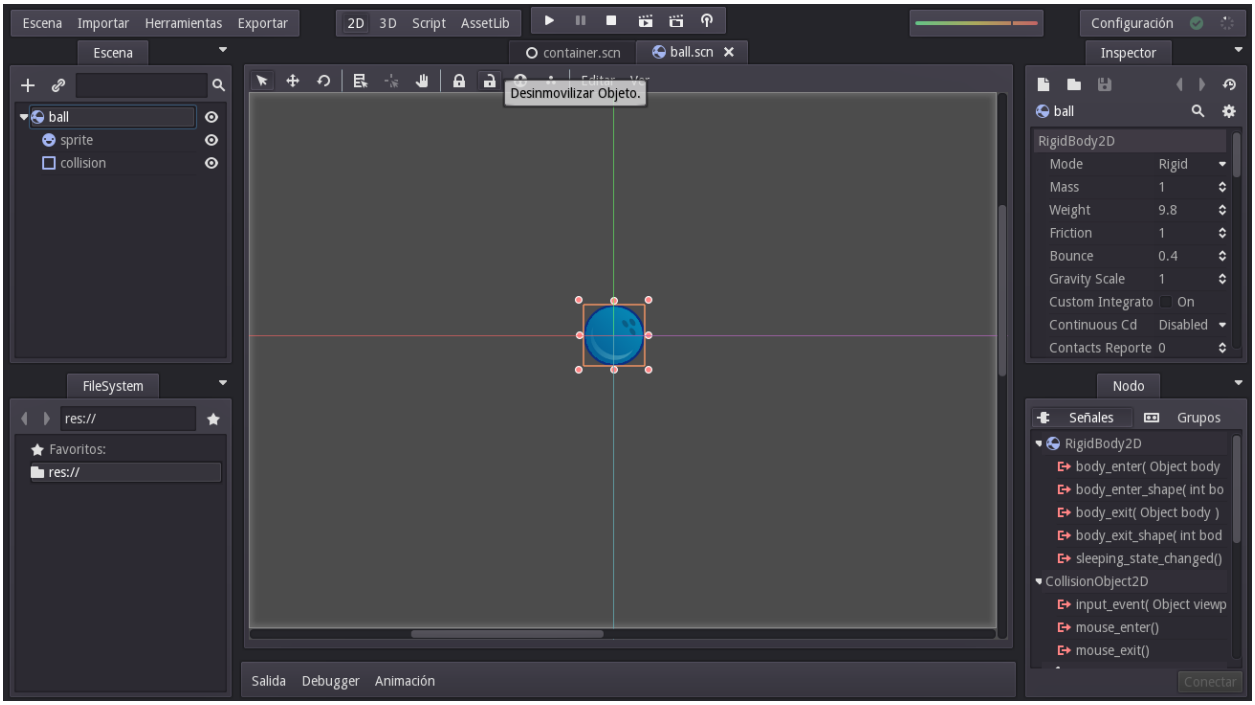
Para aprender como instanciar, comencemos descargando un proyecto de muestra: `instancing.zip`.

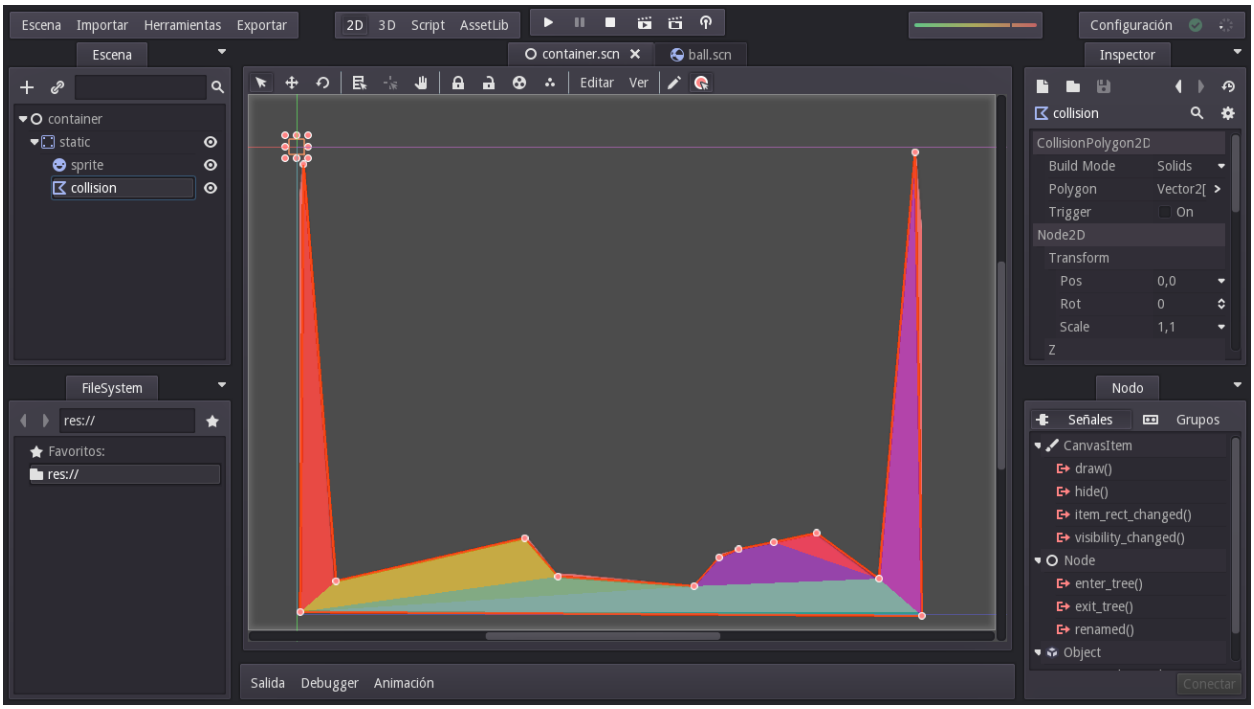
Descomprime esta escena en el lugar de tu preferencia. Luego, agrega esta escena al gestor de proyectos usando la opción 'Importar':



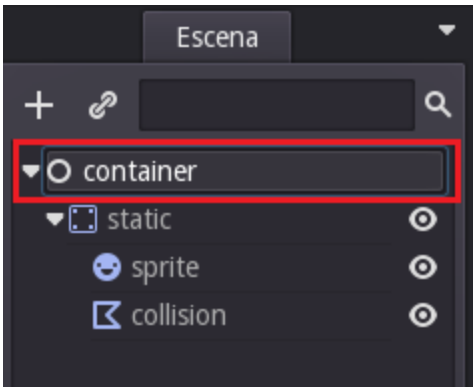
Simplemente navega hasta el lugar donde está el proyecto y abre "engine.cfg". El nuevo proyecto aparecerá en la lista de proyectos. Edita el proyecto usando la opción 'Editar'.

Este proyecto contiene dos escenas "ball.scn"(pelota) y "container.scn"(contenedor). La escena ball es solo una pelota con física, mientras que la escena container tiene una linda forma de colisión, de forma que las pelotas pueden tirarse allí.

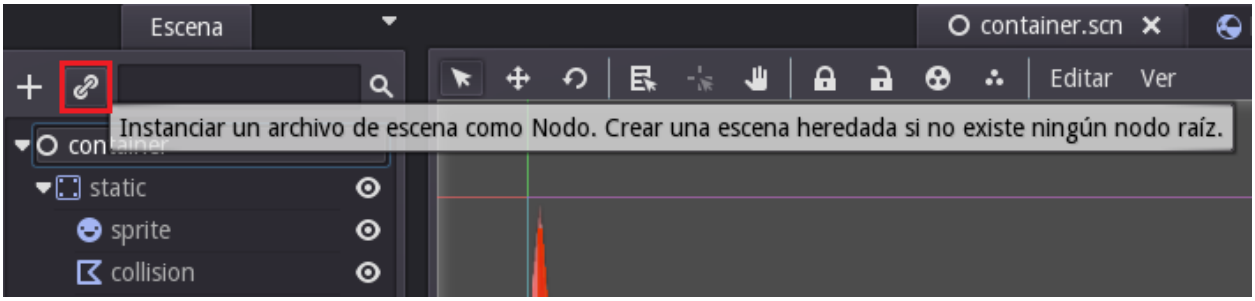




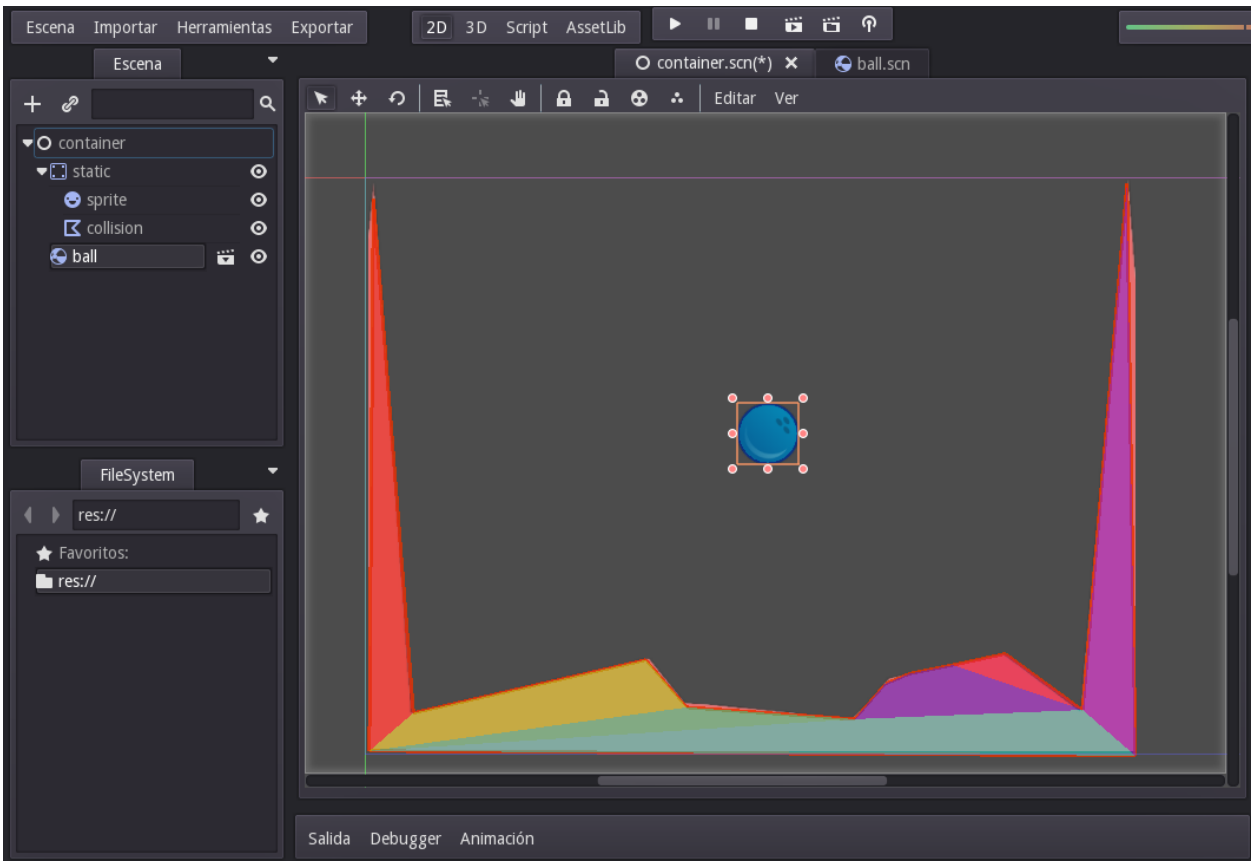
Abre la escena container, luego selecciona el nodo raíz:



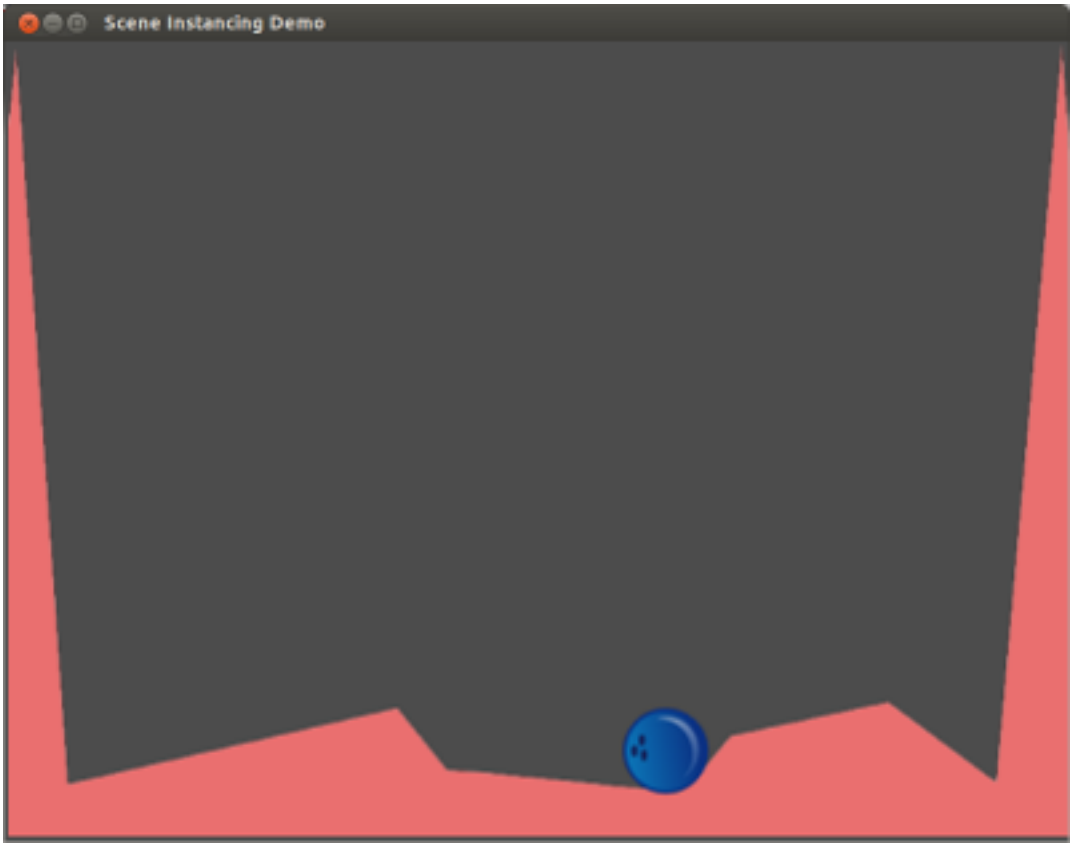
Después, presiona el botón con forma de cadena, este es el botón de instanciar!



Selecciona la escena de la pelota (ball.scn), la pelota debería aparecer en el origen (0,0), la mueves hasta el centro de la escena, algo así:



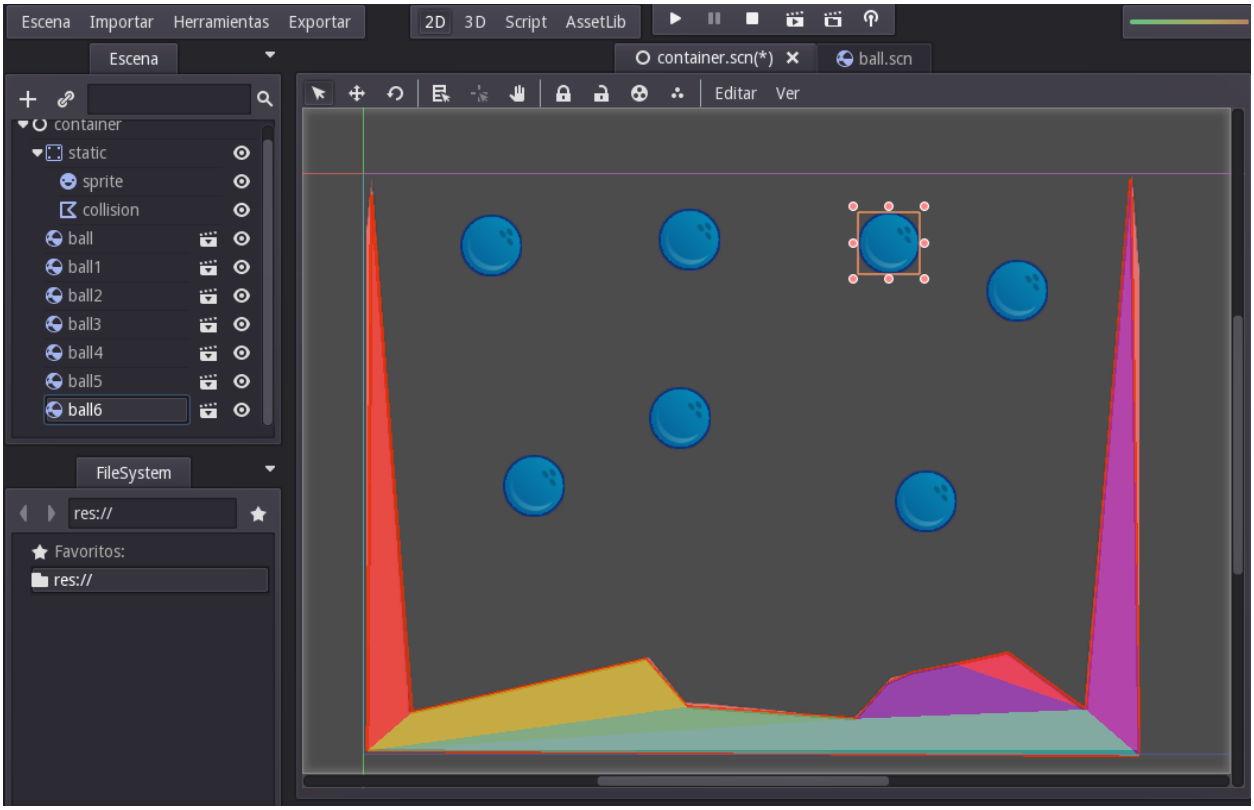
Presiona Reproducir y Voilà!



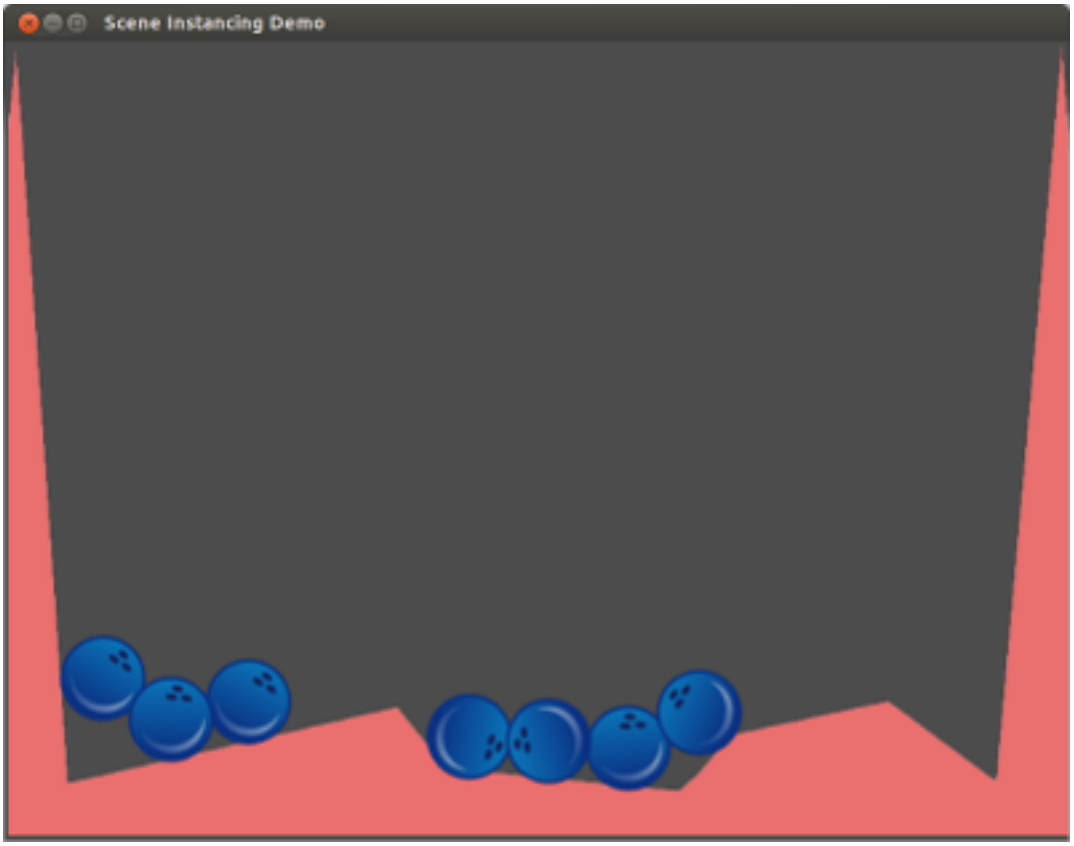
La pelota instanciada cayó hasta el fondo del pozo.

1.2.3 Un poco más

Puede haber tantas instancias como se desee en una escena, simplemente intenta instanciar más pelotas, o duplícalas (Ctrl-D o botón derecho -> Duplicar):



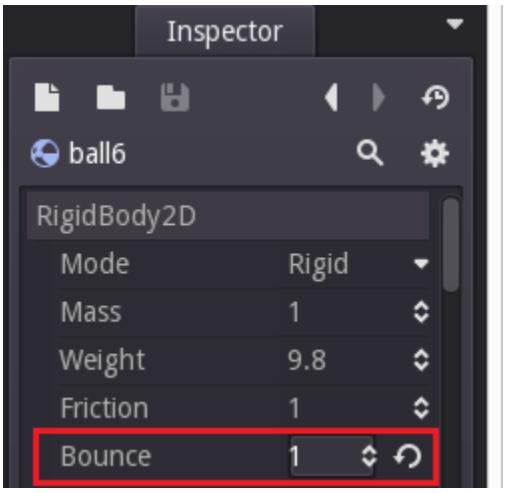
Luego intenta correr la escena nuevamente:



Está bueno, eh? Así es como funciona instanciar.

1.2.4 Editando instancias

Selecciona una de las muchas copias de las pelotas y ve al Inspector. Hagamos que rebote mucho más, por lo que busca el parámetro bounce(rebote) y configúralo en 1.0:



Lo próximo que sucederá es que un botón de "revertir" con forma de "flecha en círculo" aparecerá. Cuando este botón está presente, significa que hemos modificado una propiedad en la escena instanciada, ignorando el valor original. Aún si esa propiedad es modificada en la escena original, el valor personalizado siempre lo sobrescribirá. Tocando el botón de revertir restaurará la propiedad al valor original que vino de la escena.

1.2.5 Conclusión

Instanciar parece útil, pero hay más de lo que se ve a simple vista! La próxima parte del tutorial de instanciar cubrirá el resto...

1.3 Instanciar (continuación)

1.3.1 Recapitulación

Instanciar tiene muchos usos. A simple vista, al instanciar tienes:

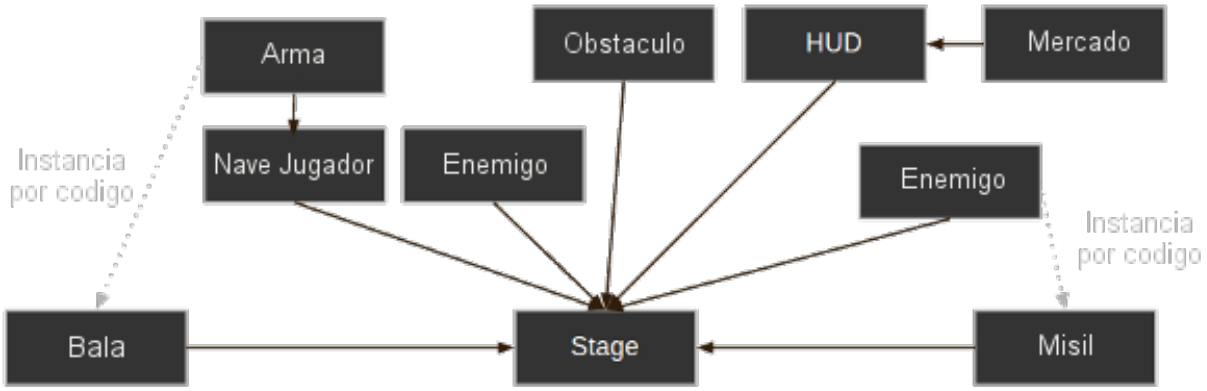
- La habilidad de subdividir las escenas y hacerlas mas fáciles de administrar.
- Una alternativa flexible a los prefabricados (y mucho mas poderoso dado que las instancias trabajan en varios niveles)
- Una forma de diseñar flujos de juegos mas complejos o incluso UIs (interfaces de usuario) (Los elementos de UIs son nodos en Godot también)

1.3.2 Lenguaje de diseño

Pero el verdadero punto fuerte de instanciar escenas es que como un excelente lenguaje de diseño. Esto es básicamente lo que hace especial a Godot y diferente a cualquier otro motor en existencia. Todo el motor fue diseñado desde cero en torno a este concepto.

Cuando se hacen juegos con Godot, el enfoque recomendado es dejar a un costado otros patrones de diseño como MVC o diagramas de entidad-relación y empezar a pensar en juegos de una forma mas natural. Comienza imaginando los elementos visibles en un juego, los que pueden ser nombrados no solo por un programador sino por cualquiera.

Por ejemplo, aquí esta como puede imaginarse un juego de disparo simple:

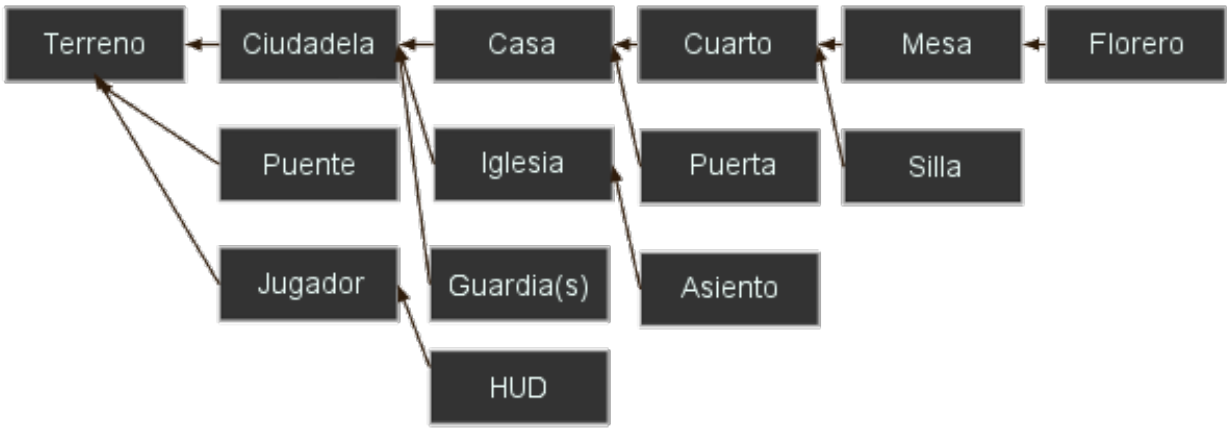


Es bastante sencillo llegar a un diagrama como este para casi cualquier tipo de juego. Solo anota los elementos que te vienen a la cabeza, y luego las flechas que representan pertenencia.

Una vez que este diagrama existe, hacer el juego se trata de crear una escena para cada uno de esos nodos, y usar instancias (ya sea por código o desde el editor) para representar la pertenencia.

La mayoría del tiempo programando juegos (o software en general) es usada diseñando una arquitectura y adecuando los componentes del juego a dicha arquitectura. Diseñar basado en escenas reemplaza eso y vuelve el desarrollo mucho mas rápido y directo, permitiendo concentrarse en el juego. El diseño basado en Escenas/Instancias es extremadamente eficiente para ahorrar una gran parte de ese trabajo, ya que la mayoría de los componentes diseñados se mapean directamente a una escena. De esta forma, se precisa poco y nada de código de arquitectura.

El siguiente es un ejemplo mas complejo, un juego de mundo abierto con un montón de assets(activos) y partes que interactúan.



Crea algunas habitaciones con muebles, luego conectalos. Crea una casa mas tarde, y usa esas habitaciones como su interior.

La casa puede ser parte de la ciudadela, que tiene muchas casas. Finalmente la ciudadela puede ser colocada en el terreno del mapa del mundo. También agrega guardias y otros NPCs(personajes no jugador) a la ciudadela, creando previamente sus escenas.

Con Godot, los juegos pueden crecer tan rápido como se desee, ya que se trata de crear mas escenas e instanciarlas.

El editor UI(interfaz de usuario) también esta diseñado para ser operado por personas que no son programadores, por lo que un equipo usual de desarrollo consiste de artistas 2D o 3D, diseñadores de niveles, diseñadores de juegos, animadores, etc todos trabajando en la interfaz del editor.

1.3.3 Sobrecarga de información!

No te preocupes demasiado, la parte importante de este tutorial es crear la conciencia de como las escenas e instanciar son usados en la vida real. La mejor forma de entender todo esto es hacer algunos juegos.

Todo se volverá muy obvio cuando se pone en practica, entonces, por favor no te rasques la cabeza y ve al siguiente tutorial!

1.4 Scripting

1.4.1 Introducción

Mucho se ha dicho sobre herramientas que permiten a los usuarios crear juegos sin programar. Ha sido un sueño para muchos desarrolladores independientes el crear juegos sin aprender a escribir código. Esto ha sido así por un largo tiempo, aun dentro de compañías, donde los desarrolladores de juegos desean tener mas control del flujo del juego (game flow).

Muchos productos han sido presentados prometiendo un entorno sin programación, pero el resultado es generalmente incompleto, demasiado complejo o ineficiente comparado con el código tradicional. Como resultado, la programación esta aquí para quedarse por un largo tiempo. De hecho, la dirección general en los motores de jugos ha sido agregar herramientas que reducen la cantidad de código que necesita ser escrito para tareas especificas, para acelerar el desarrollo.

En ese sentido, Godot ha tomado algunas decisiones de diseño útiles con ese objetivo. La primera y mas importante es el sistema de escenas. El objetivo del mismo no es obvio al principio, pero trabaja bien mas tarde. Esto es, descargar a los programadores de la responsabilidad de la arquitectura del código.

Cuando se diseñan juegos usando el sistema de escenas, el proyecto entero esta fragmentado en escenas complementarias (no individuales). Las escenas se complementar entre si, en lugar de estar separadas. Tendremos un montón de ejemplos sobre esto mas tarde, pero es muy importante recordarlo.

Para aquellos con una buena base de programación, esto significa que un patrón de diseño diferente a MVC(modelo-vista-controlador). Godot promete eficiencia al costo de dejar los hábitos MVC, los cuales se reemplazan por el patrón *escenas como complementos*.

Godot también utiliza el <http://c2.com/cgi/wiki?EmbedVsExtend>‘__ patrones para scripting, por lo que los scripts se extienden desde todas las clases disponibles.

1.4.2 GDScript

GDScript es un lenguaje de scripting de tipado dinámico hecho a medida de Godot. Fue diseñado con los siguientes objetivos:

- El primero y mas importante, hacerlo simple, familiar y fácil, tan fácil de aprender como sea posible.
- Hacer el código legible y libre de errores. La sintaxis es principalmente extraída de Python.

A los programadores generalmente les toma unos días aprenderlo, y entre las primeras dos semanas para sentirse cómodos con el.

Como con la mayoría de los lenguajes de tipado dinámico, la mayor productividad (el código es mas fácil de aprender, mas rápido de escribir, no hay compilación, etc) es balanceada con una pena de rendimiento, pero el código mas critico esta escrito en C++ en primer lugar dentro del motor (vector ops, physics, match, indexing, etc), haciendo que la rendimiento resultante sea mas que suficiente para la mayoría de los juegos.

En cualquier caso, si se requiere rendimiento, secciones criticas pueden ser reescritas en C++ y expuestas transparentemente al script. Esto permite reemplazar una clase GDScript con una clase C++ sin alterar el resto del juego.

1.4.3 Scripting de una Escena

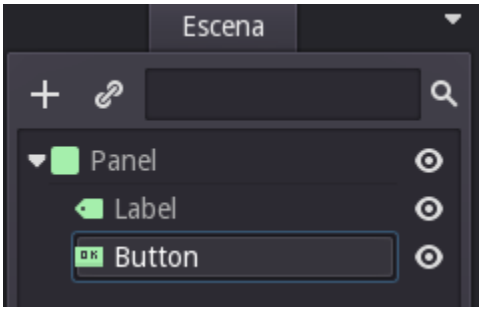
Antes de continuar, por favor asegúrate de leer la referencia *GDScript* Es un lenguaje simple y la referencia es corta, no debería llevar mas que algunos minutos darle un vistazo.

Configuración de la Escena

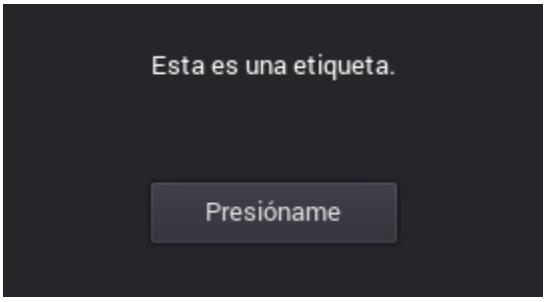
Este tutorial comenzara programando una simple escena. Usa el botón de agregar nodo (+) para crear la siguiente jerarquía, con los siguientes nodos:

- Panel
  - Label
  - Button

Debería verse así en el árbol de la escena:



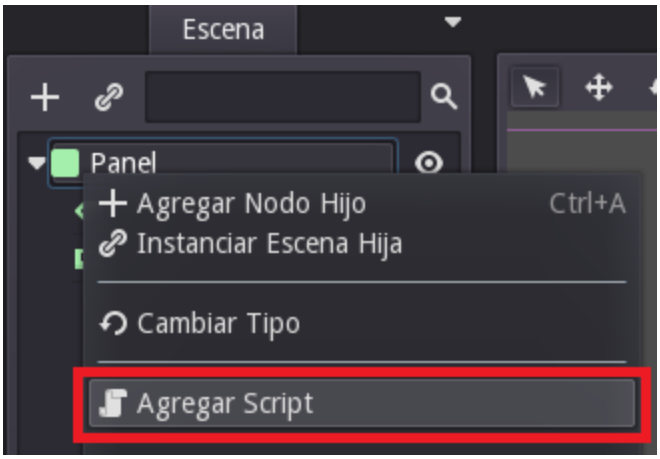
Y trata de que quede así en el editor 2D, para que tenga sentido:



Finalmente, guarda la escena, un nombre acorde podría ser “dihola.scn”

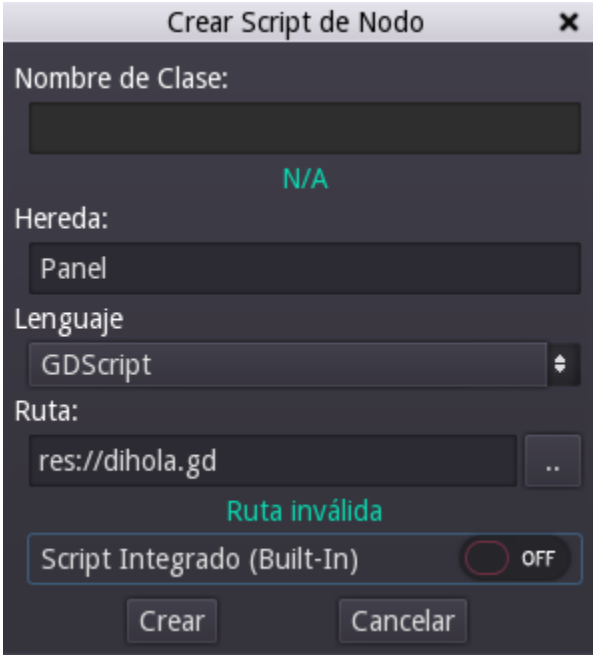
Agregando un script

Selecciona el nodo del Panel, y presiona click derecho en el mouse, luego selecciona Agregar Script:

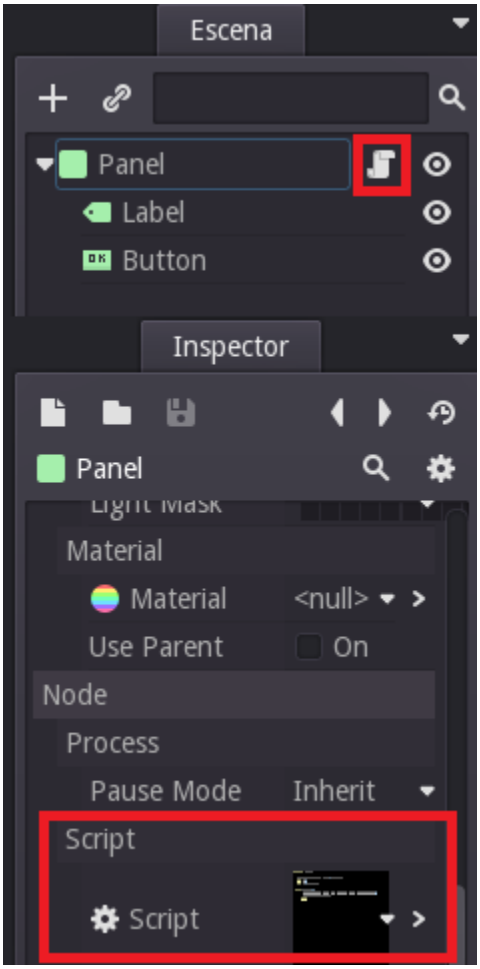


El dialogo de creación de script aparecerá. Este dialogo permite seleccionar el lenguaje, nombre de clase, etc. GDScript no usa nombres de clase en los archivos de script, por lo que este campo no es editable. El script debería heredar de "Panel" (ya que su función es extender el nodo, que es de tipo Panel, esto se llena automáticamente de todas formas).

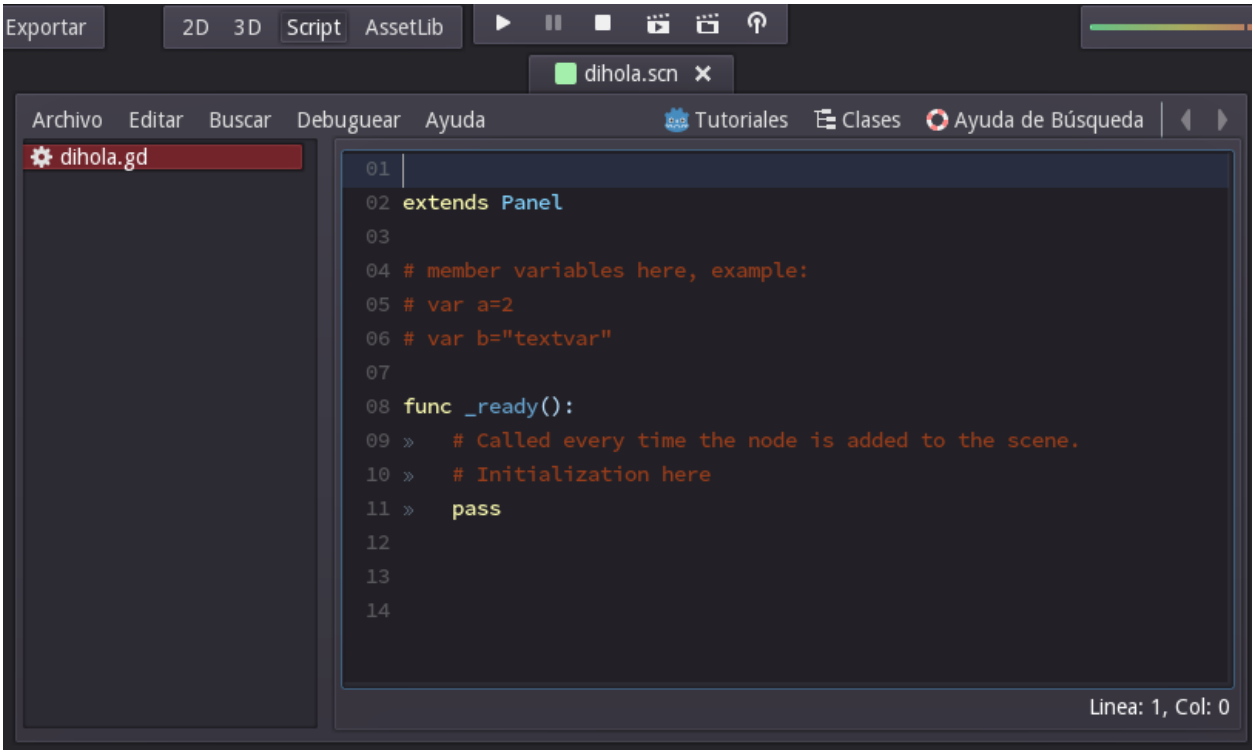
Selecciona el nombre de archivo para el script (si ya salvaste la escena previamente, uno se generara automáticamente como dihola.gd) y presiona "Crear":



Una vez hecho, el script se creara y se agregara al nodo. Puedes verlo tanto como el icono en el nodo, como en la propiedad script:



Para editar el script, presionar arriba del icono debería hacerlo (aunque, la UI(interfaz de usuario) te llevara directamente a la ventana de edicion de Script). Asique, aquí esta la plantilla del script:



No hay mucho allí. La función "_ready()" es llamada cuando el nodo (y todos sus hijos) entran en la escena activa. (Recuerda, no es un constructor, el constructor es "_init()").

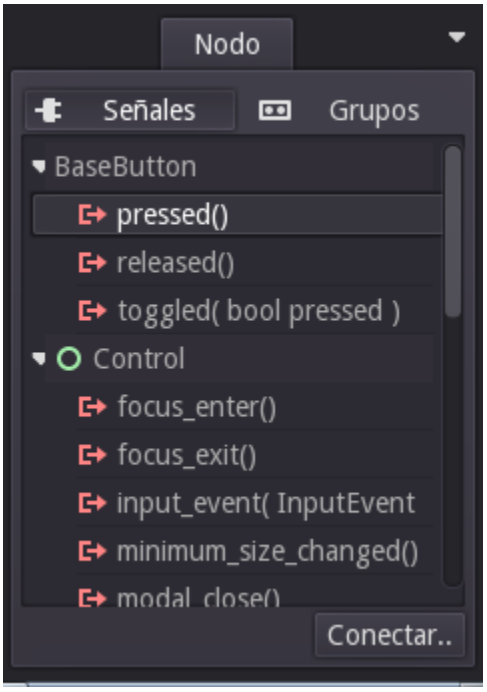
El rol del script

Un script básicamente agrega un comportamiento al nodo. Es usado para controlar las funciones del nodo así como otros nodos (hijos, padres, primos, etc). El alcance local del script es el nodo (como en cualquier herencia) y las funciones virtuales del nodo son capturadas por el script.

Manipulando una señal

Las señales son usadas principalmente en los nodos GUI(interfaz grafica de usuario) (aunque otros nodos también las tienen). Las señales se emiten cuando una acción especifica sucede, y pueden estar conectadas a cualquier otra función en cualquier de cualquier instancia de script. En este paso, la señal “pressed” del botón será conectada a una función personalizada.

En la pestaña “Nodo” puedes ver las señales disponibles para el nodo seleccionado:



Pero este ejemplo no lo usara. No queremos hacer las cosas *demasiado* fáciles. Asique por favor, cierra esa pantalla!

En cualquier caso, a esta altura es claro que estamos interesados en la señal "pressed"(presionado), asique en lugar de hacerlo con la interfaz visual, la conexión será hecha por código.

Para esto, existe una función que es probablemente la que los programadores de Godot usaran mas, esta es *Node.get_node()*. Esta función usa caminos para traer nodos en el árbol actual o en cualquier parte de la escena, relativa al nodo que posee el script.

Para traer el botón, lo siguiente debe ser utilizado:

```
get_node("Button")
```

Entonces, a continuación, un callback(llamada de retorno) será agregado cuando el botón sea presionado, que cambiara el texto de la etiqueta:

```
func _on_button_pressed():
    get_node("Label").set_text("HELLO!")
```

Finalmente, la señal "pressed" sera conectada al callback en _ready(), usando *Object.connect()*.

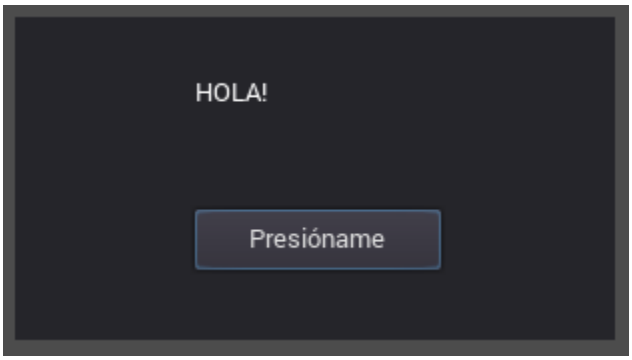
```
func _ready():
    get_node("Button").connect("pressed",self,"_on_button_pressed")
```

El script final debería verse así:

```
extends Panel

# member variables here, example:

# var a=2
# var b="textvar"

func _on_button_pressed():
    get_node("Label").set_text("HOLA!")
```

```
func _ready():
    get_node("Button").connect("pressed",self,"_on_button_pressed")
```

Correr la escena debería tener el resultado esperado cuando se presiona el botón:



Nota: Ya que es un error común en este tutorial, clarifiquemos nuevamente que get_node(camino) funciona regresando el hijo *inmediato* del nodo que es controlado por el script (en este caso, *Panel*), por lo que *Button* debe ser un hijo de *Panel* para que el código anterior funcione. Para darle mas contexto a esta aclaración, si *Button* fuese hijo de *Label*, el código para obtenerlo sería:

```
# not for this case
# but just in case
get_node("Label/Button")
```

Y, también, trata de recordar que los nodos son referenciados por nombre, no por tipo.

1.5 Scripting (continuación)

1.5.1 Procesando

Varias acciones en godot son disparadas por callbacks o funciones virtuales, por lo que no hay necesidad de escribir código de chequeo que corre todo el tiempo. Además, mucho puede ser hecho con animation players (reproductores de animación).

Sin embargo, es aun un caso muy común tener un script procesando en cada frame. Hay dos tipos de procesamiento, procesamiento idle(inactivo) y procesamiento fixed(fijo).

El procesamiento Idle es activado con la funcion *Node.set_process()* Una vez activado, el callback *Node._process()* podrá ser llamado en cada frame(cuadro). Ejemplo:

```
func _ready():
    set_process(true)

func _process(delta):
    # hacer algo...
```

El parámetro delta describe el tiempo que paso (en segundos, como numero de punto flotante) desde la llamada previa a la funcion _process(). El procesamiento fijo es similar, pero solo se necesita para sincronización con el motor de física.

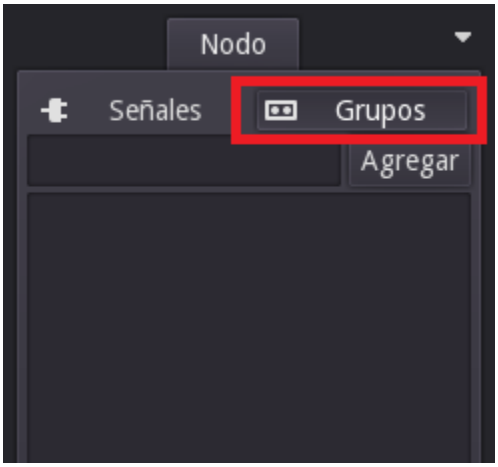
Una forma simple de probar esto es crear una escena con un solo nodo Label, con el siguiente script:

```
extends Label

var accum=0

func _ready():
    set_process(true)

func _process(delta):
    accum += delta
    set_text(str(accum))
```

Lo que mostrara un contador aumentando cada segundo.

1.5.2 Grupos

Los nodos pueden ser agregados a grupos (tantos como se desee por nodo). Esta es una característica simple pero efectiva para organizar escenas grandes. Hay dos formas de hacer esto, la primera es por la UI, con el botón Grupos en la pestaña Nodo.



Y la segunda desde el código. Un ejemplo útil podría ser, por ejemplo, marcar escenas que son enemigos.

```
func _ready():
    add_to_group("enemigos")
```

De esta forma, si el jugador, entrando sigilosamente a la base secreta, es descubierto, todos los enemigos pueden ser notificados sobre la alarma activada, usando *SceneTree.call_group()*:

```
func _on_discovered():
    get_tree().call_group(0, "guardias", "jugador_fue_descubierto")
```

El código superior llama la función "jugador_fue_descubierto" en cada miembro del grupo "guardias".

Opcionalmente, es posible obtener la lista completa de nodos "guardias" llamando a *SceneTree.get_nodes_in_group()*:

```
var guardias = get_tree().get_nodes_in_group("guardias")
```

Luego agregaremos mas sobre *SceneTree*

1.5.3 Notificaciones

Godot utiliza un sistema de notificaciones. Usualmente no son necesarias desde scripts, debido a que es demasiado bajo nivel y las funciones virtuales están disponibles para la mayoría de ellas. Es solo que es bueno saber que existen. Simplemente agrega una funcion *Object._notification()* en tu script:

```
func notificacion (what):
    if (what == NOTIFICATION_READY):
        print("Esto es lo mismo que sobrescribir _ready()...")
    elif (what == NOTIFICATION_PROCESS):
        var delta = get_process_time()
        print("Esto es lo mismo que sobrescribir _process()...")
```

La documentación de cada clase en *Class Reference* muestra las notificaciones que puede recibir. Sin embargo, nuevamente, para la mayoría de los casos los scripts proveen funciones mas simples Sobreescribibles.

1.5.4 Funciones Sobreescribibles

Como mencionamos antes, es mejor usar estas funciones. Los nodos proveen muchas funciones sobreescribibles útiles, las cuales se describen a continuación:

```
func _enter_tree():
    # Cuando el nodo entre en la _Scene Tree_. se vuelve activa
    # y esta función se llama. Los nodos hijos aun no entraron
    # la escena activa. En general, es mejor usar _ready()
    # para la mayoría de los casos.
    pass

func _ready():
    # Esta función es llamada luego de _enter_tree, pero se
    # aseguro que todos los nodos hijos también hayan entrado
    # a _Scene Tree_, y se volvieron activas.
    pass

func _exit_tree():
    # Cuando el nodo sale de _Scene Tree_. esta funcion es
    # llamada. Los nodos hijos han salido todos de _Scene Tree_
    # en este punto y todos están activos.
    pass

func _process(delta):
    # Cuando set_process() esta habilitado, esta función es
    # llamada en cada frame.
    pass

func _fixed_process(delta):
    # Cuando set_fixed_process() esta habilitado, esto es
    # llamado en cada frame de física.
    pass

func _paused():
    # Se llama cuando el juego esta en pausa, Luego de esta
    # llamada, el nodo no recibirá mas callbacks de proceso.
    pass

func _unpaused():
```

```
    # Llamada cuando el juego se reanuda.
    pass
```

1.5.5 Creando nodos

Para crear nodos desde código, solo llama el método .new(), (al igual que para cualquier otra clase basada en tipo de dato). Ejemplo:

```
var s
func _ready():
    s = Sprite.new() # crear un nuevo sprite!
    add_child(s) # lo agrega como hijo de este nodo
```

Para borrar el nodo, sea dentro o fuera de la escena, free() debe ser usado:

```
func _someaction():
    s.free() # inmediatamente remueve el nodo de la escena y
             # lo libera
```

Cuando un nodo es liberado, también son liberados todos los nodos hijos. Por este motivo, borrar nodos manualmente es mucho mas simple de lo que parece. Solo libera el nodo base y todo lo demás en el sub árbol se ira con el.

Sin embargo, puede suceder muy seguido que queramos borrar un nodo que esta actualmente "blocked"(bloqueado), esto significa, el nodo esta emitiendo una señal o llamado a función. Esto resultara en que el juego se cuelgue. Correr Godot en el debugger (depurador) a menudo va a capturar este caso y advertirte sobre el.

La forma mas segura de borrar un nodo es usando *Node.queue_free()* en su lugar. Esto borrara el nodo mientras esta inactivo, de forma segura.

```
func _someaction():
    s.queue_free() # remueve el nodo y lo borra mientras nada esta
    sucediendo.
```

1.5.6 Instanciando escenas

Instancias una escena desde código es bastante fácil y se hace en dos pasos. El primero es cargar la escena desde el disco.

```
var scene = load("res://myscene.scn") # cargara cuando el script es
instanciado
```

Precargar es mas conveniente a veces, ya que sucede en tiempo de parse (análisis gramatical).

```
var scene = preload("res://myscene.scn") # será cargado cuando el
                                         # script es "parseado"
```

Pero 'escena' todavía no es un nodo que contiene sub nodos. Esta empaquetado en un recurso especial llamado *PackedScene*. Para crear el nodo en si, la función *PackedScene.instance()* debe ser llamada. Esta regresara el árbol de nodos que puede ser agregado a la escena activa:

```
var node = scene.instance()
add_child(node)
```

La ventaja de este proceso en dos pasos es que una escena empaquetada puede mantenerse cargada y listo para usar, por lo que puede ser usada para crear tantas instancias como se quiera. Esto es especialmente útil, por ejemplo, para instanciar varios enemigos, armas, etc. de forma rápida en la escena activa.

1.6 Juego 2D Simple

1.6.1 Pong

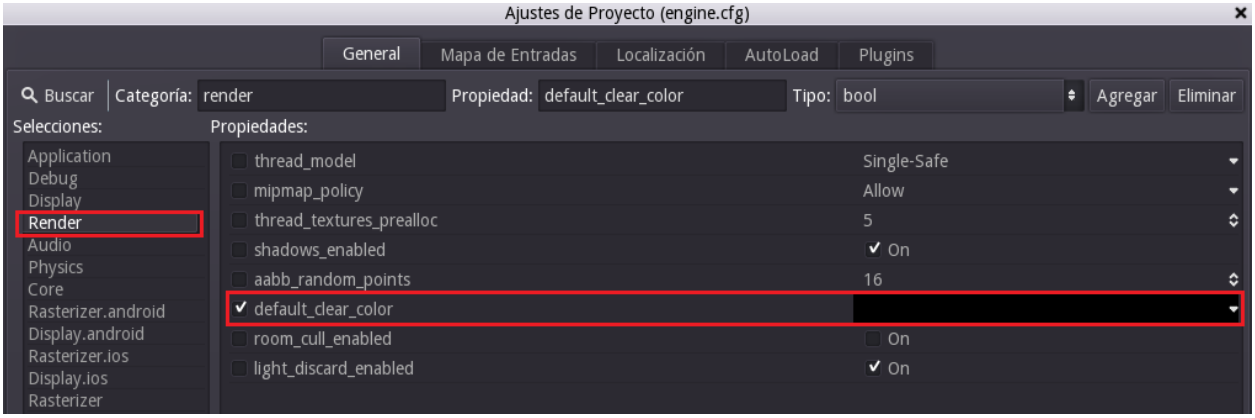
En este sencillo tutorial, un juego básico de Pong será creado. Hay un montón de ejemplos mas complejos que de pueden descargar desde el sitio oficial de Godot, pero esto debería servir como introducción a la funcionalidad básica para juegos 2D.

1.6.2 Assets (activos)

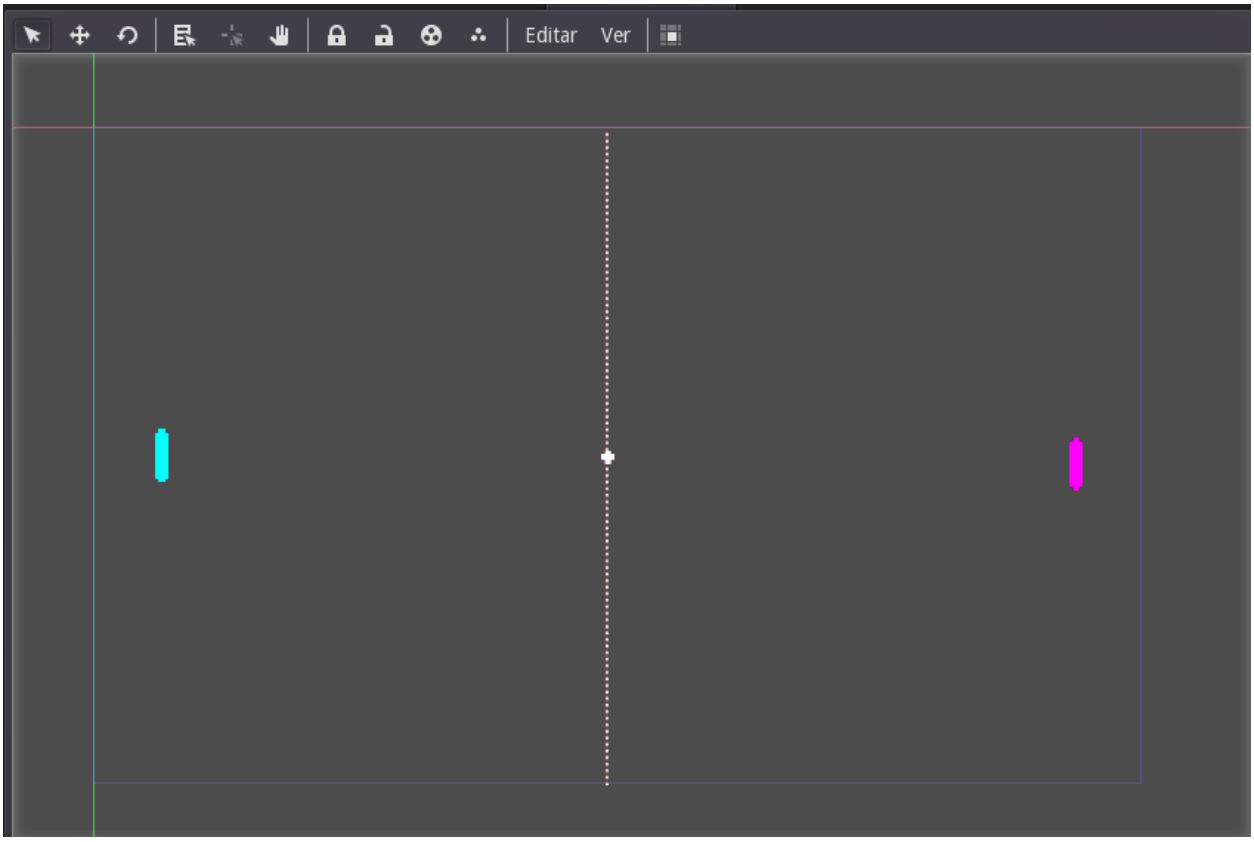
Algunos assets son necesarios para este tutorial: `pong_assets.zip`.

1.6.3 Configuración de la escena

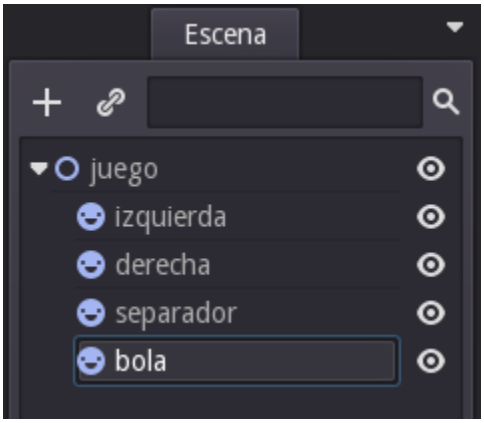
Para recordar viejos tiempos, el juego tendrá una resolución de 640x400 pixels. Esto puede ser configurado en Configuración de Proyecto (ve *Configurando el proyecto*). El color de fondo debe ajustarse a negro.



Crea un nodo :ref:‘class_Node2D‘como raíz del proyecto. Node2D es el tipo base para el motor 2D. Luego de esto, agrega algunos sprites (:ref:‘class_Sprite‘node) y ajusta cada uno a su textura correspondiente. El diseño de la escena final debe verse similar a esto (nota: la pelota esta en el medio!):



El árbol de escena, entonces, luce similar a esto:



Guarda la escena como “pong.scn” y ajusta la escena principal en las propiedades del proyecto.

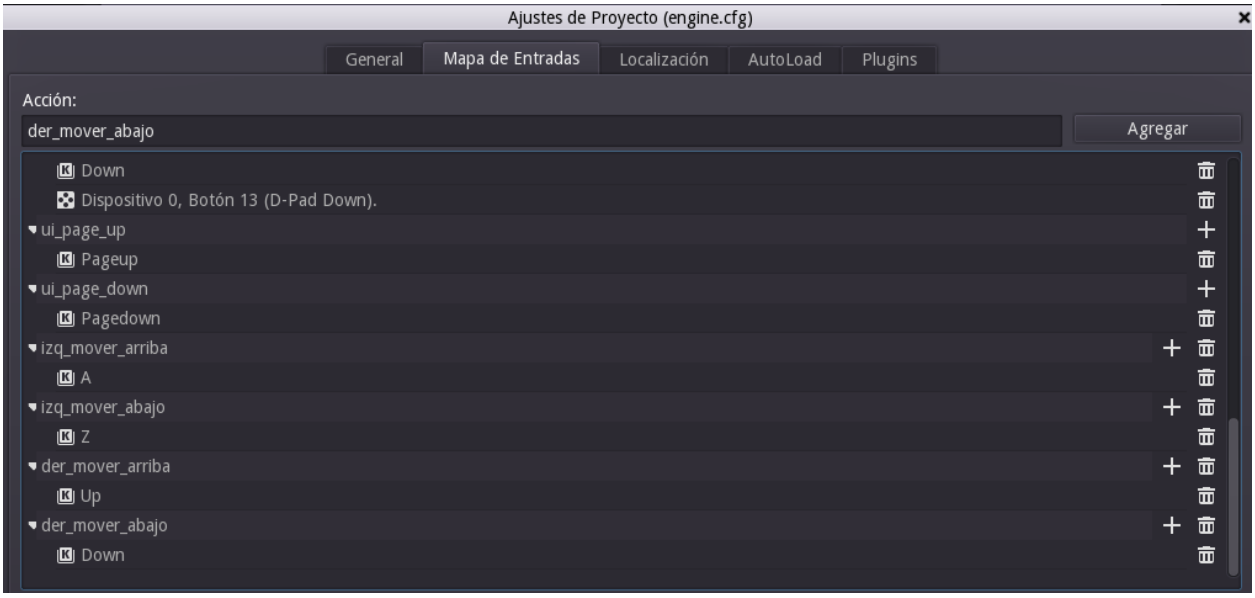
1.6.4 Configuración de acciones de entrada

Hay tantos métodos de entrada para videojuegos... Teclado, Pad, Mouse, Pantalla táctil (Multitouch). Pero esto es pong. El único control que importa es que los pads vayan arriba y abajo.

Manejar todos los posibles métodos de entrada puede ser muy frustrante y tomar un montón de código. El hecho de que la mayoría de los juegos permiten personalizar los controles hacen que este problema empeore. Para esto, Godot creo las “Acciones de Entrada”. Una acción se define, luego se agregan métodos de entrada que disparan esa acción.

Abre el dialogo de propiedades de proyecto nuevamente, pero esta vez ve a la pestaña “Mapa de entradas”.

Allí, agrega 4 acciones: `izq_mover_arriba`, `izq_mover_abajo`, `der_mover_arriba`, `der_mover_abajo`. Asigna las teclas que deseas. A/Z (para el jugador de la izquierda) y Flecha arriba/Flecha abajo (para el jugador de la derecha) como teclas debería funcionar en la mayoría de los casos.



1.6.5 Script

Crea un script para el nodo raíz de la escena y ábrelo (como se explica en *Agregando un script*). El script heredara Node2D:

```
extends Node2D

func _ready():
    pass
```

En el constructor, se harán 2 cosas. Primero es habilitar el procesamiento, y la segunda guardar algunos valores útiles. Esos valores son las dimensiones de la pantalla y el pad:

```
extends Node2D

var pantalla_tamano
var pad_tamano

func _ready():
    pantalla_tamano = get_viewport_rect().size
    pad_tamano = get_node("izquierda").get_texture().get_size()
    set_process(true)
```

Luego, algunas variables usadas para el procesamiento dentro del juego serán agregadas:

```
#velocidad de la bola (en pixeles/segundo)
var bola_velocidad = 80

#dirección de la bola (vector normal)
var direccion = Vector2(-1, 0)

#constante para la velocidad de los pads (también en
```

```
# pixeles/segundo)

const PAD_VELOCIDAD = 150
```

Finalmente, la función de procesamiento:

```
func _process(delta):
```

Toma algunos valores útiles para computar. La primera es la posición de la bola (desde el nodo), la segunda es el rectángulo (`Rect2`) para cada uno de los pads. Los sprites tienen sus texturas centradas por defecto, por lo que un pequeño ajuste de `pad_size / 2` debe ser agregado.

```
var bola_posicion = get_node("bola").get_pos()
var rect_izq = Rect2( get_node("izquierda").get_pos() - pad_tamano/2,pad_tamano)
var rect_der = Rect2 ( get_node("derecha").get_pos() - pad_tamano/2,pad_tamano)
```

Debido a que la posición de la bola ha sido obtenida, integrarla debería ser simple:

```
bola_posicion += direccion * bola_velocidad * delta
```

Luego, ahora que la bola tiene una nueva posición, debería ser probada contra todo. Primero, el piso y el techo:

```
if ( (bola_posicion.y < 0 and direccion.y < 0) or (bola_posicion.y > pantalla_tamano.
↪y and direccion.y > 0)):
    direccion.y = -direccion.y
```

Si se toco uno de los pads, cambiar la dirección e incrementar la velocidad un poco

```
if ( (rect_izq.has_point(posicion_bola) and direccion.x < 0) or (rect_der.has_
↪point(bola_posicion) and direccion.x > 0)):
    direccion.x = -direccion.x
    bola_velocidad *= 1.1
    direccion.y = randf() * 2.0 - 1
    direccion = direccion.normalized()
```

Si la bola sale de la pantalla, el juego termina. Luego se reinicia:

```
if (bola_posicion.x < 0 or bola_posicion.x > pantalla_tamano.x):
    bola_posicion = pantalla_tamano * 0.5  # la bola va al centro de la pantalla
    bola_velocidad = 80
    direccion = Vector2 (-1, 0)
```

Una vez que todo fue hecho con la bola, el nodo es actualizado con la nueva posición:

```
get_node("bola").set_pos(bola_posicion)
```

Solo actualizar los pads de acuerdo a la entrada del jugador. La clase Input es realmente útil aquí:

```
#mover pad izquierdo
var izq_posicion = get_node("izquierda").get_pos()

if (izq_posicion.y > 0 and Input.is_action_pressed("izq_mover_arriba")):
    izq_posicion.y += -PAD_SPEED * delta
if (izq_posicion.y < pantalla_tamano.y and Input.is_action_pressed("izq_mover_abajo
↪")):
    izq_posicion.y += PAD_SPEED * delta
```

```
get_node("izquierda").set_pos(izq_posicion)

#mover pad derecho
var der_posicion = get_node("derecha").get_pos()

if (der_posicion.y > 0 and Input.is_action_pressed("der_mover_arriba"))
    der_posicion.y += -PAD_SPEED * delta
if (der_posicion.y < pantalla_tamano.y and Input.is_action_pressed("der_mover_abajo
↪")):
    der_posicion.y += PAD_SPEED * delta

get_node("derecha").set_pos(der_posicion)
```

Y eso es todo! Un simple Pong fue escrito con unas pocas líneas de código.

1.7 Tutorial GUI (Interfaz Grafica de Usuario)

1.7.1 Introduccion

Si hay algo que la mayoria de los programadores realmente odian, es programar interfaces graficas de usuario (GUIs). Es aburrido, tedioso y no ofrece retos. Varios aspectos hacen este problema peor como:

- La alineacion de los elementos de la UI (interfaz de usuario) es dificil (para que se vea justo como el diseñador queria).
- Las UIs son cambiadas constantemente debido a los problemas de apariencia y usabilidad que aparecen durante el testing(prueba).
- Manejar apropiadamente el cambio de tamaño de pantalla para diferentes resoluciones de pantallas.
- Animar varios componentes de pantalla, para hacerlo parecer menos estatico.

La programacion GUI es ona de las principales causas de frustracion de los programadores. Durante el desarrollo de Godot (y previas iteraciones del motor), varias tecnicas y filosoficas para el desarrollo UI fueron puestas en practica, como un modo inmediato, contenedores, anclas, scripting, etc. Esto fue siempre hecho con el objetivo principal de reducir el estres que los programadores tienen que enfrentar cuando crear interfaces de usuario

Al final, el sistema UI resultante en Godot es una eficiente solucion para este problema, y funciona mezclando juntos algunos enfoques. Mientras que la curva de aprendizaje es un poco mas pronunciada que en otros conjuntos de herramientas, los desarrolladores pueden crear interfaces de usuario complejas en muy poco tiempo, al compartir las mismas herramientas con diseñadores y animadores.

1.7.2 Control

El nodo basico para elementos UI es *Control* (a veces llamados “Widget” o “Caja” en otras herramientas). Cada nodo que provee funcionalidad de interfaz de usuario desciende de el.

Cuando los controles son puestos en el arbol de escena como hijos de otro control, sus coordenadas (posicion, tamaño) son siempre relativas a sus padres. Esto aporta la base para editar interfaces de usuario complejas rapidamente y de manera visual.

1.7.3 Entrada y dibujado

Los controles reciben eventos de entrada a traves de la llamada de retorno *Control._input_event()*. Solo un control, el que tiene el foco, va a recibir los eventos de teclado/joypad (ve *Control.set_focus_mode()* y *Control.grab_focus()*.)

Los eventos de movimiento de mouse son recibidos por el control que esta directamente debajo del puntero de mouse. Cuando un control recibe el evento de que se presiono un boton de mouse, todas los eventos siguientes de movimiento son recibidos por el control presionado hasta que el boton se suelta, aun si el puntero se mueve fuera de los limites del control.

Como cualquier clase que hereda de *CanvasItem* (Control lo hace), una llamada de retorno *CanvasItem._draw()* sera recibida al principio y cada vez que el control deba ser redibujado (los programadores deben llamar *CanvasItem.update()* para poner en cola el CanvasItem para redibujar). Si el control no esta visible (otra propiedad CanvasItem), el control no recibe ninguna entrada.

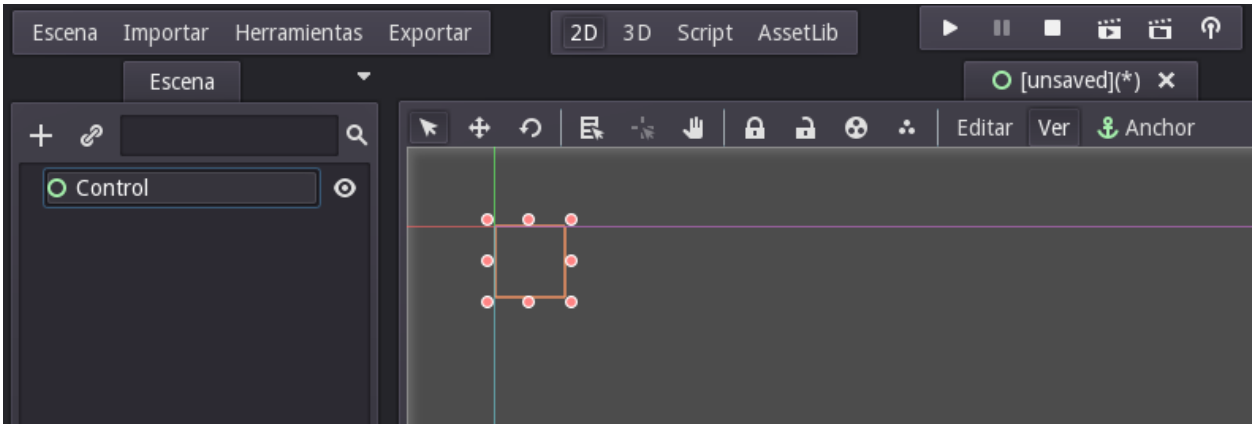
En general sin embargo, el programador no necesita lidiar con el dibujado y los eventos de entrada directamente cuando se construyen UIs, (es mas util cuando se crean controles personalizados). En su lugar, los controles emiten diferente tipos de señales con informacion contextual para cuando la accion ocurre. Por ejemplo, una *Button* emite una señal "pressed", una :ref:'Slider <class_Slider>'emitiraun "value_changed" cuando se arrastra, etc.

1.7.4 Mini tutorial de controles personalizados

Antes de ir mas profundo, crear un control personalizado sera una buena forma de entender como funcionan los controles, ya que no son tan complejos como pueden parecer.

Adicionalmente, aunque Godot viene con docenas de controles para diferentes propositos, sucede a menudo que es simplemente mas sencillo obtener la funcionalidad especidica creando uno nuevo.

Para comenzar, crea una escena con un solo nodo. El nodo es del tipo "Control" y tiene cierta area de la pantalla en el editor 2D, como esto:



Agregale un script a ese nodo, con el siguiente codigo:

```
extends Control

var pulsado=false

func _draw():

    var r = Rect2( Vector2(), get_size() )
    if (pulsado):
        draw_rect(r, Color(1,0,0) )
    else:
```

```
        draw_rect(r, Color(0,0,1) )

func _input_event(ev):

    if (ev.type==InputEvent.MOUSE_BUTTON and ev.pressed):
        pulsado=true
        update()
```

Luego corre la escena. Cuando el rectangulo es clickeado/pulsado, ira de azul a rojo. Esa sinergia entre los eventos y el dibujo es basicamente como funcionan internamente la mayoria de los controles.

1.7.5 Complejidad de la UI

Como mencionamos antes, Godot incluye docenas de controles listos para usarse en una interface. Esos controles estan divididos en dos categorias. La primera es un pequeño grupo de controles que funcionan bien para crear la mayoria de las interfaces de usuario. La segunda (y la mayoria de los controles son de este tipo) estan destinadas a interfases de usuario complejas y el skinning(aplicar un forro) a traves de estilos. Una descripcion es presentada a continuacion para ayudar a entender cual debe ser usada en que caso.

1.7.6 Controles UI simplificados

Este conjunto de controles es suficiente para la mayoria de los juegis, donde interacciones complejas o formas de presentar la informacion no son necesarios. Pueden ser "skineados" facilmente con texturas regulares.

- *Label*: Nodo usado para mostrar texto
- *TextureFrame*: Muestra una sola textura, que puede ser escalada o mantenia fija.
- *TextureButton*: Muestra una simple boton con textura para los estados como pressed, hover, disabled, etc.
- *TextureProgress*: Muestra una sola barra de progreso con textura.

Adicionalmente, el reposicionado de controles es mas eficientemente hecho con anclas en este caso (ve el tutorial *Tamaño y anclas* para mas informacion)

De cualqueir forma, sucedera seguido que aun para juegos simples, comportamientos de UI mas complejos son requeridos. Un ejemplo de esto una lista de elemenots con scrolling (desplazamiento) (por ejemplo para una tabla de puntuaciones altas), la cual necesita un *ScrollContainer* y un *VBoxContainer*. Este tipo de controles mas avanzados puede ser mezclado con los regulares sin problema (son todos controles de todas formas).

1.7.7 Controles de UI complejos

El resto de los controles (y hay docenas de ellos!) estan destinados para otro tipo de escenario, los mas comunes:

- Juegos que requieren UIs complejas, como RPGs (juegos de rol), MMOs (juegos online masivos), strategy (estrategia), sims (simulacion), etc.
- Crear herramientas de desarrollo personalizadas para acelerar la creacion de contenido.
- Crear Plugins de Editor de Godot, para extender la funcionalidad del motor.

Reposicionar controles para este tipo de interfaces es mas comunmente hecho con contenedores (ve el tutorial *Tamaño y anclas* para mas informacion).

1.8 Pantalla de bienvenida (Splash Screen)

1.8.1 Tutorial

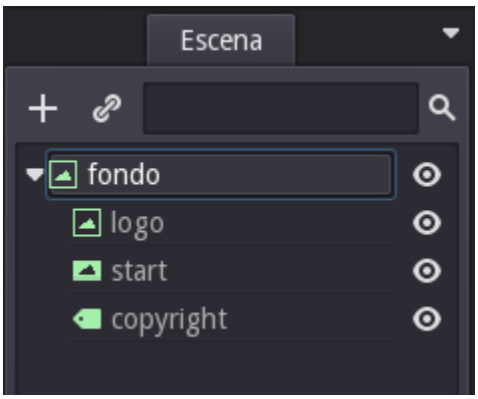
Este será un tutorial simple para cementar la idea básica de como el subsistema GUI funciona. El objetivo será crear una pantalla de bienvenida realmente simple y estática.

A continuación hay un archivo con los assets que serán usados. Estos pueden ser agregados directamente a tu carpeta de proyecto, no hay necesidad de importarlos.

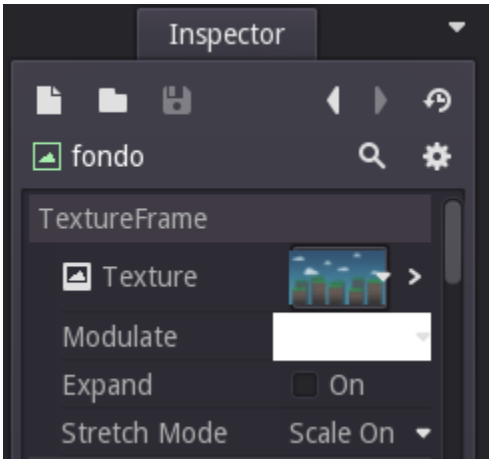
`robisplash_assets.zip`.

1.8.2 Configurando

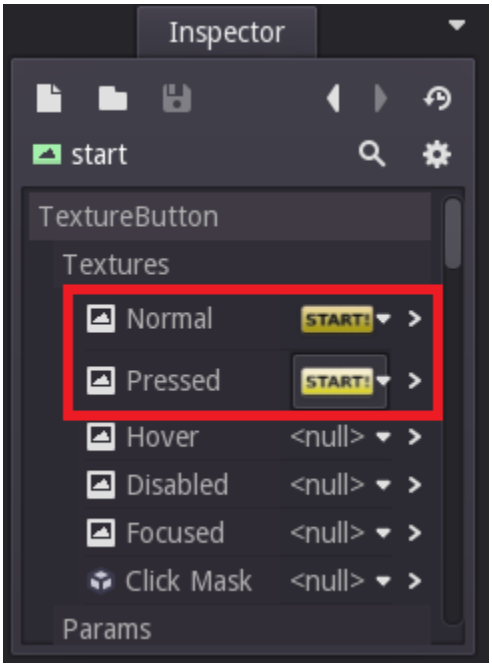
Fija la resolución de pantalla en 800x450 en la configuración de proyecto, y prepara una nueva escena como esta:



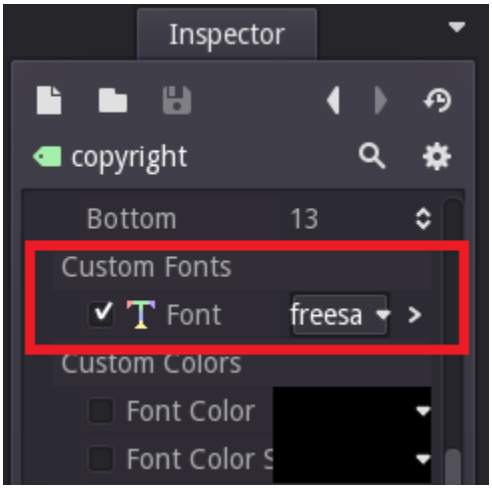
Los nodos "fondo" y "logo" son del tipo *TextureFrame*. Estos tienen una propiedad especial para configurar la textura a ser mostrada, solo carga el archivo correspondiente.



El nodo "start" es un *TextureButton*, que toma varias imágenes para diferentes estados, pero solo normal y pressed (presionado) serán proporcionados en este ejemplo:



Finalmente, el nodo "copyright" es una *Label*. Las etiquetas (Labels) pueden ser configuradas con una fuente personalizada editando la siguiente propiedad:



Como una nota aparte, la fuente fue importada de un TTF, ve *Importing fonts*.

1.9 Animaciones

1.9.1 Introducción

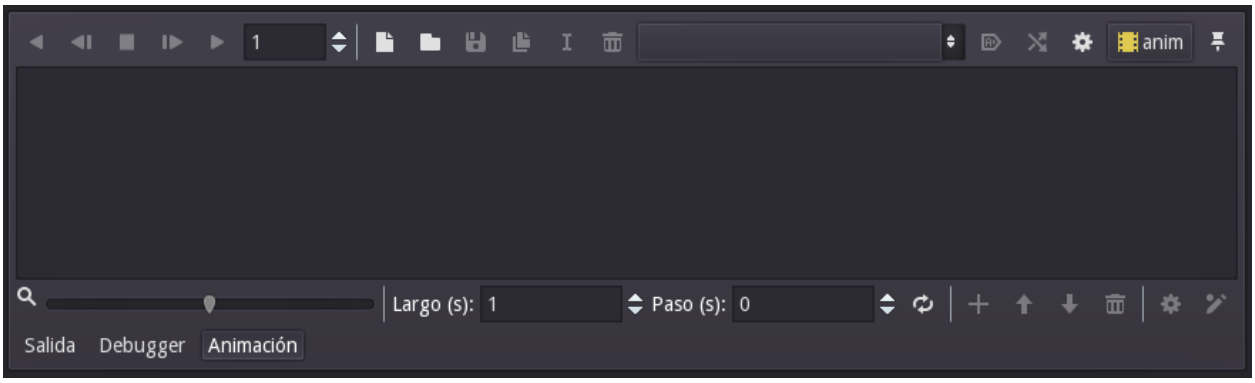
Este tutorial explicara como todo es animado en Godot. El sistema de animación de Godot es extremadamente poderoso y flexible.

Para empezar, vamos a usar la escena del tutorial previo (*Pantalla de bienvenida (Splash Screen)*). El objetivo es agregarle una animación simple. Aquí hay una copia del tutorial por las dudas: `robisplash.zip`.

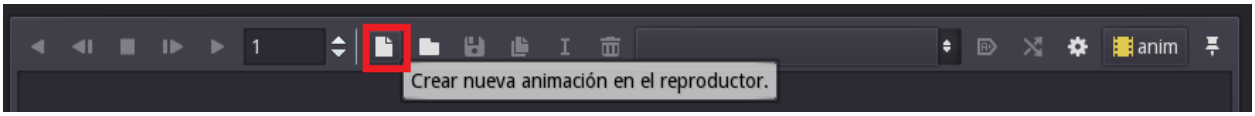
1.9.2 Creando la animación

Primero que nada, agrega un nodo *AnimationPlayer* a la escena como hijo del fondo (en nodo raiz):

Cuando un nodo de este tipo es seleccionado, el panel de edición de animaciones aparecerá:



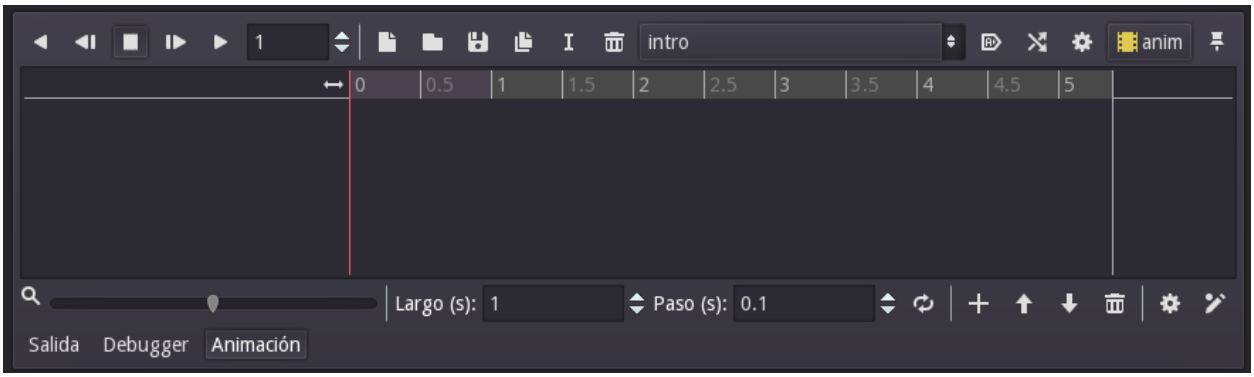
Asique, es tiempo de crear una nueva animación! Presiona el botón Nueva Animación y nómbrala "intro".



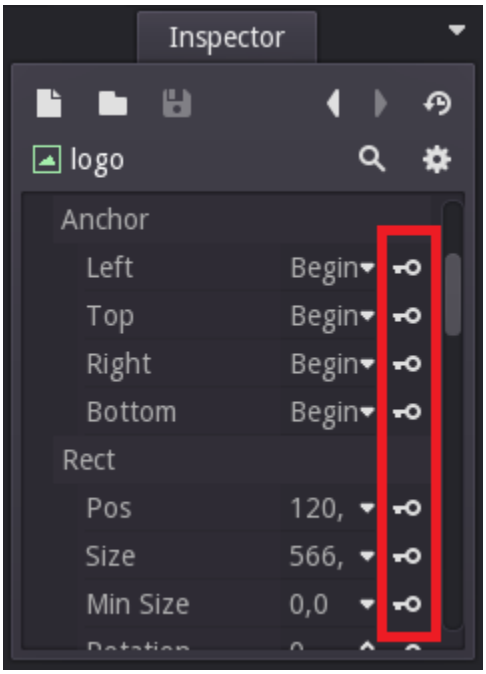
Luego de que la animación es creada, es tiempo de editarla.

1.9.3 Editando la animación

Ahora es cuando la magia sucede! Varias cosas suceden cuando se edita una animación, la primera es la aparicion del panel de edición de animación.



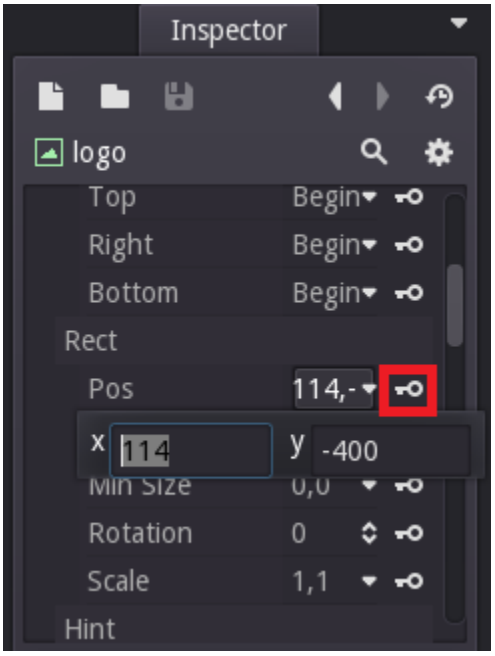
Pero la segunda, y mas importante, es que el Inspector entra en modo "animación". En este modo, un icono de llave aparece al lado de cada propiedad en el Inspector. Esto significa que, en Godot, *cualquier propiedad de cualquier objeto* puede ser animada.



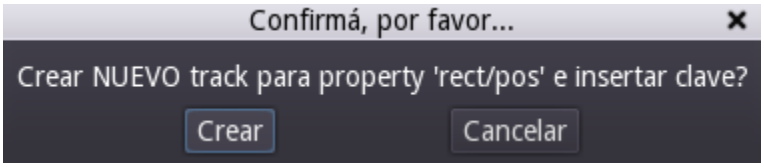
1.9.4 Haciendo que el logo aparezca

A continuación, haremos aparecer el logo desde la parte superior de la pantalla. Luego de seleccionar el reproductor de animaciones, el panel de edición se mantendrá visible hasta que sea manualmente escondido. Para tomar ventaja de esto, selecciónа el nodo "logo" y ve a la propiedad "pos" en el Inspector, muévela arriba, a la posición: 114, -400.

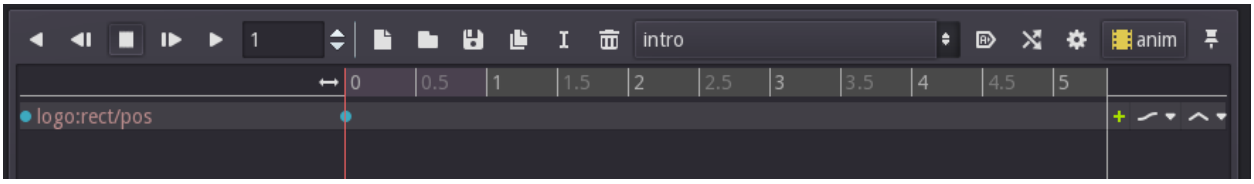
Una vez en esta posición, presiona el botón de llave al lado de la propiedad:



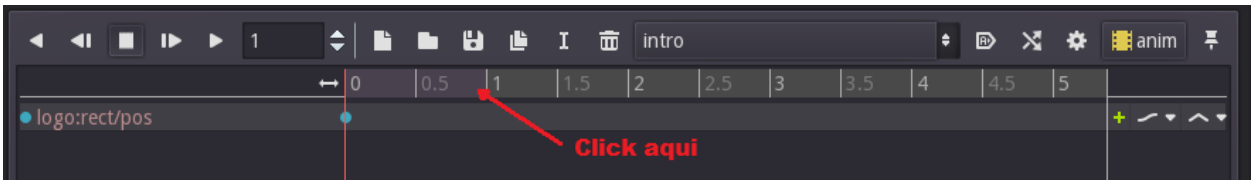
Como es un nuevo track (pista), un dialogo aparecerá preguntando para crearla. Confirma!



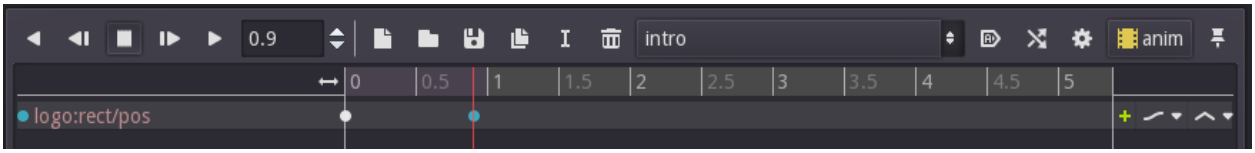
Asi el keyframe(fotograma clave) sera agregado en el editor de animación:



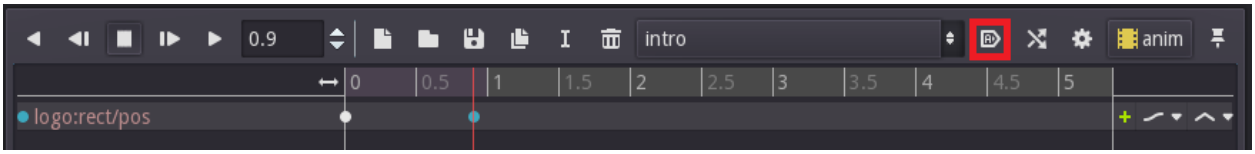
En segundo lugar, mueve el cursor del editor hasta el final, haciendo click aquí:



Cambia la posición del logo a 114,0 en el Inspector y agrega otro keyframe (haciendo click en la llave). Con dos keyframes, la animación sucede.



Pulsando Play en el panel de animación hará que el logo descienda. Para probarlo al correr la escena, el botón autoplay puede marcar la animación para que empiece automáticamente cuando la escena comienza:



Y finalmente, cuando corras la escena, la animación debería verse de esta forma:

1.10 Recursos

1.10.1 Nodos y recursos

Hasta ahora, los Nodos (*Nodes*) han sido el tipo de datos mas importante en Godot, ya que la mayoría de los comportamientos y características del motor están implementadas a través de estos. Hay, sin embargo, otro tipo de datos que es igual de importante. Son los recursos (*Resource*).

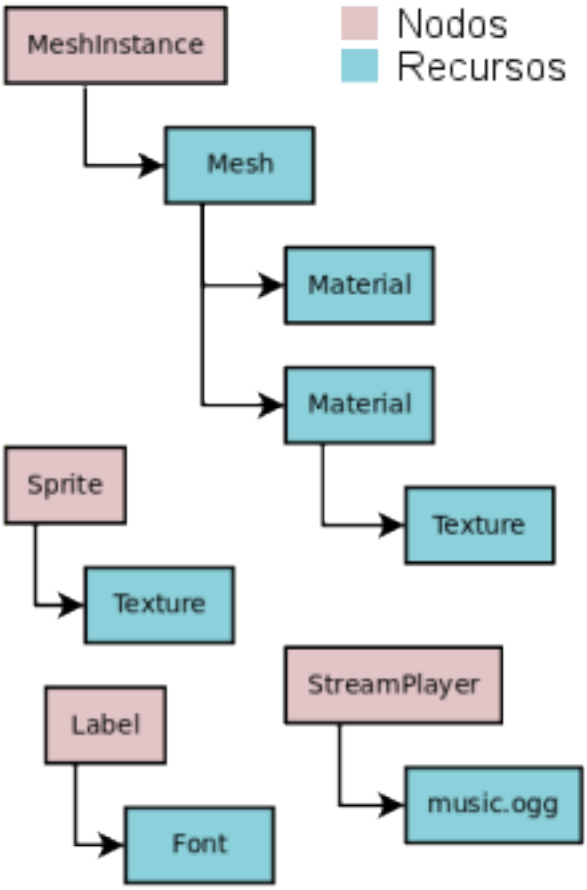
Mientras que los *Nodos* se enfocan en comportamiento, como dibujar un sprite, dibujar un modelo 3d, física, controles GUI, etc, los **Recursos** con meros **contenedores de datos**. Esto implica que no realizan ninguna acción ni procesan información. Los recursos solo contienen datos.

Ejemplos de recursos: *Texture*, *Script*, *Mesh*, *Animation*, *Sample*, *AudioStream*, *Font*, *Translation*, etc.

Cuando Godot guarda o carga (desde disco) una escena (.scn o .xml), una imagen (png, jpg), un script (.gd) o básicamente cualquier cosa, ese archivo es considerado un recurso.

Cuando un recurso es cargado desde disco, **siempre es cargado una sola vez**. Esto significa, que si hay una copia del recurso ya cargada en memoria, tratar de leer el recurso nuevamente va a devolver la misma copia una y otra vez. Esto debido al hecho de que los recursos son solo contenedores de datos, por lo que no hay necesidad de tenerlos duplicados.

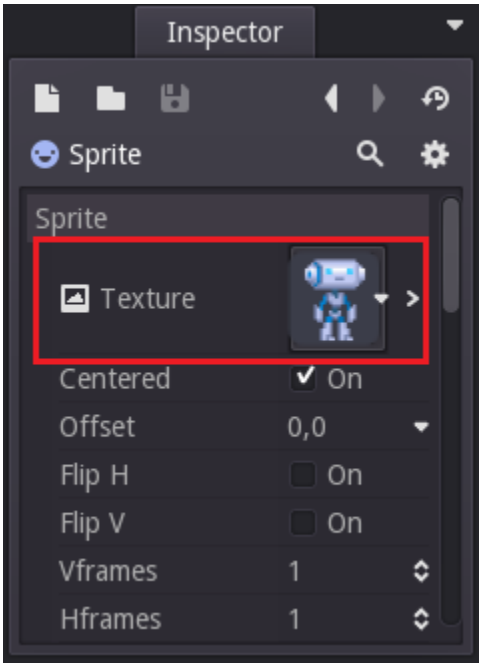
Típicamente, cada objeto en Godot (Nodo, Recurso, o cualquier cosa) puede exportar propiedades, las cuales pueden ser de muchos tipos (como un string o cadena, integer o numero entero, Vector2 o vector de 2 dimensiones, etc). y uno de esos tipos puede ser un recurso. Esto implica que ambos nodos y recursos pueden contener recursos como propiedades. Para hacerlo un poco mas visual:



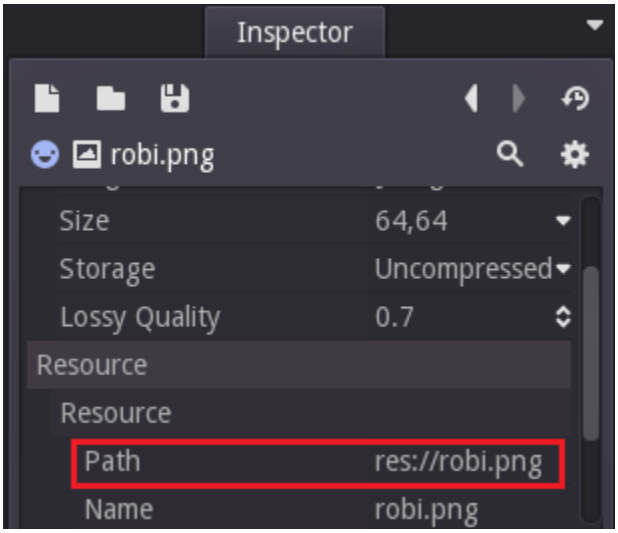
1.10.2 Externo vs Incorporado (Built-in)

Las propiedades de los recursos pueden referenciar recursos de dos maneras, *externos* (en disco) o **incorporados**.

Para ser mas especifico, aqui hay una *Texture* en un nodo *Sprite*:



Presionando el botón “>” a la derecha de la vista previa, permite ver y editar las propiedades del recurso. Una de las propiedades (path) muestra de donde vino. En este caso, desde una imagen png.



Cuando el recurso proviene de un archivo, es considerado un recurso *externo*. Si el la propiedad camino es borrada (o nunca tuvo), es considerado un recurso incorporado.

Por ejemplo, si el camino ‘’’res://robi.png‘‘‘ es borrado de la propiedad path en el ejemplo superior, y la escena es guardada, el recurso será guardado dentro del archivo de escena .scn, ya no referenciando externamente a “robi.png”. Sin embargo, aun si esta guardado de forma incorporada, y aunque la escena puede ser instanciada muchas veces, el recurso se seguirá leyendo siempre una vez. Eso significa que diferentes escenas del robot Robi que sean instanciadas al mismo tiempo conservaran la misma imagen.

1.10.3 Cargando recursos desde código

Cargar recursos desde codigo es fácil, hay dos maneras de hacerlo. La primera es usar load(), así:

```
func _ready():
        var res = load("res://robi.png") # el recurso es cargado cuando esta linea se␣
↪ejecuta
        get_node("sprite").set_texture(res)
```

La segunda forma es mas optima, pero solo funciona con un parámetro de cadena constante, porque carga el recurso en tiempo de compilación.

```
func _ready():
        var res = preload("res://robi.png") # el recurso se carga en tiempo de␣
↪compilación
        get_node("sprite").set_texture(res)
```

1.10.4 Cargar escenas

Las escenas son recursos, pero hay una excepción. Las escenas guardadas a disco son del tipo *PackedScene*, esto implica que la escena esta empacada dentro de un recurso.

Para obtener una instancia de la escena, el método *PackedScene.instance()* debe ser usado.

```
func _on_shoot():
        var bullet = preload("res://bullet.scn").instance()
        add_child(bullet)
```

Este método crea los nodos en jerarquía, los configura (ajusta todas las propiedades) y regresa el nodo raíz de la escena, el que puede ser agregado a cualquier nodo.

Este enfoque tiene varia ventajas. Como la función *PackedScene.instance()* es bastante rapida, agregar contenido extra a la escena puede ser hecho de forma eficiente. Nuevos enemigos, balas, efectos, etc pueden se agregados o quitados rápidamente, sin tener que cargarlos nuevamente de disco cada vez. Es importante recordar que, como siempre, las imágenes, meshes (mallas), etc son todas compartidas entre las instancias de escena.

1.10.5 Liberando recursos

Los recursos se extienden de *Reference*. Como tales, cuando un recurso ya no esta en uso, se liberara a si mismo de forma automática. Debido a que, en la mayoría de los casos, los Recursos están contenidos en Nodos, los scripts y otros recursos, cuando el nodo es quitado o liberado, todos los recursos hijos son liberados también.

1.10.6 Scripting

Como muchos objetos en Godot, no solo nodos, los recursos pueden ser scripts también. Sin embargo, no hay mucha ganancia, ya que los recursos son solo contenedores de datos.

1.11 Sistema de archivos (Filesystem)

1.11.1 Introducción

Los sistemas de archivos son otro tema importante en el desarrollo de un motor. El sistema de archivos gestiona como los assets son guardados, y como son accedidos. Un sistema de archivos bien diseñado también permite a múltiples desarrolladores editar los mismos archivos de código y assets mientras colaboran juntos.

Las versiones iniciales del motor Godot (y previas iteraciones antes de que se llamase Godot) usaban una base de datos. Los assets eran guardados en ella y se le asignaba un ID. Otros enfoques fuero intentados también, como base de datos local, archivos con metadatos, etc. Al final el enfoque mas simple gano y ahora Godot guarda todos los assets como archivos en el sistema de archivos.

1.11.2 Implementación

El sistema de archivos almacena los recursos en disco. Cualquier cosa, desde un script, hasta una escena o una imagen PNG es un recurso para el motor. Si un recurso contiene propiedades que referencian otros recursos en disco, los caminos (paths) a esos recursos con incluidos también. Si un recurso tiene sub-recursos que están incorporados, el recurso es guardado en un solo archivo junto a todos los demás sub-recursos. Por ejemplo, un recurso de fuente es a menudo incluido junto a las texturas de las fuentes.

En general el sistema de archivos de Godot evita usar archivos de metadatos. La razón para esto es es simple, los gestores de assets y sistemas de control de versión (VCSs) son simplemente mucho mejores que cualquier cosa que nosotros podamos implementar, entonces Godot hace su mejor esfuerzo para llevarse bien con SVN, Git, Mercurial, Preforce, etc.

Ejemplo del contenido de un sistema de archivos:

```
/engine.cfg
/enemy/enemy.scn
/enemy/enemy.gd
/enemy/enemysprite.png
/player/player.gd
```

1.11.3 engine.cfg

El archivo engine.cfg es la descripción del proyecto, y siempre se encuentra en la raíz del proyecto, de hecho su locación define donde esta la raiz. Este es el primer archivo que Godot busca cuando se abre un proyecto.

Este archivo contiene la configuración del proyecto en texto plano, usando el formato win.ini. Aun un engine.cfg vacío puede funcionar como una definición básica de un proyecto en blanco.

1.11.4 Delimitador de camino (path)

Godot solo soporta `/` como delimitador de ruta o camino. Esto es hecho por razones de portabilidad. Todos los sistemas operativos soportan esto, aun Windows, por lo que un camino como `c:\project\engine.cfg` debe ser tipiado como `c:/project/engine.cfg`.

1.11.5 Camino de recursos

Cuando se accede a recursos, usar el sistema de archivos del sistema operativo huésped puede ser complejo y no portable. Para resolver este problema, el camino especial `res://` fue creado.

El camino `res://` siempre apuntara a la raíz del proyecto (donde se encuentra engine.cfg, por lo que es un hecho que `res://engine.cfg` siempre es valido)

El sistema de archivos es de lectura-escritura solo cuando se corre el proyecto localmente desde el editor. Cuando se exporta o cuando se ejecuta en diferentes dispositivos (como teléfonos o consolas, o corre desde DVD), el sistema de archivos se vuelve de solo lectura, y la escritura no esta permitida.

1.11.6 Camino de usuario

Escribir a disco es de todas formas necesario para varias tareas como guardar el estado del juego o descargar paquetes de contenido. Para este fin existe un camino especial ‘‘user://‘‘que siempre se puede escribir.

1.11.7 Sistema de archivos de huésped

De forma alternativa se puede utilizar también caminos del sistema de archivos huésped, pero esto no es recomendado para un producto publicado ya que estos caminos no tienen garantía de funcionar en todas las plataformas. Sin embargo, usar caminos de sistema de archivos huésped puede ser muy útil cuando se escriben herramientas de desarrollo en Godot!

1.11.8 Inconvenientes

Hay algunos inconvenientes con este simple diseño de sistema de archivos. El primer tema es que mover assets (renombrarlos o moverlos de un camino a otro dentro del proyecto) va a romper las referencias a estos assets. Estas referencias deberán ser re-definidas para apuntar a la nueva locación del asset.

El segundo es que bajo Windows y OSX los nombres de archivos y caminos no toman en cuenta si son mayúsculas o minúsculas. Si un desarrollador trabajando en un sistema de archivos huésped guarda un asset como “myfile.PNG”, pero lo referencia como “myfile.png”, funcionara bien en su plataforma, pero no en otras, como Linux, Android, etc. Esto también puede aplicarse a archivos binarios exportados, que usan un paquete comprimido para guardar todos los archivos.

Es recomendado que tu equipo defina con claridad una nomenclatura para archivos que serán trabajados con Godot! Una forma simple y a prueba de errores es solo permitir nombres de archivos y caminos en minúsculas.

1.12 Scene Tree (Árbol de Escena)

1.12.1 Introducción

Aquí es donde las cosas se empiezan a poner abstractas, pero no entres en pánico, ya que no hay nada mas profundo que esto.

En los tutoriales previos, todo gira al rededor del concepto de Nodos, las escenas están hechas de ellos, y se vuelven activas cuando entran a *Scene Tree*.

Esto merece ir un poco mas profundo. De hecho, el sistema de escenas no es siquiera un componente de núcleo de Godot, ya que es posible ignorarlo y hacer un script (o código C++) que habla directamente con los servidores. Pero hacer un juego de esa forma seria un montón de trabajo y esta reservado para otros usos.

1.12.2 MainLoop (Ciclo Principal)

La forma en la que Godot trabaja internamente es la siguiente. Esta la clase *OS*, la cual es la única instancia que corre al principio. Luego, todos los controladores, servidores, lenguajes de scripting, sistema de escena, etc. son cargados.

Cuando la inicialización esta completa, la clase *OS* necesita el aporte de un *MainLoop* para correr. Hasta este punto, toda esta maquinaria es interna (puedes chequear el archivo main/main.cpp en el código fuente si alguna vez estas interesado en ver como funciona esto internamente).

El programa de usuario, o juego, comienza en el MainLoop. Esta clase tiene algunos métodos, para inicialización, idle (llamadas de retorno sincronizadas con los frames), fija (llamadas de retorno sincronizadas con física), y entrada.

Nuevamente, Esto es realmente de bajo nivel y cuando haces juegos en Godot, escribir tu propio MainLoop ni siquiera tiene sentido.

1.12.3 SceneTree (Árbol de Escena)

Una de las formas de explicar como Godot trabaja, es verlo como un motor de juegos de alto nivel sobre una middleware (lógica de intercambio de información entre aplicaciones) de bajo nivel.

El sistema de escenas es el motor de juegos, mientras que *OS* y los servidores son la API de bajo nivel.

En cualquier caso, el sistema de escenas provee su propio main loop al sistema operativo, *SceneTree*.

Esto es automáticamente instanciado y ajustado cuando se corre la escena, no hay necesidad de trabajo extra.

Es importante saber que esta clase existe porque tiene algunos usos importantes:

- Contiene la raiz *Viewport*, cuando una escena se abre por primera vez, es agregado como un hijo de ella para volverse parte del *Scene Tree* (mas sobre esto luego)
- Contiene información sobre los grupos, y tiene lo necesario para llamar a todos los nodos en un grupo, u obtener una lista de ellos.
- Contiene algo de funcionalidad sobre estados globales, como son ajustar el modo de pausa, o terminar el proceso.

Cuando un nodo es parte de un Árbol de Escena, el singleton *SceneTree* puede ser obtenido simplemente llamando *Node.get_tree()*.

1.12.4 Viewport Raíz

La raiz *Viewport* siempre esta en lo mas alto de la escena. Desde un nodo, puede ser obtenía de dos formas diferentes:

```
get_tree().get_root() # acceso vía scenemainloop
get_node("/root") # acceso vía camino absoluto
```

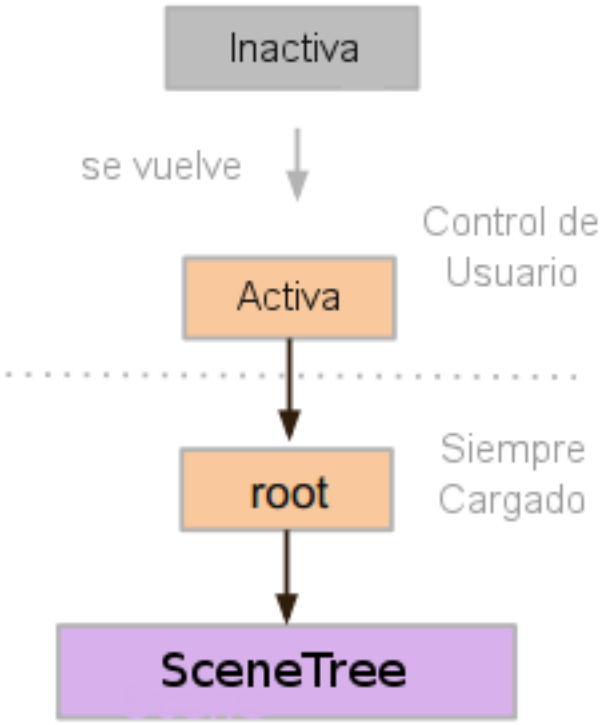
Este nodo contiene el viewport principal, cualquier cosa que sea hijo de un *Viewport* es dibujado dentro de el por defecto, por lo que tiene sentido que por encima de todos los nodos siempre haya un nodo de este tipo, de otra forma no se vería nada!

Mientras que otros viewports pueden ser creados en la escena (para efectos de pantalla dividida o similar), este es el único que nunca es creado por el usuario. Es creado automáticamente dentro de SceneTree.

1.12.5 Scene Tree (Arbol de Escena)

Cuando un nodo es conectado, directa o indirectamente, a la raíz del viewport, se vuelve parte del *Scene Tree*.

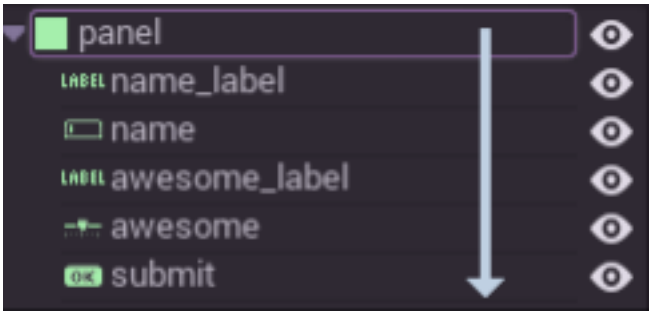
Esto significa que, como se explico en tutoriales previos, obtendrá los llamados de retorno _enter_tree() y _ready() (así como _exit_tree())



Cuando los nodos entran a *Scene Tree*, se vuelven activos. Obtienen acceso a todo lo que necesitan para procesar, obtener entradas, mostrar 2D y 3D, notificaciones, reproducir sonidos, grupos, etc. Cuando son removidos de la *Scene Tree*, lo pierden.

1.12.6 Orden del árbol

La mayoría de las operaciones con Nodos en Godot, como dibujar 2D, procesar u obtener notificaciones son hechas en el orden de árbol. Esto significa que los padres y hermanos con menor orden van a ser notificados antes que el nodo actual.



1.12.7 "Volverse activo" por entrar la *Scene Tree*

1. Una escena es cargada desde disco o creada por scripting.
2. El nodo raíz de dicha escena (solo una raíz, recuerdan?) es agregado como un hijo del Viewport "root" (desde SceneTree), o hacia cualquier hijo o nieto de el.

3. Todo nodo de la escena recientemente agregada, recibirá la notificacion “enter_tree” (llamada de retorno _enter_tree() en GDScript) en orden de arriba hacia abajo.
4. Una notificación extra, “ready” (llamada de retorno _ready() en GDScript) se provee por conveniencia, cuando un nodo y todos sus hijos están dentro de la escena activa.
5. Cuando una escena (o parte de ella) es removida, reciben la notificación “exit scene” (llamada de retorno _exit_tree()) en GDScript) en orden de abajo hacia arriba.

1.12.8 Cambiando la escena actual

Luego que una escena es cargada, suele desearse cambiar esta escena por otra. La forma simple de hacer esto es usar la funcion *SceneTree.change_scene()*:

```
func _mi_nivel_fue_completado():
    get_tree().change_scene("res://levels/level2.scn")
```

Esta es una forma fácil y rápida de cambiar de escenas, pero tiene la desventaja de que el juego se detendrá hasta que la nueva escena esta cargada y corriendo. En algún punto de tu juego, puede ser deseable crear una pantalla de carga con barra de progresa adecuada, con indicadores animados o carga por thread (en segundo plano). Esto debe ser hecho manualmente usando autoloads (ve el próximo capitulo!) y *Carga en segundo plano*.

1.13 Singletons (AutoCarga)

1.13.1 Introducción

Los Singletons de escena son muy útiles, ya que representan un caso de uso muy común, pero no es claro al principio de donde viene su valor.

El sistema de escenas es muy útil, aunque propiamente tiene algunas desventajas:

- No hay lugar común donde almacenar información (como núcleo, ítems obtenidos, etc) entre dos escenas.
- Es posible hacer una escena que carga otras escenas como hijos y las libera, mientras mantiene la información, pero si esto es hecho, no es posible correr una escena sola por si misma y esperar que funcione.
- También es posible almacenar información persistente a disco en ‘user://‘ y hacer que las escenas siempre lo carguen, pero guardar y cargar eso mientras cambiamos de escenas es incomodo.

Por lo tanto, luego de usar Godot por un tiempo, se vuelve claro que es necesario tener partes de una escena que:

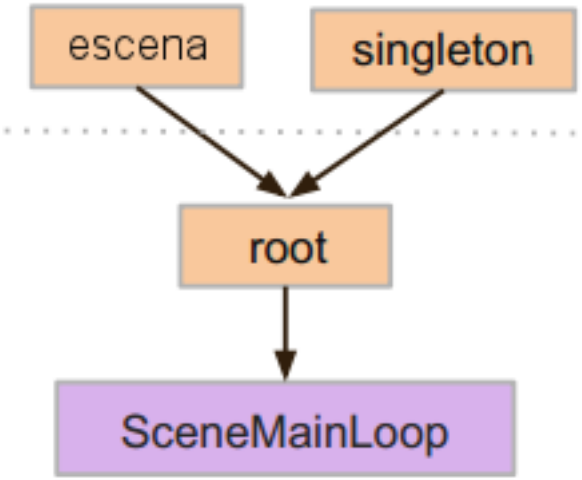
- Están siempre cargadas, no importa que escena es abierta desde el editor.
- Puede mantener variables globales, como información de jugador, ítems, dinero, etc.
- Puede manejar cambios de escenas y transiciones.
- Solo ten algo que actúe como singleton, ya que GDScript no soporta variables globales por diseño.

Por eso, la opción para auto-cargar nodos y scripts existe.

1.13.2 AutoLoad (AutoCarga)

AutoLoad puede ser una escena, o un script que hereda desde Nodo (un Nodo sera creado y el script ajustado para el). Son agregados a el proyecto en Escena > Configuración de Proyecto > pestaña AutoLoad.

Cada autoload necesita un nombre, este nombre sera el nombre del nodo, y el nodo siempre sera agregado al viewport root (raíz) antes de que alguna escena sea cargada.



Esto significa, que para un singleton llamado "jugadorvariables", cualquier nodo puede accederlo al requerirlo:

```
var jugador_vars = get_node("/root/jugadorvariables")
```

1.13.3 Conmutador(Switcher) personalizado de escena

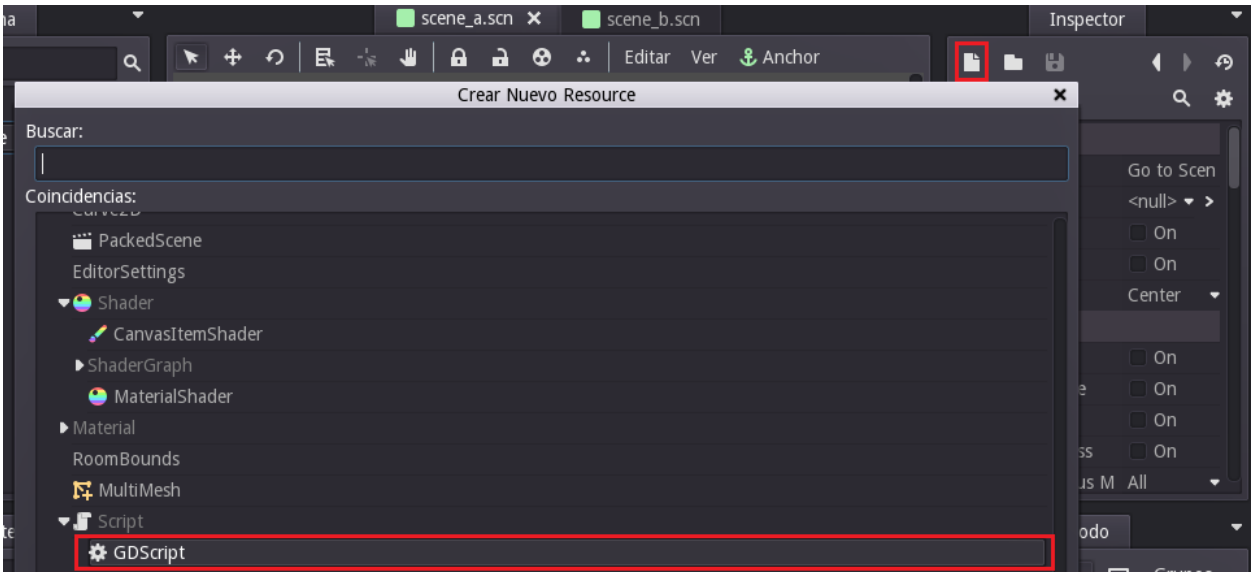
Este corto tutorial explicara como hacer para lograr un conmutador usando autoload. Para conmutar una escena de forma simple, el método *SceneTree.change_scene()* basta (descrito en *Scene Tree (Árbol de Escena)*), por lo que este método es para comportamientos mas complejos de conmutación de escenas.

Primero descarga la plantilla desde aquí: `autoload.zip`, y ábrela.

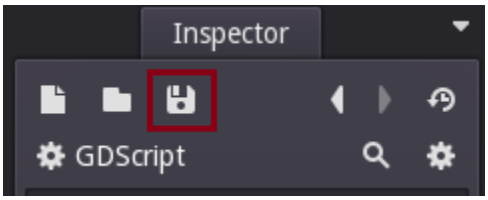
Dos escenas están presentes, scene_a.scn y scene_b.scn en un proyecto que salvo por estas escenas esta vació. Cada una es idéntica y contiene un botón conectado a la llamada de retorno para ir a la escena opuesta. Cuando el proyecto corre, comienza en scene_a.scn. Sin embargo, esto no hace nada y tocar el boton no funciona.

1.13.4 global.gd

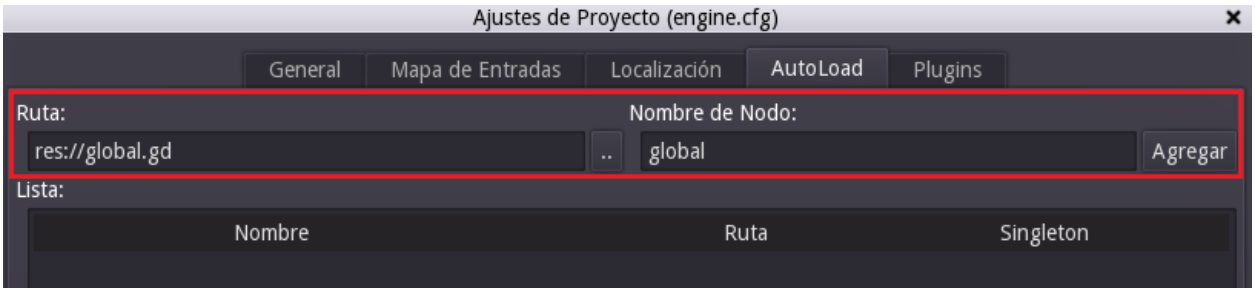
Primero que nada, crea un script global.gd. La forma mas fácil de crear un recurso desde cero es desde la pestaña del Inspector:



Guarda el script con el nombre global.gd:



El script debería estar abierto en el editor de scripts. El siguiente paso sera agregarlo a autoload, para esto, ve a: Escena, Configuración de Proyecto, Autoload y agrega un nuevo autoload con el nombre "global" que apunta a ester archivo:



Ahora, cuando la escena corre, el script siempre sera cargado.

Así que, yendo un poco atrás, en la función _ready(), la escena actual sera traída. Tanto la escena actual como global.gd son hijos de root, pero los nodos autocargados siempre están primeros. Esto implica que el ultimo hijo de de root es siempre la escena cargada.

También, asegúrate que global.gd se extienda desde Nodo, de otra forma no sera cargada.

```
extends Node

var escena_actual = null

func _ready():
        var raiz = get_tree().get_root()
        escena_actual = raiz.get_child( raiz.get_child_count() - 1 )
```

Como siguiente paso, es necesaria la función para cambiar de escena. Esta función va a borrar la escena actual y reemplazarla por la que se pidio.

```
func ir_escena(camino):

    # Esta función usualmente sera llamada de una señal de
    # llamada de retorno, o alguna otra función de la escena
    # que esta corriendo borrar la escena actual en este punto
    # puede ser una mala idea, porque puede estar dentro de una
    # llamada de retorno o función de ella. El peor caso va a
    # ser que se cuelgue o comportamiento no esperado.

    # La forma de evitar esto es difiriendo la carga para mas
    # tarde, cuando es seguro que ningún código de la escena
    # actual esta corriendo:

    call_deferred("_ir_escena_diferida",camino)


func _ir_escena_diferida(camino):

    # Inmediatamente libera la escena actual,
    # no hay riesgo aquí.
    escena_actual.free()

    # Carga la nueva escena
    var s = ResourceLoader.load(camino)

    # Instancia la nueva escena
    escena_actual = s.instance()

    # Agrégalo a la escena activa, como hijo de root
    get_tree().get_root().add_child(escena_actual)

    # Opcional, para hacerlo compatible con la API SceneTree.change_scene()
    get_tree().set_current_scene( escena_actual )
```

Como mencionamos en los comentarios de arriba, realmente queremos evitar la situación de tener la escena actual siendo borrada mientras esta siendo usada (el código de sus funciones aun corriendo), por lo que usando *Object.call_deferred()* es recomendado en este punto. El resultado es que la ejecución de los comandos en la segunda función van a suceder en un momento inmediatamente posterior inmediato cuando no hay código de la escena actual corriendo.

Finalmente, todo lo que queda es llenar las funciones vacías en scene_a.gd y scene_b.gd:

```
#agrega a scene_a.gd

func _on_goto_scene_pressed():
        get_node("/root/global").ir_escena("res://scene_b.scn")
```

y

```
#agrega a scene_b.gd

func _on_goto_scene_pressed():
        get_node("/root/global").ir_escena("res://scene_a.scn")
```

Ahora, al correr el proyecto es posible conmutar entre ambas escenas al presionar el botón!

(Para cargar escenas con una barra de progreso, chequea el próximo tutorial, *Carga en segundo plano*)

CAPÍTULO 2

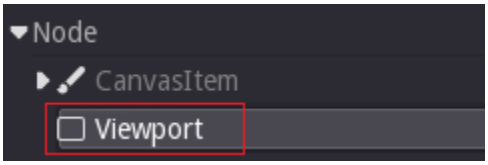
Motor

2.1 Escena, entrada y viewports

2.1.1 Viewports (Visores)

Introducción

Godot tiene una característica pequeña pero útil llamada viewports. Los Viewports son, como su nombre implica, rectángulos donde el mundo es dibujado. Tienen 3 usos principales, pero pueden ser flexiblemente adaptados para hacer mucho mas. Todo esto se hace con el nodo *Viewport*.



Los principales usos de los que hablábamos son:

- **Scene Root**: Las raiz de la escena activa siempre es un viewport. Esto es lo que muestra las escenas creadas por el usuario. (Deberías saber esto de los tutoriales previos!)
- **Sub-Viewports**: Estos pueden ser creados cuando un Viewport es hijo de un *Control*.
- **Render Targets**:: Viewports pueden ser ajustadas en el modo "RenderTarget". Esto implica que el viewport no es visible directamente, pero sus contenidos pueden ser accedidos con una *Texture*.

Entrada

Los viewports también son responsables de entregar eventos de entrada correctamente ajustados y escalados a todos sus nodos hijos. Ambos el root del viewport y sub-viewports hacen esto automáticamente, pero los rendertargets no. Por este motivo, es usuario debe hacerlo manualmente con la funcion *Viewport.input()* si es necesario.

Listener (Oyente)

Godot soporta sonido 3D (tanto en nodos 2D como 3D), mas sobre esto puede ser encontrado en otro tutorial (un día..). Para que este tipo de sonido sea audible, el viewport necesita ser habilitado como un listener (para 2D o 3D). Si estas usando un viewport para mostrar tu mundo, no olvides de activar esto!

Cámaras (2D y 3D)

Cuando se usa una *Camera Camera2D* 2D o 3D, las cámaras siempre mostraran en el viewport padre mas cercano (yendo hacia root). Por ejemplo, en la siguiente jerarquía:

- Viewport
    - Camera

Las cámaras se mostraran en el viewport padre, pero en el siguiente orden:

- Camera
    - Viewport

No lo hará (o puede mostrarse en viewport root si es una subescena).

Solo puede haber una cámara activa por viewport, por lo que si hay mas de una, asegúrate que la correcta tiene la propiedad "current" ajustada, o hazlo llamando:

```
camera.make_current()
```

Escala y estiramiento

Los viewports tienen la propiedad "rect". X e Y no son muy usados (solo el viewport root realmente los utiliza), mientras WIDTH y HEIGHT representan el tamaño del viewport en pixels. Para sub-viewports, estos valores son sobrescritos por los del control padre, pero para render targets ajusta su resolución.)

También es posible escalar el contenido 2D y hacerle creer que la resolución del viewport es otra que la especificada en el rect, al llamar:

```
viewport.set_size_override(w,h) #tamaño personalizado para 2D
viewport.set_size_override_stretch(true/false) #habilita el estiramiento para tamaño␣
↪personalizado
```

El viewport root usa esto para las opciones de estiramiento en la configuración del proyecto.

Worlds (Mundos)

Para 3D, un viewport contendrá *World*. Este es básicamente el universo que une la física y la renderización. Los nodos Spatial-base serán registrados usando el World del viewport mas cercano. Por defecto, los viewports recientemente creados no contienen World pero usan el mismo como viewport padre (pero el viewport root no tiene uno, el cual es donde los objetos son renderizados por defecto). World puede ser ajustado en un viewport usando la propiedad "world", y eso separara todos los nodos hijos de ese viewport de interactuar con el viewport world padre. Esto es especialmente útil en escenarios donde, por ejemplo, puedes querer mostrar un personaje separado en 3D impuesto sobre sobre el juego (como en Starcraft).

Como un ayudante para situaciones donde puedes querer crear viewports que muestran objetos únicos y no quieres crear un mundo, viewport tiene la opción de usar su propio World. Esto es muy útil cuando tu quieres instanciar personajes 3D u objetos en el mundo 2D.

Para 2D, cada viewport siempre contiene su propio *World2D*. Esto es suficiente en la mayoría de los casos, pero si se desea compartirlo, es posible hacerlo al llamar la API viewport manualmente.

Captura

Es posible requerir una captura de los contenidos del viewport. Para el viewport root esto es efectivamente una captura de pantalla. Esto es hecho con la siguiente API:

```
# encola una captura de pantalla, no sucederá inmediatamente
viewport.queue_screen_capture()
```

Luego de un frame o dos (check_process()), la captura estará lista, obtenla usando:

```
var capture = viewport.get_screen_capture()
```

Si la imagen retornada esta vacía, la captura aun no sucedió, espera un poco mas, esta API es asincrónica.

Sub-viewport

Si el viewport es hijo de un control, se volverá activa y mostrara lo que tenga dentro. El diseño es algo como esto:

- Control
    - Viewport

El viewport cubrirá el área de su control padre completamente.

Render target (objetivo de renderizacion)

Para ajustar como un render target, solo cambia la propiedad "render target" de el viewport a habilitado. Ten en cuenta que lo que sea que esta dentro no será visible en el editor de escena. Para mostrar el contenido, la textura de render target debe ser usada. Esto puede ser pedido con código usando (por ejemplo):

```
var rtt = viewport.get_render_target_texture()
sprite.set_texture(rtt)
```

Por defecto, la re-renderizacion del render target sucede cuando la textura del render target ha sido dibujada en el frame. Si es visible, será renderizada, de otra contrario no lo será. Este comportamiento puede ser cambiado a renderizado manual (una vez), o siempre renderizar, no importando si es visible o no.

Algunas clases son creadas para hacer esto mas sencillo en la mayoría de los casos dentro del editor:

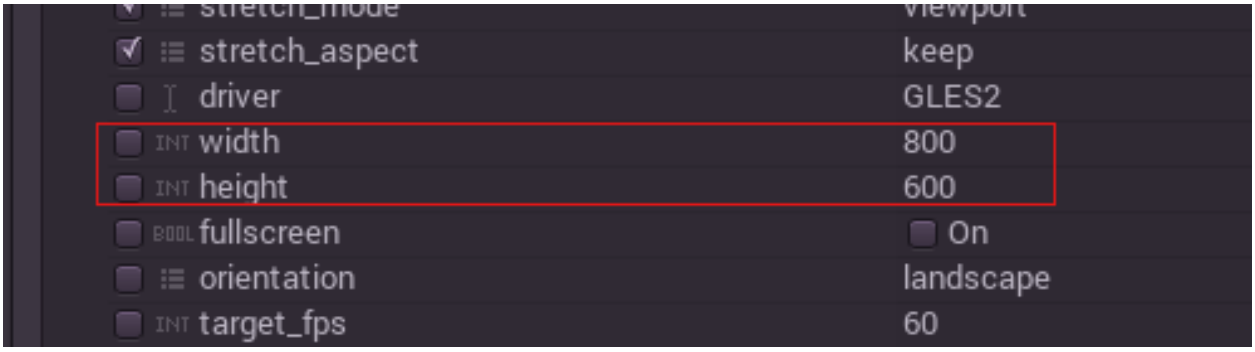
- *ViewportSprite* (para 2D).

Asegúrate de chequear los demos de viewports! La carpeta viewport en el archivo disponible para descarga en el sitio principal de Godot, o https://github.com/godotengine/godot/tree/master/demos/viewport

2.1.2 Resoluciones múltiples

Resolución base

Una resolución de pantalla base para el proyecto puede ser especificada en configuración de proyecto.



Sin embargo, lo que hace no es completamente obvio. Cuando corre en PC, el motor intentara ajustara esta resolución (o usar algo mas chico si falla). En equipos móviles, consolas u otros dispositivos con una resolución fija o renderizacion de pantalla completa, esta resolución será ignorada y la resolución nativa se usara en su lugar. Para compensar esto, Godot ofrece muchas formas de controlar como la pantalla va a cambiar de tamaño y estirarse a diferentes tamaños de pantalla.

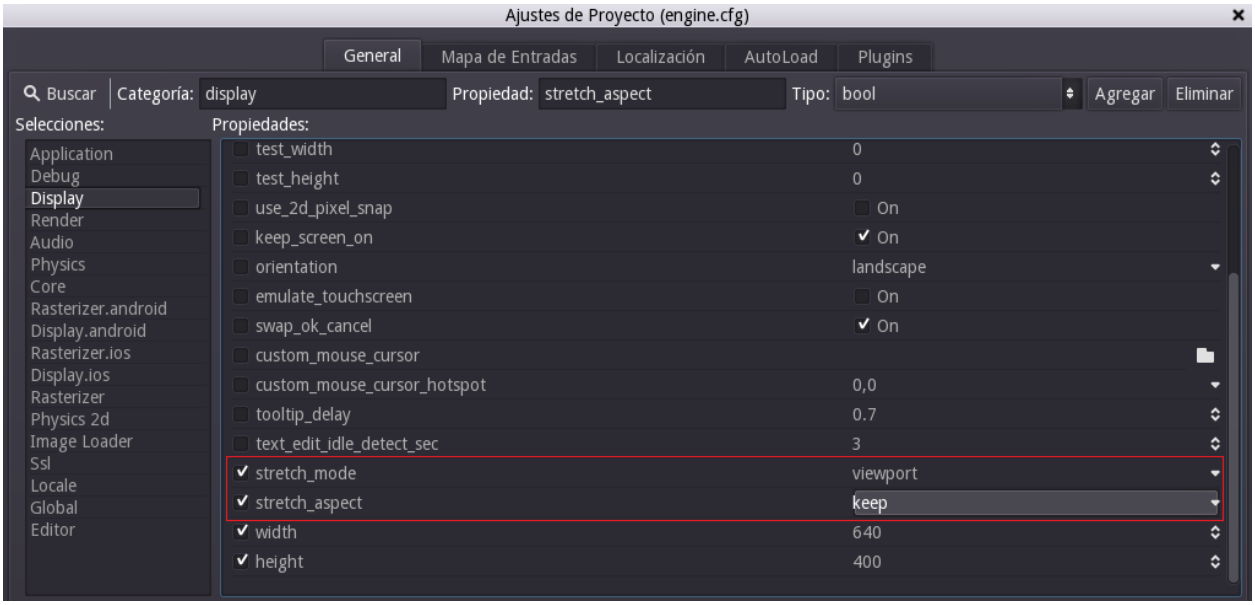
Resizing (Cambiar de tamaño)

Hay varios tipos de dispositivos, con varios tipos de pantalla, los cuales a su vez tienen diferente densidad de pixels y resolución. Manejarlos todos puede ser un montón de trabajo, asique Godot trata de hacer la vida del desarrollador un poco mas fácil. El nodo *Viewport* tiene varias funciones para manejar el cambio de tamaño, y el nodo root de la escena es siempre un viewport (las escenas cargadas son instanciadas como hijos de el, y siempre se pueden acceder llamando `get_tree().get_root()` o `get_node("/root")`).

En cualquier caso, mientras cambiar los parámetros del viewport root es probablemente la forma mas flexible de atender este problema, puede ser un montón de trabajo, código y adivinanza, por lo que Godot provee un rango de parámetros simple en la configuración de proyecto para manejar múltiples resoluciones.

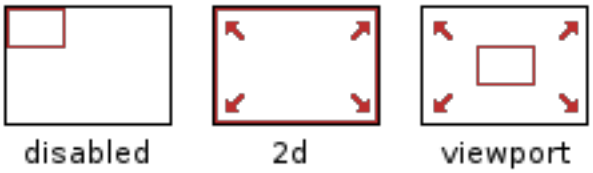
Configuraciones de estiramiento

Las configuraciones de estiramiento están localizadas en la configuración de proyecto, es solo un montón de variables de configuración que proveen varias opciones:



Modo de estiramiento

- **Disabled**: Lo primero es el modo de estiramiento. Por defecto esta deshabilitado, lo que significa que no hay estiramiento (cuanto mas grande la pantalla o ventana, mas grande la resolución, siempre igualando pixeles 1:1).
- **2D**: En este modo, la resolución especificada en display/width y display/height en la configuración del proyecto será estirada para cubrir la pantalla entera. Esto implica que 3D no sera afectado (solo va a renderizar a una resolución mas alta) y 2D también será renderizado a una resolución mayor, solo que agrandada.
- **Viewport**: El escalado de viewport es diferente, el *Viewport* root se ajusta como un render target, y aun renderiza precisamente a la resolución especificada en la sección `display/` de la configuración de proyecto. Finalmente, este viewport es copiado y escalado para entrar a la pantalla. Este modo es útil cuando trabajas con juegos que requieren precisión de pixeles, o solo con el motivo de renderizar a una resolución mas baja para mejorar la performance.



Aspecto de estiramiento

- **Ignore**: Ignorar la relación de aspecto cuando se estire la pantalla. Esto significa que la resolución original será estirada para entrar en la nueva, aun cuando sea mas ancha o mas angosta.
- **Keep**: Mantener la relación de aspecto cuando se estire la pantalla. Esto implica que la resolución original será mantenida cuando se ajuste a una nueva, y barras negras serán agregadas a los costados o borde superior e inferior de la pantalla.

- **Keep Width**: Mantener la relación de aspecto cuando se estira la pantalla, pero si la resolución de pantalla es mas alta que la resolución especificada, será estirada verticalmente (y una mayor resolución vertical será reportada en el viewport, proporcionalmente) Esta es usualmente la mejor opción para crear GUIs o HUDs que se escalan, asi algunos controles pueden ser anclados hacia abajo (*Tamaño y anclas*).
- **Keep Height**: Mantener la relación de aspecto cuando se estira la pantalla, pero si la pantalla resultante es mas ancha que la resolución especificada, será estirada horizontalmente (y mas resoluciones horizontales serán reportadas en el viewport, proporcionalmente). Esta es usualmente la mejor opción para juegos 2D que tienen scroll(desplazamiento) horizontal (como juegos de plataforma)

2.1.3 InputEvent (Evento de Entrada)

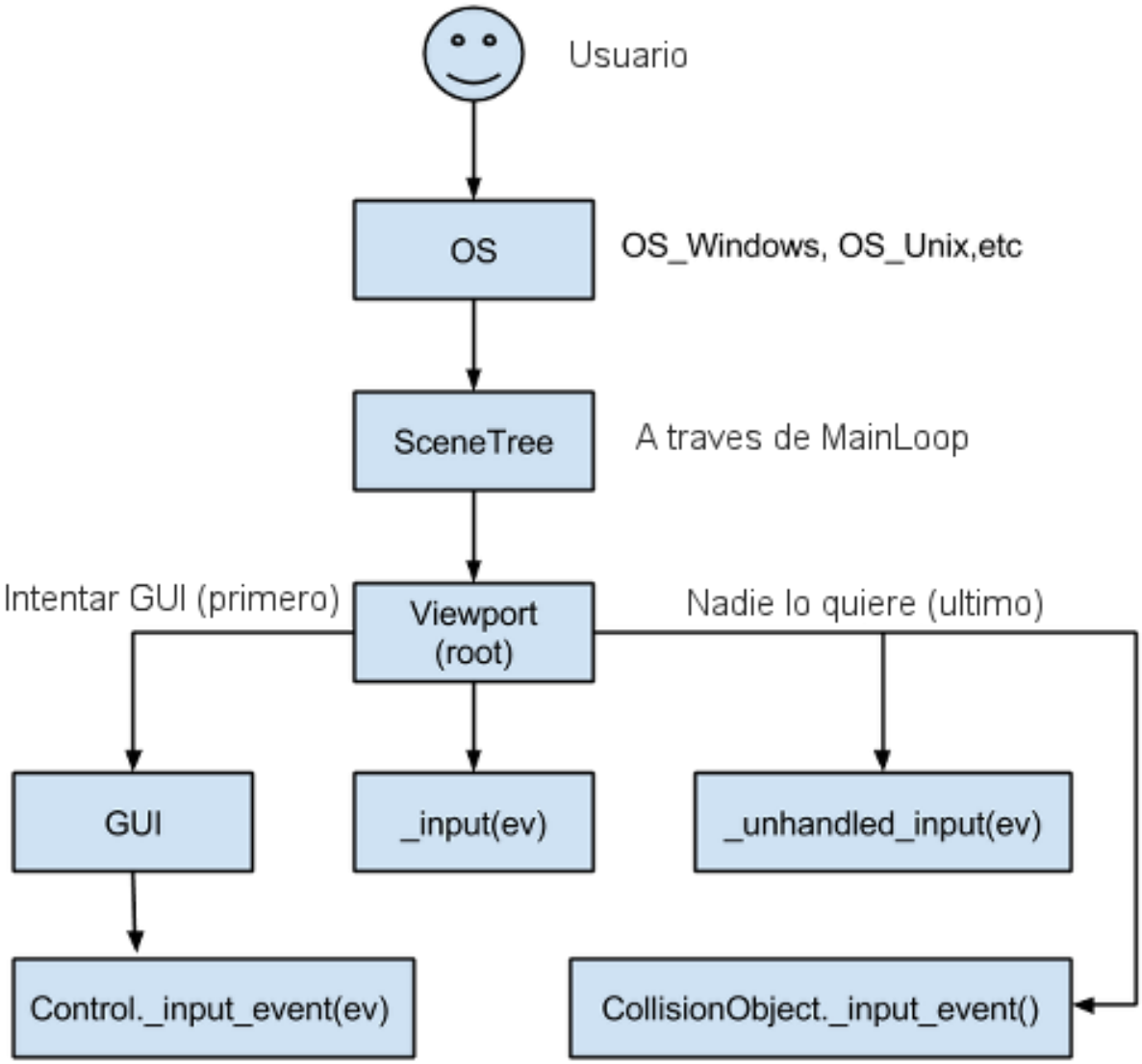
Que es?

Gestionar entradas es usualmente complejo, no importa el SO o la plataforma. Para facilitarlo un poco, un objeto especial incorporado se provee, *InputEvent*. Este tipo de datos puede ser configurado para contener varios tipos de eventos de entrada. Input Events viaja a través del motor y puede ser recibido en múltiples lugares, dependiendo del propósito.

Como funciona?

Cada evento de entrada se origina desde el usuario/jugador (aunque es posible generar un InputEvent y dárselo de vuelta al motor, lo cual es util para gestos). El objeto OS para cada plataforma va a leer eventos desde el dispositivo, luego pasándoselo al MainLoop. Como *SceneTree* es la implementación por defecto de MainLoop, los eventos se envían allí. Godot provee una función para obtener el objeto SceneTree actual: **get_tree()**

Pero SceneTree no sabe que hacer con este evento, por lo que se lo va a dar a los viewports, empezando desde "root" *Viewport* (el primer nodo del arbol de escena). Viewport hace una cantidad considerable de cosas con la entrada recibida, en orden:



1. Primero, intentara darle la entrada a la GUI, y ver si algún control lo recibe. De ser asi, *Control* será llamado por la función virtual *Control._input_event()* y la señal "input_event" sera emitida (esta función es reimplementable por script al heredarla). Si el control quiere "consumir" el evento, va a llamar *Control.accept_event()* y el evento no se esparcirá mas.
2. Si la GUI no quiere el evento, la función standard _input sera llamada en cualquier nodo con procesamiento de entrada habilitado (habilítalo con *Node.set_process_input()* y sobrescribe *Node._input()*). Si alguna función consume el evento, puede llamar *SceneTree.set_input_as_handled()*, y el evento no se esparcirá mas.
3. Si aun nadie consumió el evento, la llamada de retorno unhandled será invocada (habilítala con *Node.set_process_unhandled_input()* y sobrescribe *Node._unhandled_input()*). Si alguna funcion consume el evento, puede llamar a *SceneTree.set_input_as_handled()*,, y el evento no se esparcirá mas.
4. Si nadie ha querido el evento hasta ahora, y una *Camera* es asignada al viewport, un rayo al mundo de física (en la dirección de rayo del clic) será emitido. Si este rayo toca un objeto, llamara a la funcion *CollisionObject._input_event()* en el objeto físico relevante (los cuerpos reciben esta llamada de retorno por defecto, pero las áreas no. Esto puede ser configurado a través de las propiedades *Area*).

5. Finalmente, si el evento no fue utilizado, será pasado al siguiente Viewport en el árbol, de otra forma será ignorado.

Anatomía de un InputEvent

InputEvent es solo un tipo base incorporado, no representa nada y solo contiene información básica, como el ID de evento (el cual es incrementado para cada evento), índice de dispositivo, etc.

InputEvent tiene un miembro "type". Al asignarlo, se puede volver diferentes tipos de entrada. Todo tipo de InputEvent tiene propiedades diferentes, de acuerdo a su rol.

Ejemplo de un tipo de evento cambiante.

```
# crear evento
var ev = InputEvent()
# ajustar tipo index (índice)
ev.type = InputEvent.MOUSE_BUTTON
# button_index solo esta disponible para el tipo de arriba
ev.button_index = BUTTON_LEFT
```

Hay varios tipo de InputEvent, descritos en la tabla de abajo:

Evento	Type Index	Descripción
InputEvent	NONE	Evento de entrada vacío.
InputEventKey	KEY	Contiene un valor scancode y unicode, además de modificadores.
InputEventMouseButton	MOUSE_BUTTON	Contiene información de clics, como botón, modificadores, etc.
InputEventMouseMotion	MOUSE_MOTION	Contiene información de movimiento, como pos. relativas y absolutas, velocidad
InputEventJoystickMotion	JOYSTICK_MOTION	Contiene información de ejes análogos de Joystick/Joypad
InputEventJoystickButton	JOYSTICK_BUTTON	Contiene información de botones de Joystick/Joypad
InputEventScreenTouch	SCREEN_TOUCH	Contiene información multi-touch de presionar/soltar. (solo disponible en dispositivos móviles)
InputEventScreenDrag	SCREEN_DRAG	Contiene información multi-touch de arrastre (solo en disp. móviles)
InputEventAction	SCREEN_ACTION	Contiene una acción genérica. estos eventos suelen generarse por el program- ador como feedback. (mas de esto abajo)

Acciones

Un InputEvent puede o no representar una acción predefinida. Las acciones son útiles porque abstraen el dispositivo de entrada cuando programamos la lógica de juego. Esto permite:

- Que el mismo código trabaje en diferentes dispositivos con diferentes entradas (por ej. teclado en PC, Joypad en consola).
- Entrada a ser reconfigurada en tiempo de ejecución.

Las acciones pueden ser creadas desde la Configuración de Proyecto en la pestaña Actions. Lee *Configuración de acciones de entrada* para una explicación de como funciona el editor de acciones.

Cualquier evento tiene los metodos *InputEvent.is_action()*, *InputEvent.is_pressed()* y *InputEvent*.

De forma alternativa, puede desearse suplir al juego de vuelta con una acción desde el código (un buen ejemplo es detectar gestos). SceneTree (derivado de MainLoop) tiene un método para esto: *MainLoop.input_event()*. Se usaría normalmente de esta forma:

```
var ev = InputEvent()
ev.type = InputEvent.ACTION
# ajustar como move_left, presionado
ev.set_as_action("move_left", true)
# feedback
get_tree().input_event(ev)
```

InputMap (Mapa de controles)

Personalizar y re mapear entrada desde código es a menudo deseado. Si tu worflow depende de acciones, el singleton *InputMap* es ideal para reasignar o crear diferentes acciones en tiempo de ejecución. Este singleton no es guardado (debe ser modificado manualmente) y su estado es corrido desde los ajustes de proyecto (engine.cfg). Por lo que cualquier sistema dinámico de este tipo necesita guardar su configuración de la forma que el programador lo crea conveniente.

2.1.4 Coordenadas de mouse y entrada

Acerca

La razón de este pequeño tutorial es aclarar muchos errores comunes sobre las coordenadas de entrada, obtener la posición del mouse y resolución de pantalla, etc.

Coordenadas de pantalla de hardware

Usar coordenadas de hardware tiene sentido en el caso de escribir UIs complejas destinada a correr en PC, como editores, MMOs, herramientas, etc. De todas formas, no tiene mucho sentido fuera de ese alcance.

Coordenadas de pantalla de viewport

Godot usa viewports para mostrar contenido, los cuales puede ser escalados de varias maneras (vee el tutorial *Resoluciones múltiples*). Usa, pues, las funciones de los nodos para obtener las coordenadas de mouse y tamaño del viewport, por ejemplo:

```
func _input(ev):
    # Mouse en coordenadas viewport

    if (ev.type==InputEvent.MOUSE_BUTTON):
        print("El mouse fue Click/Unclick en: ",ev.pos)
    elif (ev.type==InputEvent.MOUSE_MOTION):
        print("Movimiento de mouse en: ",ev.pos)

    # Imprime el tamaño del viewport

    print("La resolución del viewport es: ",get_viewport_rect().size)

func _ready():
     set_process_input(true)
```

Alternativamente es posible pedir al viewport la posición del mouse:

```
get_viewport().get_mouse_pos()
```

2.2 Filesystem (Sistema de archivos)

2.2.1 Organización de proyectos

Introducción

Este tutorial esta dirigido a proponer un workflow simple sobre como organizar proyectos. Debido a que Godot permite que el programador use el sistema de archivos como el o ella quieren, encontrar una forma de organizar los proyectos cuando empiezas a usar el motor puede ser algo difícil. Por esto, un workflow simple sera descrito, el cual puede ser usado o no, pero deberia servir como un punto inicial.

Ademas, usar control de versiones puede ser un rato, por lo que este documento también lo incluirá.

Organización

Otros motores de juegos a menudo trabajan teniendo una base de datos, donde puedes navegar imágenes, modelos, sonidos, etc. Godot es mas basado en escenas por naturaleza así que la mayor parte del tiempo los assets están empacados dentro de las escenas o simplemente existen como archivos pero no referenciados desde escenas.

Importación & directorio del juego

Es muy a menudo necesario usar la importación de assets en Godot. Como los assets de origen para importar también son reconocidos como recursos por el motor, esto puede volverse un problema si ambos están dentro de la carpeta de proyecto, porque al momento de exportar el exportador va a reconocerlos y exportar ambos.

Para resolver esto, es una buena practica tener tu carpeta de juego dentro de otra carpeta (la verdadera carpeta del proyecto). Esto permite tener los assets del juego separados de los assets de origen, y también permite el uso de control de versiones (como svn o git) para ambos. Aquí hay un ejemplo:

```
myproject/art/models/house.max
myproject/art/models/sometexture.png
myproject/sound/door_open.wav
myproject/sound/door_close.wav
myproject/translations/sheet.csv
```

Luego también, el juego en si, en este caso, dentro de la carpeta game/:

```
myproject/game/engine.cfg
myproject/game/scenes/house/house.scn
myproject/game/scenes/house/sometexture.tex
myproject/game/sound/door_open.smp
myproject/game/sound/door_close.smp
myproject/game/translations/sheet.en.xl
myproject/game/translations/sheet.es.xl
```

Siguiendo este diseño, muchas cosas pueden ser hechas:

- El proyecto entero sigue estando dentro de una carpeta (myproject/).
- Exportar el proyecto no exportara los archivos .wav y .png que fueron importados.

- myproject/ puede ser puesta directamente dentro de un VCS (como svn o git) para control de versión, tanto los assets de origen como del juego serán seguidos.
- Si un equipo esta trabajando en el proyecto, los assets pueden ser re-importados por otros miembros del proyecto, porque Godot sigue a los assets de origen usando paths relativos.

Organización de escena

Dentro de la carpeta del juego, una pregunta que a menudo aparece es como organizar las escenas en el sistema de archivos. Muchos desarrolladores intentan la organización de assets por tipo y terminan con un desorden luego de un tiempo, por lo que la mejor respuesta probablemente es organizarlos basados en como funciona el juego y no en el tipo de asset. Aquí algunos ejemplos.

Si organizas tu proyecto basado en tipo de asset, luciría como esto:

```
game/engine.cfg
game/scenes/scene1.scn
game/scenes/scene2.scn
game/textures/texturea.png
game/textures/another.tex
game/sounds/sound1.smp
game/sounds/sound2.wav
game/music/music1.ogg
```

Lo cual en general es una mala idea. Cuando un proyecto comienza a crecer mas allá de cierto punto, esto se vuelve inmanejable. Es realmente difícil decir que pertenece donde.

En general es una mejor idea usar organización basada en contexto del juego, algo como esto:

```
game/engine.cfg
game/scenes/house/house.scn
game/scenes/house/texture.tex
game/scenes/valley/canyon.scn
game/scenes/valley/rock.scn
game/scenes/valley/rock.tex
game/scenes/common/tree.scn
game/scenes/common/tree.tex
game/player/player.scn
game/player/player.gd
game/npc/theking.scn
game/npc/theking.gd
game/gui/main_screen/main_sceen.scn
game/gui/options/options.scn
```

Este modelo o modelos similares permiten que los proyectos crezcan hasta tamaños realmente grandes y aun ser completamente administrable. Nota que todo esta basado en partes del jugo que pueden ser nombradas o descritas, como la pantalla de ajustes (settings screen) o el valle (valley). Debido a que todo en Godot es hecho con escenas, y todo lo que puede ser nombrado o descrito puede ser una escena, este workflow es muy suave y llevadero.

Archivos de Cache

Godot usa un archivo oculto llamado ".fscache" en la raíz del proyecto. En el, se cachean los archivos de proyecto y es usado para rápidamente saber cuando uno es modificado. Asegúrate de **no enviar este archivo** a git o svn, ya que contiene información local y puede confundir otra instancia del editor en otra computadora.

2.2.2 Paths(Caminos) de Datos

Separadores de caminos

Con el motivo de soportar tantas plataformas como es posible, Godot solo acepta separadores de camino de estilo unix (`/`). Estos funcionan en cualquier lado, incluyendo Windows.

Un camino como `C:\Projects` se transforma en `C:/Projects`.

Path de recursos

Como se menciono antes, Godot considera que un proyecto existe en cualquier carpeta que contenga un archivo de texto "engine.cfg", aun si el mismo esta vacío.

Acceder a archivos de proyectos puede ser hecho abriendo cualquier camino con `res://` como base. Por ejemplo, una textura localizada en la raíz de la carpeta del proyecto puede ser abierta con el siguiente camino: `res://algunatextura.png`.

Camino de datos de usuario (datos persistentes)

Mientras el proyecto esta corriendo, es un escenario muy común que el camino de recursos sea de solo lectura, debido a que esta dentro de un paquete, ejecutable autónomo, o lugar de instalación de sistema.

Guardar archivos persistentes en esos escenarios debería ser hecho usando el prefijo `user://`, por ejemplo: `user://gamesave.txt`.

En algunos dispositivos (por ejemplo, móvil y consola) este camino es único para la applicacion. Bajo sistemas operativos de escritorio, el motor usa el nombre tipico ~/.Name (chequea el nombre de proyecto bajo los ajustes) en OSX y Linux, y APPDATA/Name para Windows.

2.2.3 Salvando juegos

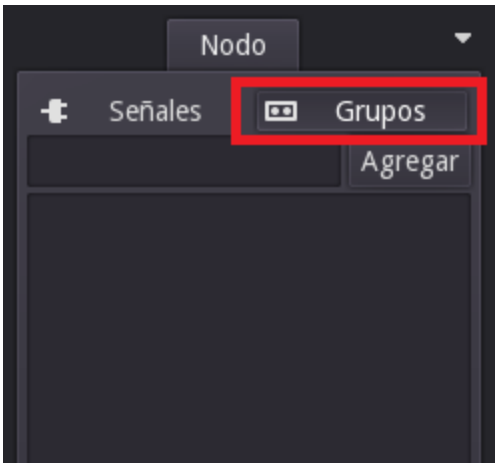
Introduccion

Salvar juegos puede ser complicado. Puede desearse guardar mas informacion que la que el nivel actual o numero de estrellas ganadas en un nivel. Juegos salvados mas avanzados pueden necesitar guardar informacion adicional sobre un numero arbitrario de objetos. Esto permitira que la funcion de salvar sea escalada en la medida que el juego crece en complejidad.

Identificar objetos persistentes

Primeor deberiamos identificar que objetos queremos mantener entre sesiones y que informacion queremos mantener de esos objetos. Para este tutorial, vamos a usar grupos para marcar y manipular objetos para ser salvadospero otros metodos son ciertamente posibles.

Vamos a empezar agregando objetos que queremos salvar al grupo "Persist". Como en el tutorial *Scripting (continuación)*, podemos hacer esto a traves de GUI o de script. Agreguemos los nodos relevantes usando la GUI:



Una vez que esto esta hecho vamos a necesitar salvar el juego, podemos obtener todos los objetos para salvarlos y luego decirle a todos ellos que salven con este script:

```
var nodossalvados = get_tree().get_nodes_in_group("Persist")
    for i in nodossalvados:
    # Ahora podemos llamar nuestra funcion de salvar en cada nodo.
```

Serializando

El siguiente paso es serializar los datos. Esto vuelve mucho mas sencillo leer y guardar en disco. En este caso, estamos asumiendo que cada miembro del grupo Persist es un nodo instanciado y por lo tanto tiene path. GDScript tiene funciones de ayuda para esto, como *Dictionary.to_json()* y *Dictionary.parse_json()*, asi que usaremos este diccionario. Nuestro nodo necesita contener una funcion de salvar para que regresas estos datos. La funcion de salvar sera similar a esto:

```
func salvar():
    var salvardicc = {
        nombrearchivo=get_filename(),
        padre=get_parent().get_path(),
        posx=get_pos().x, #Vector2 no es soportado por json
        posy=get_pos().y,
        ataque=ataque,
        defensa=defensa,
        saludactual=saludactual,
        saludmaxima=saludmaxima,
        daño=daño,
        regen=regen,
        experiencia=experiencia,
        TNL=TNL,
        nivel=nivel,
        ataquecrece=ataquecrece,
        defensacrece=defensacrece,
        saludcrece=saludcrece,
        estavivo=estavivo,
        ultimo_ataque=ultimo=ataque
        }
    return salvardicc
```

Esto nos da un diccionario con el estilo `{ "nombre_variable":valor_esa_variable }` el cual puede ser util para cargar.

Salvar y leer datos

Como cubrimos en el tutorial *Sistema de archivos (Filesystem)*, necesitaremos abrir un archivo y escribirlo y luego leer de el. Ahora que tenemos una forma de llamar nuestros grupos y obtener los datos relevantes, vamos a usar to_json() para convertirlos en un string que sea facilmente guardado y lo grabaremos en un archivo. Haciendo esto asegura que cada linea es su propio objeto de forma que tenemos una forma facil de obtener los datos desde el archivo tambien.

```
# Nota: Esto puede ser llamado desde cualquier parte dentro del arbol. Esta funcion␣
↪es independiente del camino.
# Ve a traves de todo en la categoria persistente y pideles el regreso de un␣
↪diccionario de variables relevantes

func salvar_juego():
    var juegosalvado = File.new()
    juegosalvado.open("user://juegosalvado.save", File.WRITE)
    var salvanodos = get_tree().get_nodes_in_group("Persist")
    for i in salvanodos:
        var datodenodo = i.save()
        juegosalvado.store_line(nodedata.to_json())
    juegosalvado.close()
```

Juego salvado! Cargarlo es bastante simple tambien. Para eso vamos a leer cada linea, usar parse_json() para leerlo de nuevo a un diccionario, y luego iterar sobre el diccionario para leer los valores. Pero primero necesitaremos crear el objeto y podemos usar el nombre de archivo y valores del padre para lograrlo. Aqui esta nuestra funcion de carga:

```
# Nota: Esto puede ser llamado desde cualqueir lugar dentro del
# arbol. Esta funcion es independiente del camino.

func load_game():
    var juegosalvado = File.new()
    if !juegosalvado.file_exists("user://juegosalvado.save"):
        return #Error!  No tenemos un juego salvado para cargar

    # Necesitamos revertir el estado del juego asi no clonamos objetos durante la␣
↪carga.
    # Esto variara mucho dependiendo de las necesidades del proyecto, asi que ten␣
↪cuidado con este paso.
    # Para nuestro ejemplo, vamos a lograr esto borrando los objetos que se pueden␣
↪salvar.
    var savenodes = get_tree().get_nodes_in_group("Persist")
    for i in savenodes:
        i.queue_free()

    # Carga el archivo linea por linea y procesa ese diccionario para restaurar los␣
↪objetos que representan
    var currentline = {} # dict.parse_json() requiere un diccionario declarado
    juegosalvado.open("user://juegosalvado.save", File.READ)
    while (!juegosalvado.eof_reached()):
        currentline.parse_json(juegosalvado.get_line())
        # Primero necesitamos crear el objeto y agregarlo al arbol ademas de ajustar␣
↪su posicion
        var newobject = load(currentline["filename"]).instance()
        get_node(currentline["parent"]).add_child(newobject)
        newobject.set_pos(Vector2(currentline["posx"],currentline["posy"]))
        # Ahora ajustamos las variables restantes
        for i in currentline.keys():
            if (i == "filename" or i == "parent" or i == "posx" or i == "posy"):
                continue
```

```
            newobject.set(i, currentline[i])
    juegosalvado.close()
```

Y ahora podemos salvar y cargar un numero arbitrario de objetos distribuidos practicamente en cualquier lado a traves del arbol de escena! Cada objeto puede guardar diferentes datos dependiendo en que necesita salvar.

Algunas notas

Podemos haber pasado por alto un paso, pero ajustar el estado del juego a uno acondicionado para empezar a cargar datos puede ser muy complicado. Este paso va a necesitar ser fuertemente personalizado basado en las necesidades de un proyecto individual.

Esta implementacion asume que ningun objeto persistente es hijo de otro objeto persistente. Ello crearia paths invalidos. Si esta es una de las necesidades del proyecto esto tiene que ser considerado. Salvar objetos en etapas (objetos padre primero) de forma que esten disponibles cuando los objetos hijos son cargados asegurara que estan disponibles para la llamada add_child(). Tambien se necesitara algun forma de vincular hijos a padres ya que el nodepath probablemente sera invalido.

2.2.4 Encripando juegos salvados

Porque?

Porque el mundo de hoy no es el de ayer. Una oligarquia capitalista esta a cargo del mundo y nos fuerza a consumir para mentener los engranajes de esta sociedad decadente en marcha. Es un mercado de pobres almas forzadas a consumir de forma compulsiva contenido digital para olvidar la miseria de su vida diaria, en cualquier momento breve que tienen donde no estan siendo usados par producir mercancias o servicios para la clase dominante. Estos individuos necesitar mantenerse enfocados en sus video juegos (ya que de no hacerlo produciria una tremenda angustia existencial), por lo que van hasta el punto de gastar plata con el fin de ampliar sus experiencias, y la forma preferida de hacerlo es con compras dentro de la app y monedas virtuales.

Pero, imagina si alguien encuentra la forma de editar los juegos salvados y asignar items y dinero sin esfuerzo? Esto seria terrible, porque ayudaria a los jugadores a consumir contenido mucho mas rapido, y por lo tanto que se termine antes de lo esperado. Si esto sucede no tendran nada que evite que piensen, y la agonia tremenda de darse cuenta de su propia irrelevancia nuevamente tendria el control de sus vidas.

No, definitivamente no queremos que todo esto suceda, por lo que veamos como hacer para encriptar juegos salvados y proteger el orden mundial.

Como?

La clase *File* es simple de usar, solo abre una locacion y lee/escribe datos (enteros, cadenas y variantes). Para crear un archivo encriptado, una passphrase debe proveerse, como esta:

```
var f = File.new()
var err = f.open_encrypted_with_pass("user://juegosalvado.bin", File.WRITE, "miclave")
f.store_var(estado_juego)
f.close()
```

Esto volvera imposible de leer el archivo para los usuarios, pero no evitara que compartan juegos salvados. Para resolver esto, usando el id unico o alguna forma identificacion de usuario es necesaria, por ejemplo:

```
var f = File.new()
var err = f.open_encrypted_with_pass("user://juegosalvado.bin", File.WRITE, OS.get_
↪unique_ID())
f.store_var(estado_juego)
f.close()
```

Esto es todo! Gracias por tu cooperacion, ciudadano.

2.3 Internacionalización

2.3.1 Internacionalizando juegos

Introducción

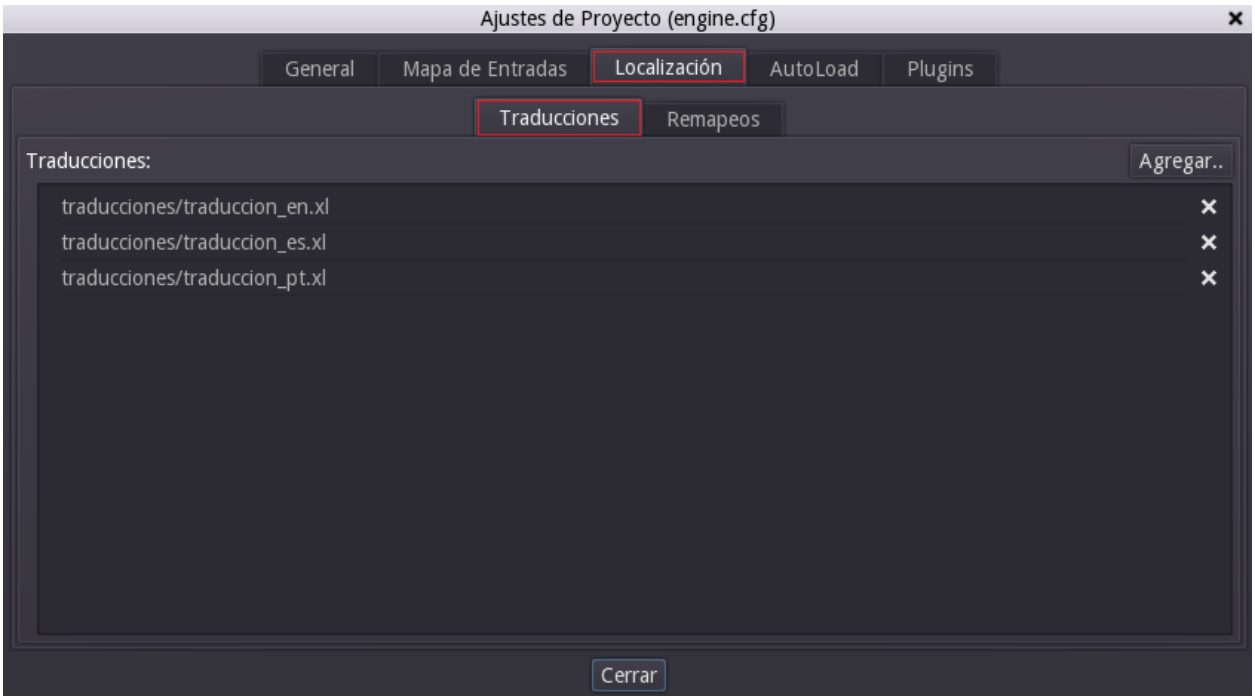
"It would be great if the world would speak just one language." Desgraciadamente para nosotros desarrolladores, este no es el caso. Aunque en general no es un requerimiento grande cuando se desarrollan juegos indies o de nicho, es también muy común que los juegos que van a ir a un mercado mas masivo requiera localización.

Godot ofrece muchas herramientas para hacer este proceso sencillo, por lo que esto tutorial se parece a una colección de consejos y trucos.

La localización es usualmente hecho por estudios específicos contratados para hacer el trabajo y, a pesar de la enorme cantidad de software y formatos de archivos disponibles para esto, la forma mas común de localización hasta este día es con planillas. El proceso de crear las planillas e importarlas ya esta cubierto en el tutorial *Importing translations*, por lo que este puede ser visto mas como una continuación de aquel.

Configurando la traducción importada

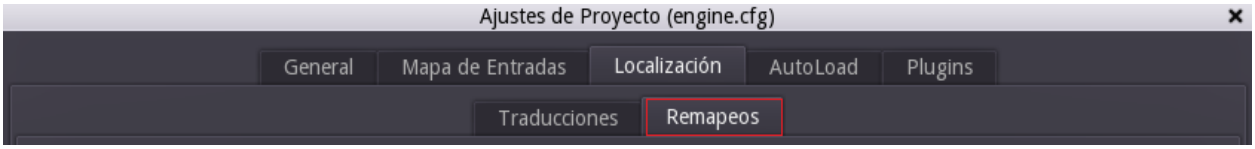
Las traducciones puede ser actualizadas y re-importadas cuando cambian, pero de todas formas deben ser agregadas al proyecto. Esto es hecho en Escena > Configuración de Proyecto > Localización:



Este dialogo permite agregar o quitar traducciones que afectan al proyecto entero.

Recursos de localización

También es posible ordenarle a Godot que abra versiones alternativas de assets (recursos) dependiendo el lenguaje actual. Para esto existe la pestaña "Remaps":



Selecciona el recurso a ser re-mapeado, y las alternativas para cada idioma

Convertir claves a texto

Algunos controles como *Button*, *Label*, etc. van a traer una traducción cada vez que sean ajustadas a una clave en lugar de texto. Por ejemplo, si una etiqueta esta asignada a "PANTALLA_PRINCIPAL_SALUDO1" y una clave para diferentes lenguajes existe en las traducciones, esto será convertido automáticamente. Pero este proceso es hecho durante la carga, por lo que si el proyecto en cuestión tiene un dialogo que permite cambiar el lenguajes en los ajustes, las escenas (o al menos la escena de ajustes) debe ser cargada nuevamente para que el nuevo texto tenga efecto. fect.

Por código, la función *Object.tr()* puede ser utilizada. Esto solo buscara el texto en las traducciones y convertirlo si es encontrado:

```
level.set_text(tr("NOMBRE_NIVEL_5"))
status.set_text(tr("ESTADO_JUEGO_" + str(indice_estado)))
```

Haciendo controles que cambian de tamaño

El mismo texto en diferentes lenguajes puede variar de gran forma de largo. Para esto, asegúrate de leer el tutorial en *Tamaño y anclas*, ya que tener controles que ajusten su tamaño dinámicamente puede ayudar. Los *Container* pueden ser muy útiles, así como las opciones múltiples en *Label* para acomodar el texto.

TranslationServer (Servidor de traducción)

Godot tiene un servidor para manejar la administración de traducción de bajo nivel llamada *TranslationServer*. Las traducciones pueden ser agregadas o quitadas en tiempo de ejecución, y el lenguaje actual puede ser cambiado también.

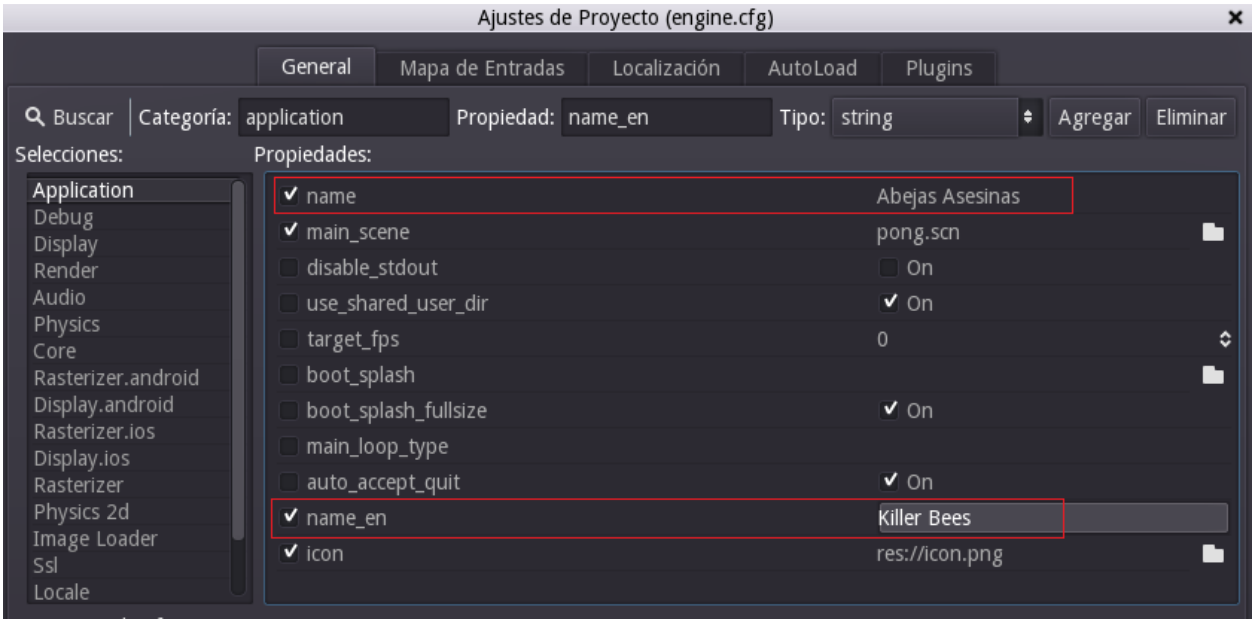
Línea de Comando

El lenguaje puede ser probado cuando se corre Godot desde una línea de comandos. Por ejemplo, para probar el juego en francés, los siguientes argumentos son suministrados:

```
c:\MiJuego> godot -lang fr
```

Traducir el nombre de proyecto.

El nombre de proyecto se vuelve el nombre de la aplicación cuando se exporta a diferentes sistemas y plataformas. Para especificar el nombre de proyecto en mas de un lenguaje, crea un nuevo ajuste de aplicación/nombre en el dialogo de configuración de proyecto y agrégale el identificador de lenguaje. Por ejemplo:



Como siempre, si no sabes el código de un lenguaje o zona, *chequea la lista*.

2.4 Game flow

2.4.1 Pausando juegos

Pausa?

En la mayoría de los juegos es deseable, en algún punto, interrumpir el juego para hacer algo mas, como descansar o cambiar opciones. Sin embargo esto no es tan simple como parece. El juego podría estar detenido, pero puede quererse que algunos menús y animaciones continúen trabajando.

Implementar un control fino de lo que puede ser pausado (y lo que no) es un montón de trabajo, por lo que un framework simple para pausar es incluido en Godot.

Como funciona la pausa

Para poner el modo pausa, el estado de pausa debe ser ajustado. Esto es hecho llamando *SceneTree.set_pause()* con el argumento "true":

```
get_tree().set_pause(true)
```

Hacerlo tendrá el siguiente comportamiento:

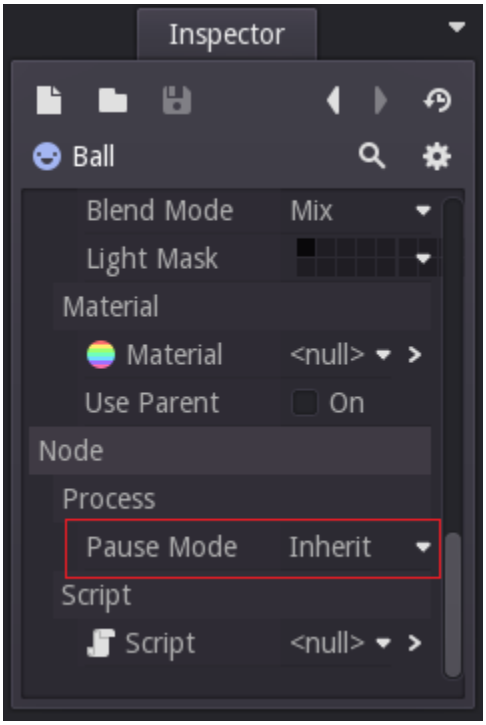
- La física 2D y 3D será detenida.
- _process y _fixed_process no serán mas llamados en los nodos.

- _input y _input_event no serán mas llamados tampoco.

Esto efectivamente detiene del juego por completo. Llamar esta función desde un script, por defecto, resultara en un estado no recuperable (nada seguirá funcionando!).

Lista blanca de nodos

Antes de habilitar la pausa, asegúrate que los nodos que deben seguir trabajando durante la pausa están en la lista blanca. Esto es hecho editando la propiedad "Pause Mode" en el nodo:



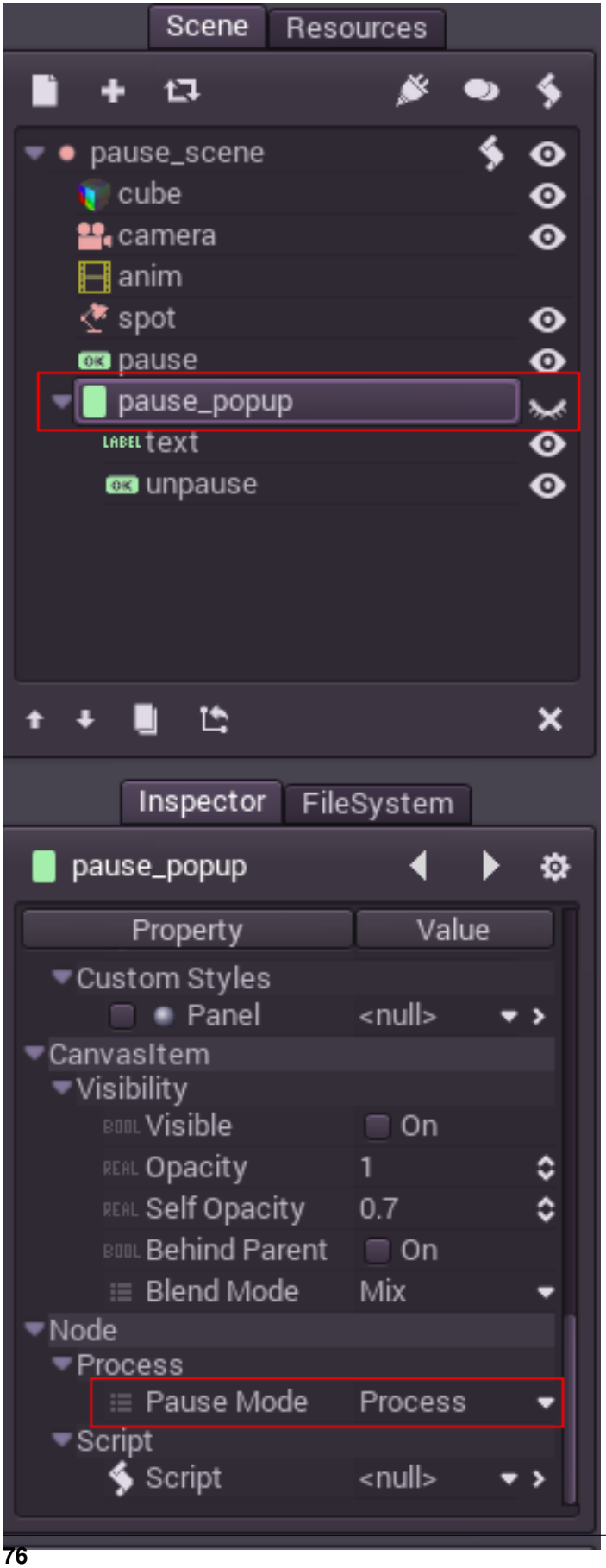
Por defecto todos los nodos tienen esta propiedad en el estado "Inherit" (Heredar). Esto significa, que solo van a procesar (o no) dependiendo en que esta ajustada esta propiedad en el nodo padre. Si el padre esta en "Inherit", entonces se chequeara el abuelo, y así sigue. Al final, si un estado no puede ser encontrado en ninguno de los abuelos, el estado de pausa de SceneTree es usado. Esto implica que, por defecto, cuando el juegos es pausado todo nodo será pausado.

Asique los tres estados posibles para nodos son:

- **Inherit**: Procesar dependiendo en el estado del padre, abuelo, etc. El primer padre que tenga un estado no heredado.
- **Stop**: Detiene el nodo no importa que (e hijos en el modo Inherit). Cuando en pausa, este nodo no procesa.
- **Process**: Procesar el nodo no importa que (y los hijos en modo Inherit). Pausado o no, este nodo procesara.

Ejemplo

Un ejemplo de esto es crear una ventana emergente o panel con controles dentro, y ajustar su modo pausa a "Process" luego solo esconderlo:


Scene
Resources
pause_scene
cube
camera
anim
spot
pause
pause_popup
text
unpause
Inspector
FileSystem
pause_popup
Property
Value
Custom Styles
Panel
<null>
CanvasItem
Visibility
Visible
On
Opacity
1
Self Opacity
0.7
Behind Parent
On
Blend Mode
Mix
Node
Process
Pause Mode
Process
Script
Script
<null>

Solo con ajustar la raíz de la ventana emergente de pausa a "Process", todos los hijos y nietos heredaran el estado. De esta forma, esta rama del árbol de escena continuara trabajando cuando este en pausa.

Finalmente, haz que cuando el botón de pausa es presionado (cualquier botón servirá), se habilite la pausa y muestre la pantalla de pausa.

```
func _on_pause_button_pressed():
    get_tree().set_pause(true)
    get_node("pause_popup").show()
```

Para quitar la pausa, solo has lo opuesto cuando la ventana de pausa es cerrada:

```
func _on_pause_popup_close_pressed():
    get_node("pause_popup").hide()
    get_tree().set_pause(false)
```

Y eso debería ser todo!

2.4.2 Carga en segundo plano

Cuando cambias la escena principal de tu juego (por ejemplo yendo a un nuevo nivel), puedes querer mostrar una pantalla de carga con alguna indicacion del progreso. El metodo principal de carga (`ResourceLoader::load` or just `load` from gdscript) bloquea tu thread (hilo) mientras los recursos estan siendo cargados, por lo que no es bueno. Este documento discute la clase `ResourceInteractiveLoader` para pantallas de carga mas suaves.

ResourceInteractiveLoader

La clase `ResourceInteractiveLoader` te permite cargar el recurso en etapas. Cada vez que el metodo `poll` es llamado, una nueva etapa es cargada, y el control se devuelve al que llamo. Cada etapa es generalmente un sub-recurso que es cargado por el recurso principal. Por ejemplo, si vas a cargar una escena que tiene 10 imagenes, cada imagen sera una etapa.

Uso

La forma de usarlo es generalmente la siguiente:

Obteniendo un ResourceInteractiveLoader

```
Ref<ResourceInteractiveLoader> ResourceLoader::load_interactive(String p_path);
```

Este metodo te dara un ResourceInteractiveLoader que usaras para gestionar la operacion de carga.

Polling

```
Error ResourceInteractiveLoader::poll();
```

Usa este metodo para avanzar el progreso de la carga. Cada llamada a `poll` cargara la siguiente etapa de tu recurso. Ten en cuenta que cada etapa es un recurso "atomico" completo, como una imagen, una malla, por lo que llevara varios frames para cargarlo.

Devuleve `OK` si no hay errores, `ERR_FILE_EOF` cuando la carga termina. Cualquier otro valor implicar que hubo un error y la carga se detuvo.

Progreso de carga (opcional)

Para consultar el progreso de la carga, usa los siguientes metodos:

```
int ResourceInteractiveLoader::get_stage_count() const;
int ResourceInteractiveLoader::get_stage() const;
```

`get_stage_count` devuelve el numero total de etapas a cargar. `get_stage` devuelve la etapa actual siendo cargada.

Forzar que complete (opcional)

```
Error ResourceInteractiveLoader::wait();
```

Usa este metodo si necesitas cargar el recurso entero en el frame actual, sin pasos adicionales.

Obtener el recurso

```
Ref<Resource> ResourceInteractiveLoader::get_resource();
```

Si todo va bien, usa este metodo para traer tu recurso cargado.

Ejemplo

Este ejemplo demuestra como cargar una nueva escena. Consideralo en el contexto del ejemplo *Singletons (AutoCarga)*.

Primero debemos declarar algunas variables e inicializar "current_scene" con la escena principal del juego:

```
var loader
var wait_frames
var time_max = 100 # milisegundos
var current_scene

func _ready():
    var root = get_tree().get_root()
    current_scene = root.get_child(root.get_child_count() -1)
```

La funcion `goto_scene` es llamada desde el juego cuando la escena necesita ser conmutada. Pide una carga interactiva, y llama `set_progress(true)` para empezar a hacer polling de la carga en la llamada de retorno `_progress`. Tambien inicia una animacion de "carga", que puede mostrar una barra de progreso o pantalla de carga,

etc.

```
func goto_scene(path): # el juego pide cambiar a este escena
    loader = ResourceLoader.load_interactive(path)
    if loader == null: # chequear errores
        show_error()
        return
    set_process(true)

    current_scene.queue_free() # dejar de lado la escena vieja
```

```
    # Empieza tu animacion "loading..."("cargando"...)
    get_node("animation").play("loading")

    wait_frames = 1
```

_process es donde se le hace polling a la carga. poll es llamado, y luego nos encargamos del valor de retorno de esa llamada. OK significa mantente haciendo polling, ERR_FILE_EOF que la carga esta completa, cualquier otra cosa que hubo un error. Tambien fijate que salteamos un frame (via wait_frames, ajustada en la funcion ´´goto_scene´´) para permitir que la pantalla de carga se muestre.

Fijate como usar OS.get_ticks_msec para controlar cuanto tiempo bloqueamos el thread. Algunas etapas podran cargarse realmente rapido, lo que significa que podriamos sumar mas de una llamada a poll en un frame, algunas pueden tomar mucho mas que el valor de time_max, por lo que ten en cuenta que no tenemos control preciso sobre estos timing.

```
func _process(time):
    if loader == null:
        #  no hace falta procesar mas
        set_process(false)
        return

    if wait_frames > 0: # espera por frames para permitir que la animacion "loading"␣
↪se muestre
        wait_frames -= 1
        return

    var t = OS.get_ticks_msec()
    while OS.get_ticks_msec() < t + time_max: # usa "time_max" para controlar durante␣
↪cuanto tiempo bloqueamos este thread


        # haciendole poll al loader
        var err = loader.poll()

        if err == ERR_FILE_EOF: # la carga termino
            var resource = loader.get_resource()
            loader = null
            set_new_scene(resource)
            break
        elif err == OK:
            update_progress()
        else: # error durante la carga
            show_error()
            loader = null
            break
```

Algunas funciones extra ayudantes. update_progress actualiza la barra de progreso o puede tambien actualizar una animacion en pausa (la animacion representa la carga completa de principio a fin). set_new_scene pone la escena recientemente cargada en el arbol. Como es una escena que se esta cargando, instance() necesita ser llamado en el recurso obtenido desde el que carga (loader).

```
func update_progress():
    var progress = float(loader.get_stage()) / loader.get_stage_count()
    # actualizamos tu barra de progreso?
    get_node("progress").set_progress(progress)

    # o actualizamos una animacion de progreso?
```

```
    var len = get_node("animation").get_current_animation_length()

    # llama esto en una animacion en pausa
    # Usa "true" como el segundo parametro para forzar la animacion a actualizarse
    get_node("animation").seek(progress * len, true)

func set_new_scene(scene_resource):
    current_scene = scene_resource.instance()
    get_node("/root").add_child(current_scene)
```

Usando multiples threads

ResourceInteractiveLoader puede ser usando desde multiples threads. Un par de cosas a tener en cuenta si lo intentas:

Usa un semaforo

Mientras tu thread espera por el pedido de un nuevo recurso a el thread principal, usa un semaforo para dormir (en lugar de un bucle ocupado o algo similar).

No bloquear el thread principal durante el polling

Si tenes un mutex para permitir llamados desde el thread principal a tu clase de carga, no le hagas lock mientras llamas a `pol` en el que carga. Cuando un recurso termina de cargarse, puede requerir algunos recursos desde las APIs de bajo nivel (VisualServer, etc), lo cual puede necesitar hacer lock al thread principal para adquirirlos. Esto puede causar un deadlock (punto muerto) si el thread principal esta esperando por tu mutex mientras tu thread esta esperamdo cargar un recurso.

Clase ejemplo

Tu puedes encontrar una clase de ejemplo para cargar recursos en threads aqui: `resource_queue.gd`. El uso es el siguiente:

```
func start()
```

Llamar luego que instancias la clase para empezar el thread.

```
func queue_resource(path, p_in_front = false)
```

Encola un recurso. Usa los parametros opcionales “p_in_front” para ponerlo al frente de la cola.

```
func cancel_resource(path)
```

Remueve un recurso desde la cola, descartando cualquier carga hecha.

```
func is_ready(path)
```

Retorna true si el recurso termino de cargar y esta listo para ser recuperado.

```
func get_progress(path)
```

Obten el progreso de un recurso. Retorna -1 para error (por ejemplo si el recurso no esta en la cola), o un numero entre 0.0 y 1.0 con el progreso de la carga. Usar principalmente con propositos cosmeticos (actualizar barras de progreso, etc), usa `is_ready` para encontrar si un recurso esta realmente listo.

```
func get_resource(path)
```

Retorna el recurso completamente cargado, o null para error. Si el recurso no termino de cargar (`is_ready` regresa falso), bloqueara tu thread y terminara la carga. Si el recurso no esta en la misma cola, llamara `ResourceLoader::load` para cargarla con normalidad y retornarla.

Ejemplo:

```
# inicializar
queue = preload("res://resource_queue.gd").new()
queue.start()

# supone que tu juego empieza con una presentacion de 10 segundos, durante los cuales␣
↪el usuario no puede
# interactuar con el juego. Durante ese tiempo sabemos que no usaran el menu pausa,␣
↪por lo que podemos
# encolarlo para que cargue durante la presentacion:
queue.queue_resource("res://pause_menu.xml")
start_curscene()

# mas tarde cuando el usuario presiona el boton pausa por primer vez:
pause_menu = queue.get_resource("res://pause_menu.xml").instance()
pause_menu.show()

# cuando se necesita una escena nueva:
queue.queue_resource("res://level_1.xml", true) # usa "true" como segundo parametro␣
↪para ponerlo al frente
                                                # de la cola, pausando la carga de␣
↪cualqueir otro
                                                # recurso

# para chequear el progreso
if queue.is_ready("res://level_1.xml"):
    show_new_level(queue.get_resource("res://level_1.xml"))
else:
    update_progress(queue.get_process("res://level_1.xml"))

#   cuando el usuario sale fuera de la zona trigger en tu juego Metroidvania:
queue.cancel_resource("res://zone_2.xml")
```

Nota: este codigo en su forma actual no ha sido testeado en escenarios reales. Preguntale a punto en IRC (#godotengine en irc.freenode.net) por ayuda.

2.4.3 Manejando requerimientos de quit (abandono)

Abandonando

La mayoria de las plataformas tienen la opcion de pedir que la aplicacion abandone. En escritorios, esto es usualmente hecho con el icono "x" en la barra de titulo de la ventana. En Android, el boton back (atras) es usado para abandonar cuando estas en la ventana principal (o para ir atras si no es la ventana principal).

Manejando la notificacion

El *MainLoop* tiene una notificacion especial que es enviada a todos los nodos cuando se requiere abandonar: MainLopp.NOTIFICATION_WM_QUIT.

Se maneja de la siguiente forma (en cualquier nodo):

```
func _notification(what):
    if (what == MainLoop.NOTIFICATION_WM_QUIT_REQUEST):
        get_tree().quit() # comportamiento por defecto
```

Cuando desarrollas aplicaciones moviles, abandonar no es deseado a no ser que el usuario este en la pantalla principal, por lo que el comportamiento puede cambiar.

Es importante notar que por defecto, las aplicaciones de Godot tienen comportamiento incorporado para abandonar cuando se les pide quit, esto puede ser cambiado:

```
get_tree().set_auto_accept_quit(false)
```

CAPÍTULO 3

Tutoriales 2D

3.1 Gráficos

3.1.1 Capas Canvas

Viewport e ítems Canvas

Los nodos 2D regulares, como Node2D heredan desde *CanvasItem*, que es la base para todos los nodos 2D. CanvasItems pueden ser organizados en arboles y heredaran sus transformaciones. Esto implica que cuando mueves al padre, el hijo se moverá también.

Estos nodos son puestos como hijos directos o indirectos de un *Viewport*, y serán mostrados a través de el.

Viewport tiene una propiedad "canvas_transform" *Viewport.set_canvas_transform()*, que permite transformar toda la jerarquía CanvasItem por una transformación *Matrix32* personalizada. Los nodos como *Camera2D*, trabajan cambiando dicha transformación.

Cambiar la transformación de canvas es útil porque es mucho mas eficiente que mover que mover el ítem canvas raíz (y por lo tanto toda la escena). Canvas transform es una matriz simple que afecta al dibujo 2D por completo, por lo que es la forma mas eficiente de hacer scrolling.

No es suficiente...

Pero esto no es suficiente. A menudo hay situaciones donde el juego o aplicación puede no querer *todo* transformado por la transformación canvas. Ejemplos de esto son:

- **Fondos Parallax**: Los fondos que se mueven mas lento que el resto del nivel.
- **HUD**: Visualización Head's up, o interfaz de usuario. Si el mundo se mueve, el contador de vida, puntaje, etc. debe mantenerse estático.
- **Transiciones**: Efectos usados para transiciones (fades, blends) pueden también querer mantenerse en una posición fija.

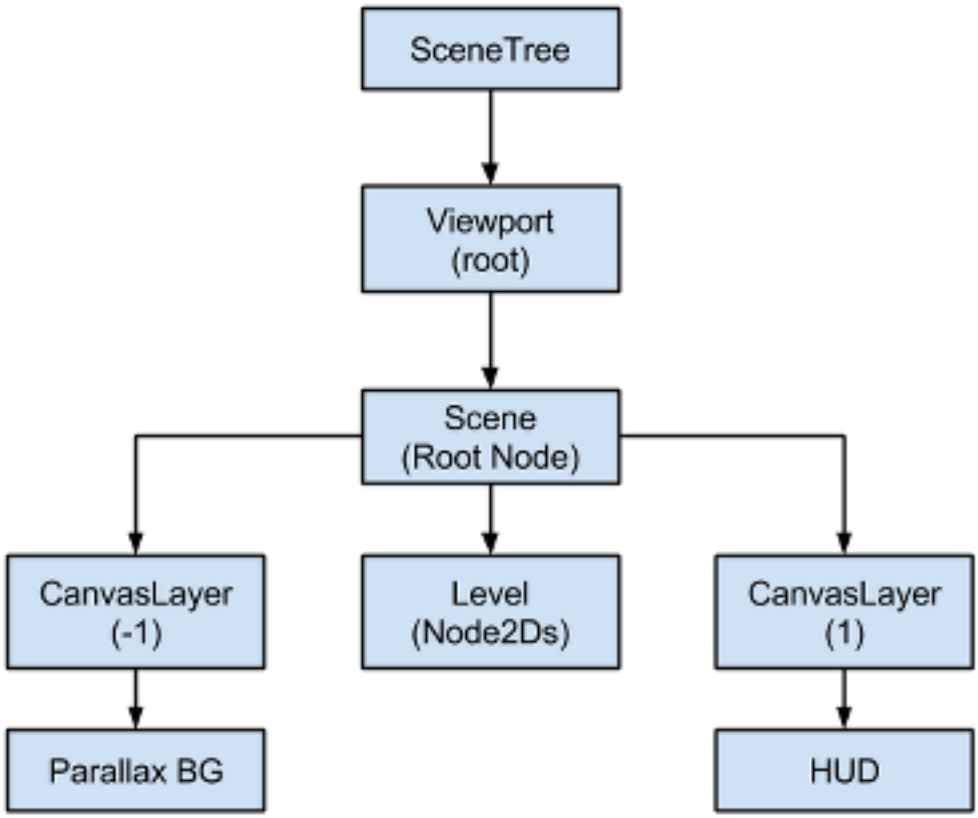
Como pueden estos problemas ser resueltos en un único árbol de escena?

CanvasLayers

La respuesta es *CanvasLayer*, que es un nodo que agrega una capa 2D de renderizacion 2D separada para todos sus hijos y nietos. Los hijos del viewport van a mostrarse por defecto en el layer "0", mientras un CanvasLayer va a dibujarse en cualquier numero (profundidad) de capa. Las capas con un numero mayor serán dibujados por encima de los que tienen menor numero. CanvasLayers también tiene su propia transformación, y no depende de la transformación de las otras capas. Esto permite que la UI este fija en su lugar, mientras el mundo se mueve.

Un ejemplo de esto es crear un fondo parallax. Esto puede ser hecho con un CanvasLayer en la capa "-1". La pantalla con los puntos, contador de vida y botón de pausa pueden también ser creada en la capa "1".

Aquí un diagrama de como funciona:



Los CanvasLayers son independientes del orden de árbol, y solo dependen de su numero de capa, por lo que pueden ser instanciados cuando se precisan.

Rendimiento

Aunque no debería haber ninguna limitación de rendimiento, no es recomendado usar una excesiva cantidad de layers para acomodar el orden de dibujado de los nodos. La forma mas optima siempre será acomodarlos por orden de árbol. Los nodos 2D también tienen una propiedad para controlar su orden de dibujado (see *Node2D.set_z()*).

3.1.2 Viewport y transformaciones canvas

Introducción

Este tutorial esta creado por un tema que es algo oscuro para la mayoría de los usuarios, y explica todas las transformaciones 2D que suceden en los nodos desde el momento que dibujan su contenido localmente a el tiempo que son dibujados en la pantalla.

Transformaciones Canvas

Como mencionamos en el tutorial previo, *Capas Canvas*, todos los nodos CanvasItem (recuerda que los nodos basados en Node2D y Control usan CanvasItem como su raíz común) van a residir en una *Capa Canvas*. Toda capa canvas tiene transformación (traslación, rotación, escala, etc.) que puede ser accedida como una *Matrix32*.

También cubierto en el tutorial previo, los nodos son dibujados por defecto en el Layer 0, en el canvas incorporado. Para poner nodos en diferentes capas, un nodo :ref:‘CanvasLayer <class_CanvasLayer> puede ser usado.

Transformaciones globales de canvas

Los Viewports también tienen una transformación Global Canvas (que también es una *Matrix32*). Esta es la transformación maestra y afecta todas las transformaciones individuales *Canvas Layer*. Generalmente esta transformación no es de mucha utilidad, pero es usada en el CanvasItem Editor en el editor Godot.

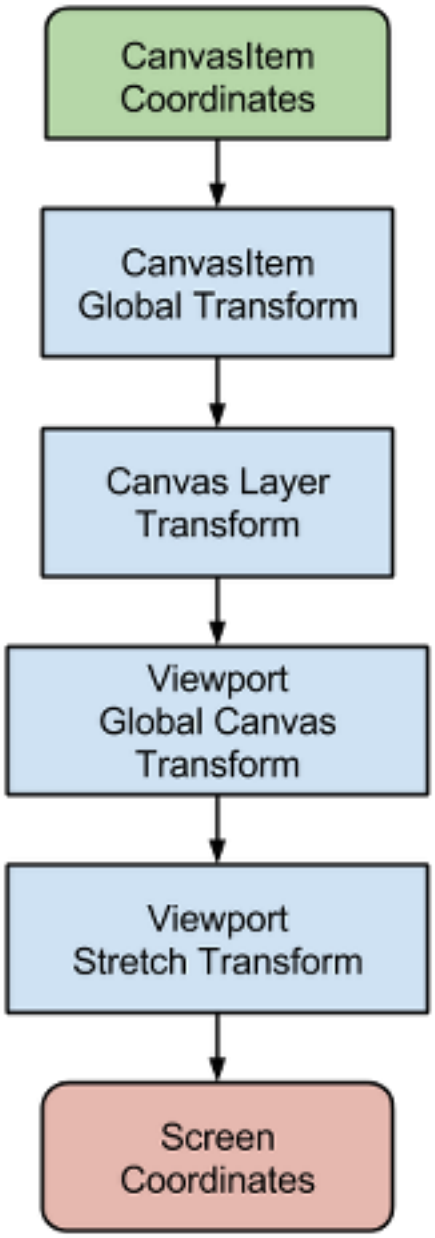
Transformación de estiramiento

Finalmente, los viewports tienen *Stretch Transform*, la cual es usada cuando se cambia de tamaño o se estira la pantalla. Esta transformación es usada internamente (como se describe en *Resoluciones múltiples*), pero puede también ser ajustada manualmente en cada viewport.

Los eventos de entrada recibidos en la llamada de retorno *MainLoop._input_event()* son multiplicados por esta transformación, pero carece de las superiores. Para convertir coordenadas InputEvent a coordenadas locales CanvasItem, la funcion *CanvasItem.make_input_local()* fue agregada para comodidad.

Orden de transformación

Para que una coordenada en las propiedades locales CanvasItem se vuelva una coordenada de pantalla, la siguiente cadena de transformaciones debe ser aplicada:



Funciones de transformación

Obtener cada transformación puede ser logrado con las siguientes funciones:

Tipo	Transformacion
CanvasItem	*CanvasItem.get_global_transform()*
CanvasLayer	*CanvasItem.get_canvas_transform()*
CanvasLayer+GlobalCanvas+Stretch	*CanvasItem.get_viewport_transform()*

Finalmente pues, para convertir coordenadas CanvasItem locales a coordenadas de pantalla, solo multiplica en el siguiente orden:

```
var coord_pant = get_viewport_transform() + ( get_global_transform() + pos_local )
```

Ten en mente, sin embargo, que en general no es deseable trabajar con coordenadas de pantalla. El enfoque recomendado es simplemente trabajar con coordenadas Canvas (`CanvasItem.get_global_transform()`), para permitir que el cambio de tamaño automático de resolución de pantalla funcione adecuadamente.

Alimentando eventos de entrada personalizados

Es a menudo deseable alimentar eventos de entrada personalizados al árbol de escena. Con el conocimiento anterior, para lograr esto correctamente, debe ser hecho de la siguiente manera:

```
var pos_local = Vector2(10,20) # local a Control/Node2D
var ie = InputEvent()
ie.type = InputEvent.MOUSE_BUTTON
ie.button_index = BUTTON_LEFT
ie.pos = get_viewport_transform() + (get_global_transform() + pos_local)
get_tree().input_event(ie)
```

3.1.3 Dibujo personalizado en 2D

Porque?

Godot ofrece nodos para dibujar sprites, polígonos, partículas y todo tipo de cosas. Para la mayoría de los casos esto es suficiente, pero no siempre. Si se desea algo que no esta soportado, y antes de llorar de miedo y angustia porque un nodo para dibujar ese-algo-especifico no existe... seria bueno saber que es posible hacer que cualquier nodo 2D (sea basado en *Control* o *Node2D*) dibuje comandos personalizados. Es *realmente* fácil de hacer.

Pero...

Dibujar personalizadamente de forma manual en un nodo es *realmente* útil. Aquí algunos ejemplos de porque:

- Dibujar formas o lógica que no esta manejada por nodos (ejemplos: hacer un nodo que dibuje un circulo, una imagen con estelas, un tipo especial de polígono animado, etc).
- Las visualizaciones que no son tan compatibles con nodos: (ejemplo: una tabla de tetris). El ejemplo de tetris usa una función de dibujo personalizada para dibujar los bloques.
- Gestionar la lógica de dibujo de una gran cantidad de objetos simples (en los cientos de miles). Usar mil nodos probablemente ni cerca tan eficiente como dibujar, pero mil llamadas de dibujo son baratas. Chequea el demo "Shower of Bullets" como ejemplo.
- Hacer un control de UI personalizado. Hay muchos controles disponibles, pero es fácil llegar a la necesidad de hacer uno nuevo, personalizado.

OK, como?

Agrega un script a cualquier nodo derivado de *CanvasItem*, como *Control* o *Node2D*. Sobrescribe la función _draw().

```
extends Node2D

func _draw():
    #Tus comandos de dibujo aquí
    pass
```

Los comandos de dibujo son descritos en la referencia de clase *CanvasItem*. Hay muchos de ellos.

Actualizando

La función _draw() es solo llamada una vez, y luego los comandos de dibujo son cacheados y recordados, por lo que llamadas siguientes no son necesarias.

Si se necesita redibujar porque un estado o algo cambio, solo llama *CanvasItem.update()* en ese mismo nodo y una nueva llamada a _draw() se realizara.

Aquí hay un ejemplo algo mas complejo. Una variable de textura que será redibujada si cambia:

```
extends Node2D

var texture setget _set_texture

func _set_texture(value):
    #si la variable textura es modificada externamente,
    #esta llamada de retorno es realizada.
    texture=value #La textura ha cambiado
    update() #actualiza el nodo

func _draw():
    draw_texture(texture,Vector2())
```

En algunos casos, puede deseare dibujar cada frame. Para esto, solo llama a update() desde la llamada de retorno _process(), asi:

```
extends Node2D

func _draw():
    #Tus comandos aquí
    pass

func _process(delta):
    update()

func _ready():
    set_process(true)
```

Un ejemplo: dibujar arcos circulares

Ahora usaremos la funcionalidad personalizada de dibujo del Motor Godot para dibujar algo que Godot no provee como funciones. Como un ejemplo, Godot provee una función draw_circle() que dibuja un circulo completo. Sin embargo, que hay sobre dibujar una porción de un circulo? Vas a necesitar programar una función para hacer esto, y dibujarla tu mismo.

Función Arc (Arco)

Un arco se define por sus parámetros soporte del circulo, esto es: la posición del centro, y el radio. El arco en si mismo se define por el ángulo donde empieza, y el ángulo donde termina. Estos son los 4 parámetros que tenemos que proveer a nuestro dibujo. También proveeremos el valor de color asi podemos dibujar el arco en diferentes colores si lo deseamos.

Básicamente, dibujar una forma en la pantalla requiere que sea descompuesta en un cierto numero de puntos unidos uno a uno con el siguiente. Como puedes imaginar, cuanto mayor cantidad de puntos tenga tu forma, mas suave aparecerá, pero mas pesada será en términos de costo de procesador. En general, si tu forma es enorme (o en 3D, cercana a la cámara), requerirá mas puntos para ser dibujado sin verse angular. Por el contrario, si tu forma es pequeña (o en 3D, lejana de la cámara), tu puedes reducir su numero de puntos y ahorrar costo de procesamiento. Esto se llama *Level of Detail (LoD)*. En nuestro ejemplo, vamos simplemente a usar un numero fijo de puntos, no importando el radio.

```
func draw_circle_arc( center, radius, angle_from, angle_to, color ):
    var nb_points = 32
    var points_arc = Vector2Array()

    for i in range(nb_points+1):
        var angle_point = angle_from + i*(angle_to-angle_from)/nb_points - 90
        var point = center + Vector2( cos(deg2rad(angle_point)), sin(deg2rad(angle_
↪point)) ) * radius
        points_arc.push_back( point )

    for indexPoint in range(nb_points):
        draw_line(points_arc[indexPoint], points_arc[indexPoint+1], color)
```

Recuerdas el numero de puntos en que nuestra forma tiene que ser descompuesta? Fijamos este numero en la variable nb_points al valor de 32. Luego, inicializamos un Vector2Array vacío, el cual es simplemente un arreglo de tipo Vector2.

El siguiente paso consiste en computar las posiciones actuales de estos 32 puntos que componen el arco. Esto es hecho en el primer for-loop: iteramos sobre el numero de puntos con los que queremos computar las posiciones, mas uno para incluir el ultimo punto. Primero determinamos el ángulo de cada punto, entre los ángulos de comienzo y fin.

La razón por la cual cada ángulo es reducido 90º es que vamos a computar posiciones 2D a partir de cada ángulo usando trigonometría (ya sabes, cosas de coseno y seno...). Sin embargo, para ser simple, cos() y sin() usan radianes, no grados. El ángulo de 0º (0 radian) empieza a las 3 en punto, aunque queremos empezar a contar a las 0 en punto. Entonces, vamos a reducir cada ángulo 90º para poder empezar a contar desde las 0 en punto.

La posición actual de un punto localizado en un circulo a un ángulo 'angle' (en radianes) es dada por Vector2(cos(angle), sen(angle)). Ya que cos() y sin() regresa valores entre -1 y 1, la posición esta localizada en un circulo de radio 1. Para tener esta posición en nuestro circulo de soporte, el cual tiene radio 'radius', debemos simplemente multiplicar la posición por 'radius'. Finalmente, necesitamos posicionar nuestro circulo de soporte en la posición 'center', lo cual es hecho al agregarlo a nuestro valor Vector2. Finalmente, insertamos el punto en el Vector2Array que fue previamente definido.

Ahora, necesitamos dibujar los puntos. Como puedes imaginar, no vamos simplemente a dibujar nuestros 32 puntos: necesitamos dibujar todo lo que esta entre medio de ellos. Podríamos haber computado cada punto nosotros mismos usando el método previo, y dibujarlos uno por uno, pero es muy complicado e ineficiente (a no ser que sea realmente necesario). Así que, vamos simplemente a dibujar líneas entre cada par de puntos. A no ser que el radio de nuestro circulo de soporte sea muy grande, el largo de cada línea entre un par de puntos nunca será suficientemente largo para verlos todos. Si esto sucede, solo tendríamos que incrementar el numero de puntos.

Dibujar el arco en la pantalla

Ahora tenemos una función que dibuja cosas en la pantalla, es tiempo de llamarla en la función _draw().

```
func _draw():
    var center = Vector2(200,200)
    var radius = 80
    var angle_from = 75
    var angle_to = 195
```

```
    var color = Color(1.0, 0.0, 0.0)
    draw_circle_arc( center, radius, angle_from, angle_to, color )
```

Result:

Función arco polígono

Podemos llevar esto un paso mas y escribir una función que dibuja la porción llana del disco definido por el arco, no solo la forma. Este método es exactamente el mismo que el previo, excepto que dibujamos un polígono en lugar de líneas:

```
func draw_circle_arc_poly( center, radius, angle_from, angle_to, color ):
    var nb_points = 32
    var points_arc = Vector2Array()
    points_arc.push_back(center)
    var colors = ColorArray([color])

    for i in range(nb_points+1):
        var angle_point = angle_from + i*(angle_to-angle_from)/nb_points - 90
        points_arc.push_back(center + Vector2( cos( deg2rad(angle_point) ), sin(␣
↪deg2rad(angle_point) ) ) * radius)
    draw_polygon(points_arc, colors)
```

Dibujo personalizado dinámico

Bien, ahora somos capaces de dibujar cosas personalizadas en la pantalla. Sin embargo, es muy estático: hagamos que esta forma gire alrededor de su centro. La solución para hacer esto es simplemente cambiar los valores angle_from y angle_to a lo largo del tiempo. Para nuestro ejemplo, vamos a incrementarlos en 50. Este valor de incremento tiene que permanecer constante, de lo contrario la velocidad de rotación cambiara de forma acorde.

Primero, tenemos que hacer ambos angle_from y angle_to variables globales al comienzo del script. También toma nota que puedes guardarlos en otros nodos y accederlo con get_node().

```
extends Node2D

var rotation_ang = 50
var angle_from = 75
var angle_to = 195
```

Hacemos que estos valores cambien en la función _process(delta). Para activar esta función, necesitamos llamar set_process(true) en la función _ready().

También incrementamos aquí nuestros valores angle_from y angle_to. Sin embargo, no debemos olvidar hacer wrap() a los valores resultantes entre 0º y 360º! Esto es, si el ángulo es 361º, entonces en realidad es 1º. Si no haces wrap de estos valores, el script funcionara correctamente pero los valores de ángulo crecerán mas y mas en el tiempo, hasta llegar al máximo valor entero que Godot puede manejar (2^31 - 1). Cuando esto sucede, Godot puede colgarse o producir comportamiento no esperado. Como Godot no provee una función wrap(), vamos a crearla aqui, ya que es relativamente simple.

Finalmente, no debemos olvidar llamar la función update(), que de forma automática llama a _draw(). De esta forma, puedes controlar cuando quieres refrescar el frame.

```
func _ready():
    set_process(true)

func wrap(value, min_val, max_val):
    var f1 = value - min_val
    var f2 = max_val - min_val
    return fmod(f1, f2) + min_val

func _process(delta):
    angle_from += rotation_ang
    angle_to += rotation_ang

    # solo hacemos wrap si los angulos son mayores que 360
    if (angle_from > 360 && angle_to > 360):
        angle_from = wrap(angle_from, 0, 360)
        angle_to = wrap(angle_to, 0, 360)
    update()
```

También, no olvides modificar la función _draw() para hacer uso de estas variables:

func _draw(): var center = Vector2(200,200) var radius = 80 var color = Color(1.0, 0.0, 0.0)

draw_circle_arc(center, radius, angle_from, angle_to, color)

Corrámoslo! Funciona, pero el arco gira extremadamente rápido! Que salió mal?

La razón es que tu GPU esta mostrando frames tan rápido como puede. Necesitamos "normalizar" el dibujo por esta velocidad. Para lograrlo, tenemos que hacer uso del parámetro 'delta' en la función _process(). 'delta' contiene el tiempo transcurrido entre los dos últimos frames mostrados. En general es pequeño (unos 0.0003 segundos, pero esto depende de tu hardware). Por lo que, usar 'delta' para controlar tu dibujado asegura a tu programa correr a la misma velocidad en todo hardware.

En nuestro caso, vamos a necesitar multiplicar nuestra variable 'rotation_angle' por 'delta' en la funcion _process(). De esta forma, nuestros 2 ángulos se incrementaran por un valor mucho menor, el cual depende directamente de la velocidad de renderizado.

```
func _process(delta):
    angle_from += rotation_ang * delta
    angle_to += rotation_ang * delta

    # solo hacemos wrap en los ángulos si ambos son mayores de 360
    if (angle_from > 360 && angle_to > 360):
        angle_from = wrap(angle_from, 0, 360)
        angle_to = wrap(angle_to, 0, 360)
    update()
```

Corrámoslo nuevamente! Esta vez, la rotación se muestra bien!

Herramientas

Dibujar tus propios nodos puede también ser deseable mientras los corres dentro del editor, para usar como pre visualización o visualización de alguna característica o comportamiento.

Recuerda solo usar la palabra clave "tool" en la parte superior del script (chequea la referencia *GDScript* si olvidaste que logra esto)

3.1.4 Shaders que leen la pantalla

Introducción

A menudo es deseable hacer un shader que lee de la misma pantalla en la que esta escribiendo. Las APIs 3D como OpenGL o DirectX hacen esto bastante difícil por limitaciones internas de hardware. Las GPUs son extremadamente paralelas, por lo que leer y escribir causa todo tipo de problemas de orden y coherencia de cache. Como resultado, ni el hardware mas moderno soporta esto adecuadamente.

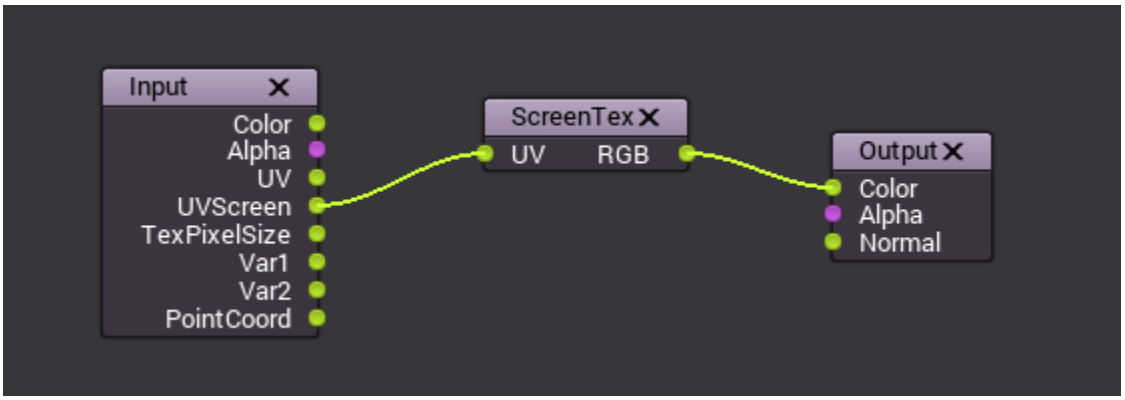
La solución es hacer una copia de la pantalla, o de una parte de la pantalla, a un back-buffer y luego leer desde el mientras se dibuja. Godot provee algunas herramientas para hacer este proceso fácil!

Instrucción shader TexScreen

El lenguaje *Shading language* tiene una instrucción especial, "texscreen", toma como parámetros el UV de la pantalla y retorna un vec3 RGB con el color. Una variable especial incorporada: SCREEN_UV puede ser usada para obtener el UV del fragmento actual. Como resultado, este simple shader de fragmento 2D:

```
COLOR=vec4( texscreen(SCREEN_UV), 1.0 );
```

termina en un objeto invisible, porque solo muestra lo que esta atrás. El mismo shader usando el editor visual luce así:



Ejemplo TexScreen

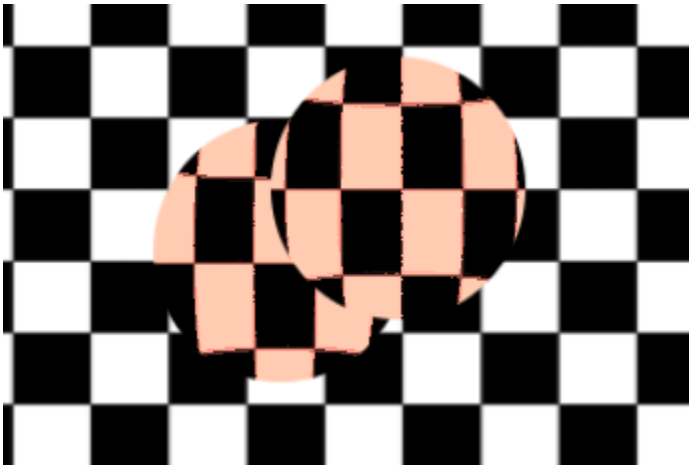
La instrucción TexScreen puede ser usada para muchas cosas. Hay una demo especial para *Screen Space Shaders*, que puedes descargar para verla y aprender. Un ejemplo es un shader simple para ajustar brillo, contraste y saturación:

```
uniform float brightness = 1.0;
uniform float contrast = 1.0;
uniform float saturation = 1.0;

vec3 c = texscreen(SCREEN_UV);

c.rgb = mix(vec3(0.0), c.rgb, brightness);
c.rgb = mix(vec3(0.5), c.rgb, contrast);
c.rgb = mix(vec3(dot(vec3(1.0), c.rgb)*0.33333), c.rgb, saturation);

COLOR.rgb = c;
```

Detrás de escena

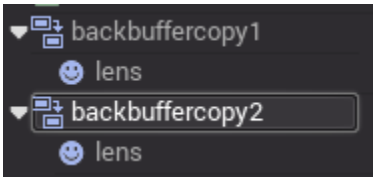
Mientras esto puede parecer mágico, no lo es. La instrucción Texscreen, cuando se encuentra por primera vez en un nodo que esta por ser dibujado, hace una copia de pantalla completa al back-buffer. Los nodos siguientes que usen texscreen() dentro de shaders no tendrán la pantalla copiada para ellos, porque esto seria muy ineficiente.

Como resultado, si se superponen shaders que usan texscreen(), el segundo no usara el resultado del primero, resultando en imágenes no esperadas:

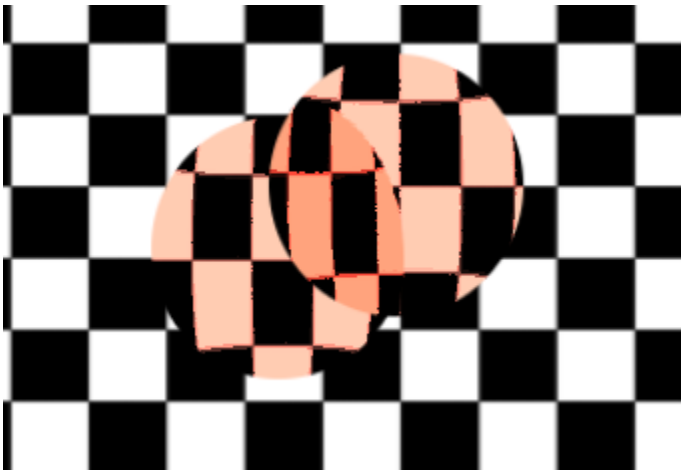


En la imagen de arriba, la segunda esfera (arriba derecha) esta usando el mismo origen para texscreen() que la primera debajo, por lo que la primera "desaparece", o no es visible.

Para corregir esto, un nodo *BackBufferCopy* puede ser instanciado entre ambas esferas. BackBufferCopy puede funcionar tanto especificando una región de pantalla o la pantalla entera:



Copiando adecuadamente el back-buffer, las dos esferas se mezclan correctamente:



Lógica de Back-buffer

Entonces, para dejarlo claro, aquí esta como la Lógica de copia de backbuffer funciona en Godot:

- Si un nodo usa texscreen(), la pantalla entera es copiada al back buffer antes de dibujar ese nodo. Esto solo sucede la primera vez, los siguientes nodos no lo dispararan.
- Si un nodo BackBufferCopy fue procesado antes de la situación en el punto de arriba (aun si texscreen() no ha sido usado), el comportamiento descrito en el punto de arriba no sucede. En otras palabras, la copia automática de la pantalla entera solo sucede si texscreen() es usado en un nodo por primera vez y ningún nodo BackBufferCopy (no deshabilitado) fue encontrado en el orden del árbol.
- BackBufferCopy puede copiar tanto la pantalla entera o una región. Si se ajusta solo a una región (no la pantalla entera) y tu shader usa pixels que no están en la región copiada, el resultado de esa lectura no esta definido (lo mas probable que sea basura de frames previos). En otras palabras, es posible usar BackBufferCopy para copiar de regreso una región de la pantalla y luego usar texscreen() en una región diferente. Evita este comportamiento!

3.1.5 Sistema de Partículas (2D)

Intro

Un sistema simple (aunque suficientemente flexible para la mayoría de los usos) de partículas se provee. Los sistemas de partícula son usados para simular efectos fisicos complejos como chispas, fuego, partículas de magia, humo, niebla, magia, etc.

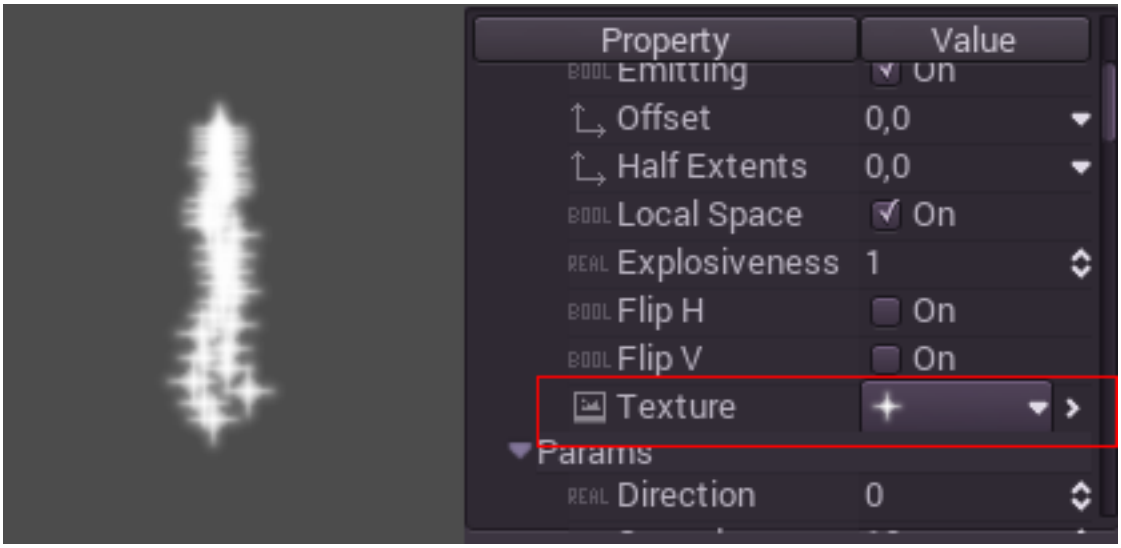
La idea es que la "partícula" es emitida en un intervalo fijo y con un tiempo de vida fijo. Durante su vida, cada partícula tendrá el mismo comportamiento base. Lo que hace diferente a cada partícula y provee un aspecto mas orgánico es la "aleatoriedad" asociada a cada parámetro. En esencia, crear un sistema de partículas significa ajustar parámetros de física base y luego agregarles aleatoriedad.

Particles2D

Los sistema de partículas son agregados a la escena con el nodo *Particles2D*. Están habilitadas por defecto y empiezan a emitir puntos blancos hacia abajo (como si estuviesen afectados por la gravedad). Esto provee un punto de arranque razonable para empezar a adaptarse a nuestras necesidades.

Textura

Un sistema de partícula usa una sola textura (en el futuro esto puede ser ampliado a texturas animadas por spritesheet). La textura se ajusta con la propiedad de textura:



Variables de Física

Antes de dar un vistazo a los parámetros globales del sistema de partículas, vamos primero a ver que pasa cuando las variables de física son modificadas.

Direction (dirección)

Este es el ángulo base en el que las partículas emiten. Por defecto es 0 (abajo):

Modificarlo cambiara la dirección del emisor, pero la gravedad seguirá afectándolas:

Este parámetro es útil porque, al rotar el nodo, la gravedad también será rotada. Cambiar dirección los mantiene separados.

Spread (propagación)

Spread es el ángulo en el que las partículas aleatorias serán emitidas. Aumentar el spread aumentara el ángulo. Un spread de 180 emitirá en todas direcciones.

Linear velocity (velocidad lineal)

Linear velocity es la velocidad a la que las partículas serán emitidas (en pixels por segundo). La velocidad puede ser luego modificada por la gravedad u otras aceleraciones (como se describe mas abajo).

Spin velocity (velocidad de giro)

Spin velocity es la velocidad a la cual las partículas giran al rededor de su centro (en grados por segundo).

Orbit velocity (velocidad orbital)

Velocidad orbital es usado para hacer que las partículas giren al rededor del centro.

Gravity direction & strength (dirección de gravedad y fuerza)

Gravity puede ser modificado como dirección y fuerza. La gravedad afecta capa partícula que esta viva.

Radial acceleration (aceleración radial)

Si esta aceleración es positiva, las partículas son aceleradas hacia fuera del centro. Si son negativas, son absorbidas hacia el centro.

Tangential acceleration (Aceleración tangencial)

Esta aceleración usara el vector tangente al centro. Combinándola con aceleración radial se pueden hacer lindos efectos.

Damping

Damping aplica fricción a las partículas, forzándolas a parar. Es especialmente útil para chispas o explosiones, las cuales usualmente empiezan con una alta velocidad linear y luego se detienen en la medida que se apagan.

Initial angle (Angulo inicial)

Determina el ángulo inicial de la partícula (en grados). Este parámetro es mas que nada útil cuando se usa de forma aleatoria.

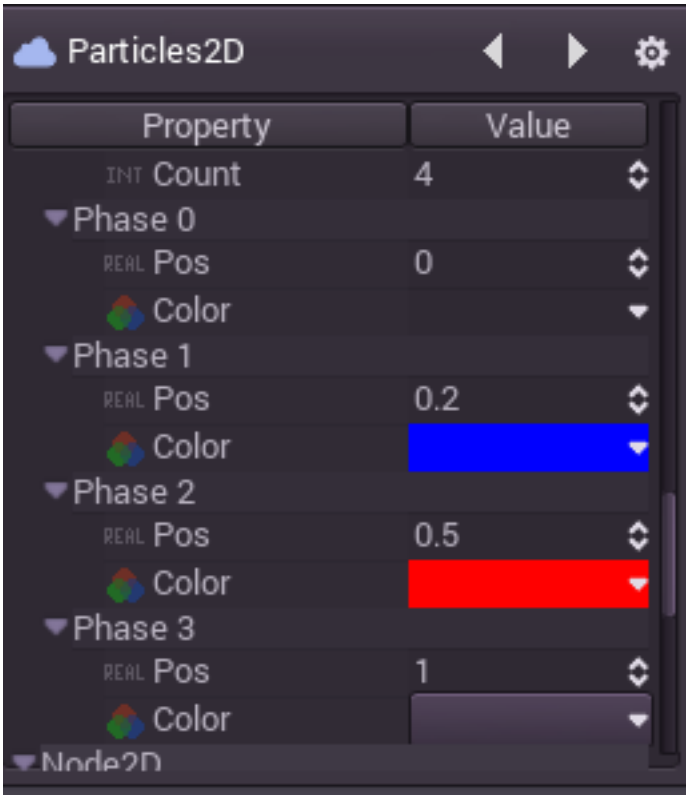
Initial & final size (Fase inicial y final)

Determina las escalas inicial y final de la partícula.

Color phases (Fases de color)

Las partículas pueden usar hasta 4 fases de color. Cada fase de color puede incluir transparencia.

Las fases deben proveer un valor offset del 0 a 1, y siempre en orden ascendente. Por ejemplo, un color va a empezar con offset 0 y terminar con offset 1, pero 4 colores pueden usar diferentes offsets, como 0, 0.2, 0.8 y 1.0 para las diferentes fases:



Resultara en:

Global parameters (Parámetros globales)

Estos parámetros afectan el comportamiento del sistema entero.

Lifetime (Tiempo de vida)

El tiempo en segundos que cada partícula estará viva. Cuando lifetime llega a su fin, una nueva partícula es creada para reemplazarla.

Lifetime: 0.5

Lifetime: 4.0

Timescale (Escala de tiempo)

Sucede a menudo que el efecto que se alcanza es perfecto, excepto que es muy rápido o muy lento. Timescale ayuda ajustando la velocidad en su conjunto.

Timescale everything 2x:

Preprocess (Pre procesamiento)

Los sistemas de partícula empiezan con 0 partículas emitidas, luego empiezan a emitir. Esto puede ser un inconveniente cuando recién cargas una escena y sistemas como antorchas, nieble, etc empiezan a emitir en el momento que entras. Preprocess es usado para dejar al sistema procesar una cantidad dada de segundos antes de que se muestre por primera vez.

Emit timeout (Tiempo limite de emisión)

Esta variable va a apagar la emision luego estar encendida una cantidad dada de segundos. Si es cero, estará deshabilitada.

Offset

Permite mover el centro del emisor fuera del centro.

Half extents

Hace el centro (por defecto 1 pixel) mas ancho, hasta el valor en pixels deseado. Las partículas serán emitidas de forma aleatoria dentro de esta area.

También es posible ajustar una mascara de emisión usando este valor. Chequea el menu “Particles” en el viewport del editor de escena 2D y selecciona tu textura favorita. Los pixels opacos serán usados como potenciales lugares de emisión, mientras que los transparentes serán ignorados.

Local space

Por defecto esta opción esta habilitada, y significa que el espacio hacia el cual son emitidas las partículas esta contenido en el nodo. Si el nodo es movido, todas las partículas se mueven con el:

Si se deshabilita, las partículas se emitirán a espacio global, lo que implica que si el nodo es movido, el emisor se mueve también:

Explosiveness (Explosividad)

Si lifetime es 1 y hay 10 partículas, significa que cada partícula será emitida cada .1 segundos. El parámetro explosiveness cambia esto, y fuerza que las partículas sean emitidas todas juntas. Los rangos son:

- 0: Emite todas las partículas juntas.
- 1: Emite las partículas en intervalos iguales.

Los valores entre medio también son permitidos. Esta característica es útil para crear explosiones o ráfagas repentinas de partículas:

Randomness (Aleatoriedad)

Todos los parámetros físicos pueden ser aleatorizados. Las variables aleatorias van de 0 a 1. La formula para volver aleatorio un parámetro es:

```
initial_value = param_value + param_value*randomness
```

3.1.6 Animación Cutout

Que es?

Cut-out es una técnica de animación en 2D donde las piezas de papel (u otro material similar) son cortadas en formas especiales y acomodadas una sobre la otra. Los papeles son animados y fotografiados, frame por frame usando técnica stop-motion (mas info aqui).

Con el advenimiento de la era digital, esta técnica se hizo posible usando computadores, lo que resulto en un incremento de la cantidad de shows de TV animados usando Cut-out digital. Ejemplos notables son South Park or Jake and the Never Land Pirates.

En juegos de video, esta técnica también se volvió muy popular. Ejemplos de esto son Paper Mario o Rayman Origins .

Cutout en Godot

Godot provee algunas herramientas para trabajar con este tipo de assets, pero su diseño en conjunto lo hace ideal para el workflow. La razón es que, a diferencia de otras herramientas hechas para esto, Godot tiene las siguiente ventajas:

- ** El sistema de animación esta completamente integrado con el motor**: Esto significa, las animaciones pueden controlar mucho mas que solo el movimiento de los objetos, como texturas, tamaño de sprites, pivots, opacidad, modulación de color, etc. Todo puede ser animado y mezclado.
- **Mezcla con tradicional**: AnimatedSprite permite mezclar animación tradicional, muy util para objetos complejos, como la forma de las manos y pies, cambiar la expresión facial, etc.
- **Elementos con Formas Personalizadas**: Pueden ser creados con *Polygon2D* permitiendo la mezcla de animación UV, deformaciones, etc.
- **Sistema de partículas**: Puede ser mezclado con la jerarquía de animación tradicional, útil para efectos de magia, jetpacks, etc.
- **Colisionadores personalizados**: Ajusta los colisionadores e influencia áreas en diferentes partes del esqueleto, excelente para jefes, juegos de pelea, etc.

- **Árbol de animación**: Permite combinaciones complejas y mezclas de varias animaciones, de la misma forma que funciona en 3D.

Y mucho mas!

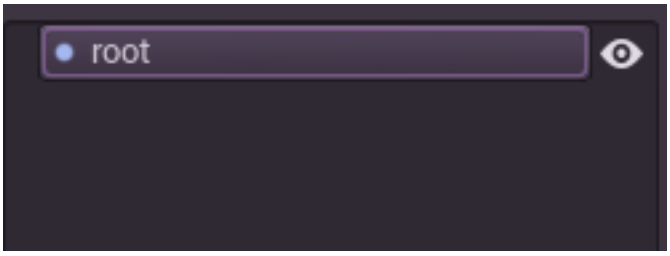
Haciendo el GBot!

Para este tutorial, vamos a usar como contenido del demo las piezas de el personaje *GBot <https://www.youtube.com/watch?v=S13FrWuBMx4&list=UUckpus81gNin1aV8WSffRKw>*, creado por Andreas Esau.

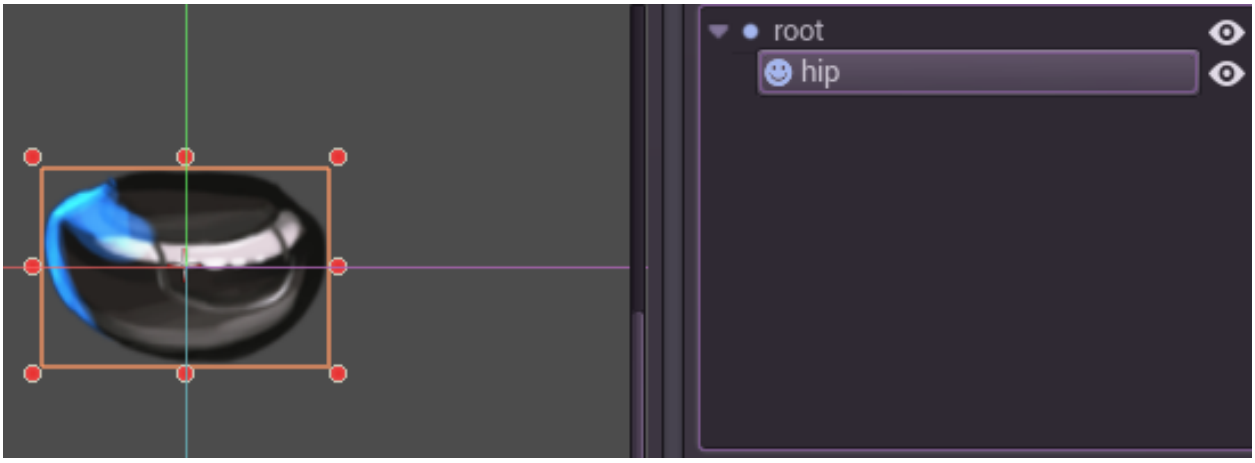
Obtén tus assets: `gbot_resources.zip`.

Construyendo el rig

Crea un Node2D vacío como raíz de la escena, trabajaremos bajo el:



OK, el primer nodo del modelo que vamos a crear será la cadera(hip). Generalmente, tanto en 2D como 3D, la cadera es la raíz del esqueleto. Esto hace mas sencillo animarlo:

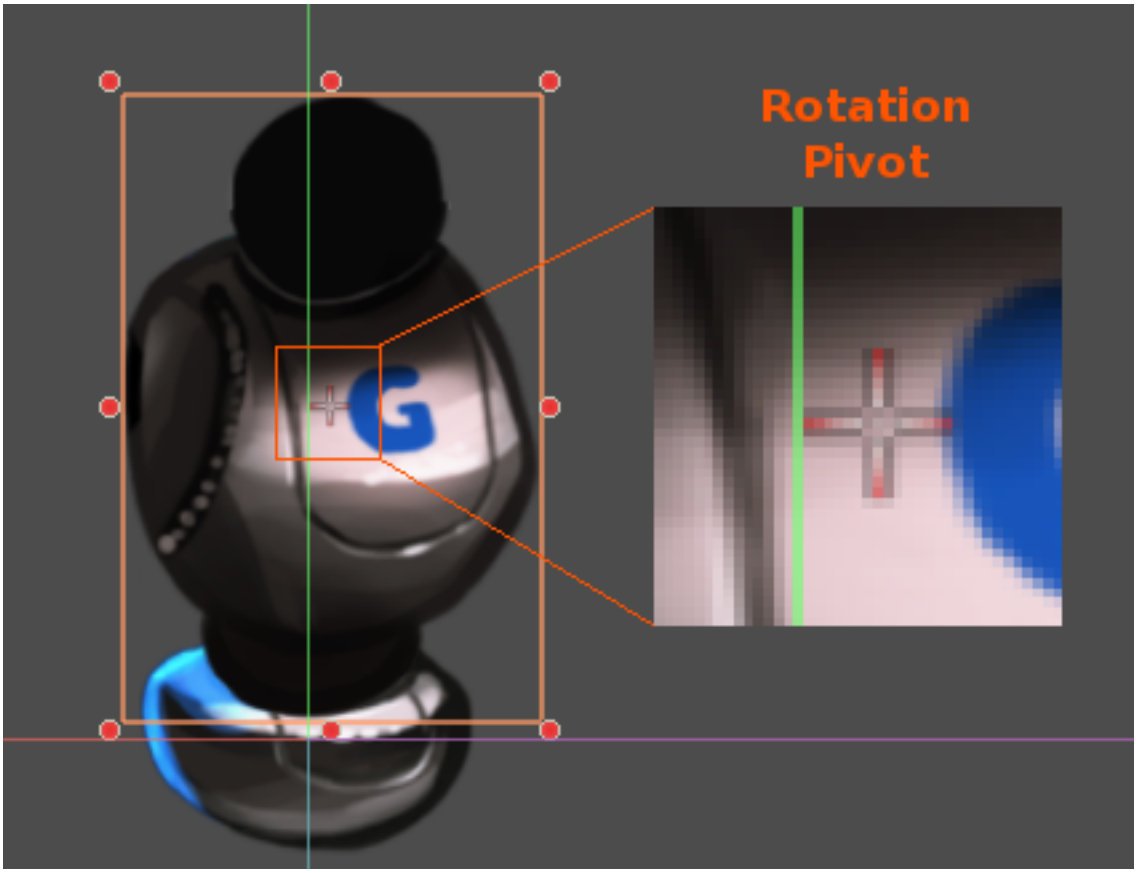


Luego será el torso. El torso necesita ser un hijo de la cadera, así que crea un sprite hijo y carga el torso, luego acomódalo adecuadamente:

Esto luce bien. Veamos si nuestra jerarquía trabaja como un esqueleto al rotar el torso:

Ouch, no luce bien! El pivot de rotación esta mal, esto implica que debe ser ajustado.

Esta pequeña cruz en el medio de el *Sprite* es el pivot de rotación:

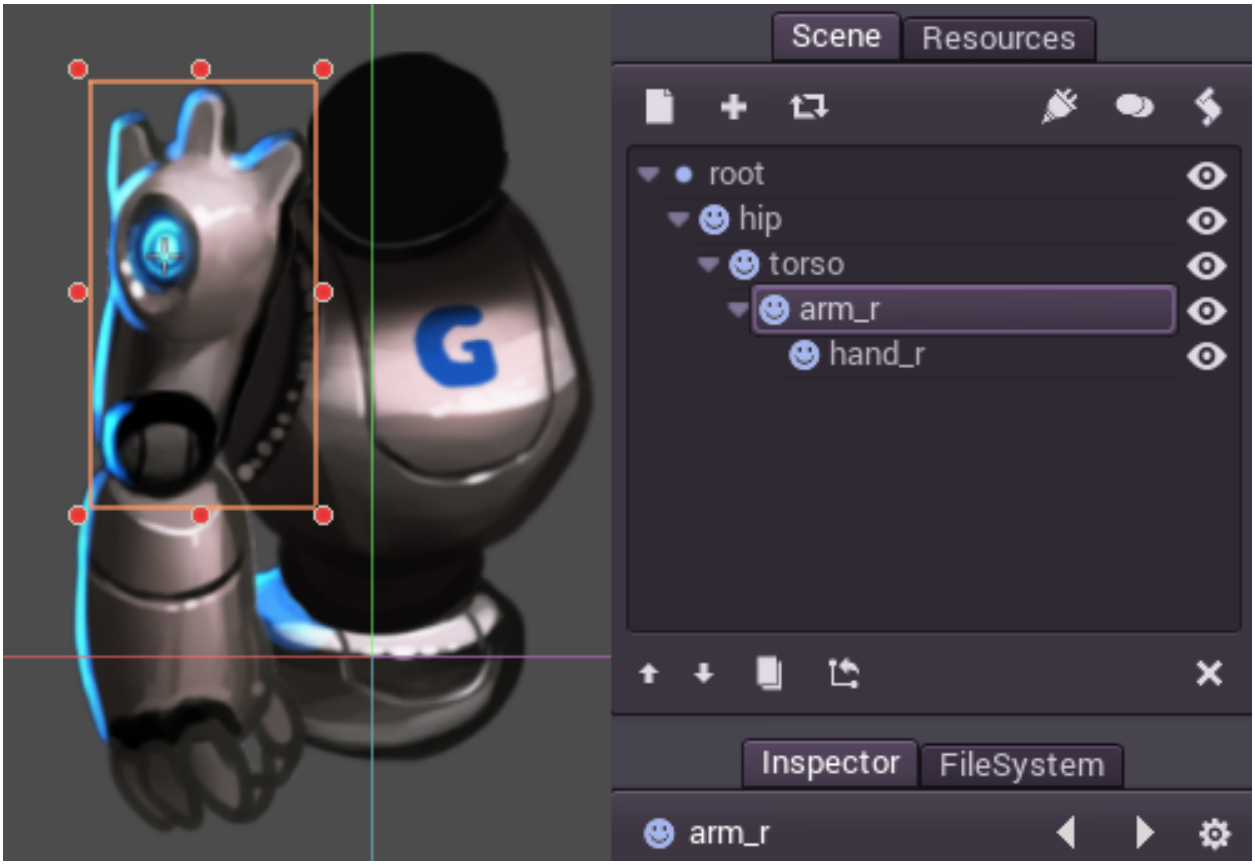


Ajustando el pivot

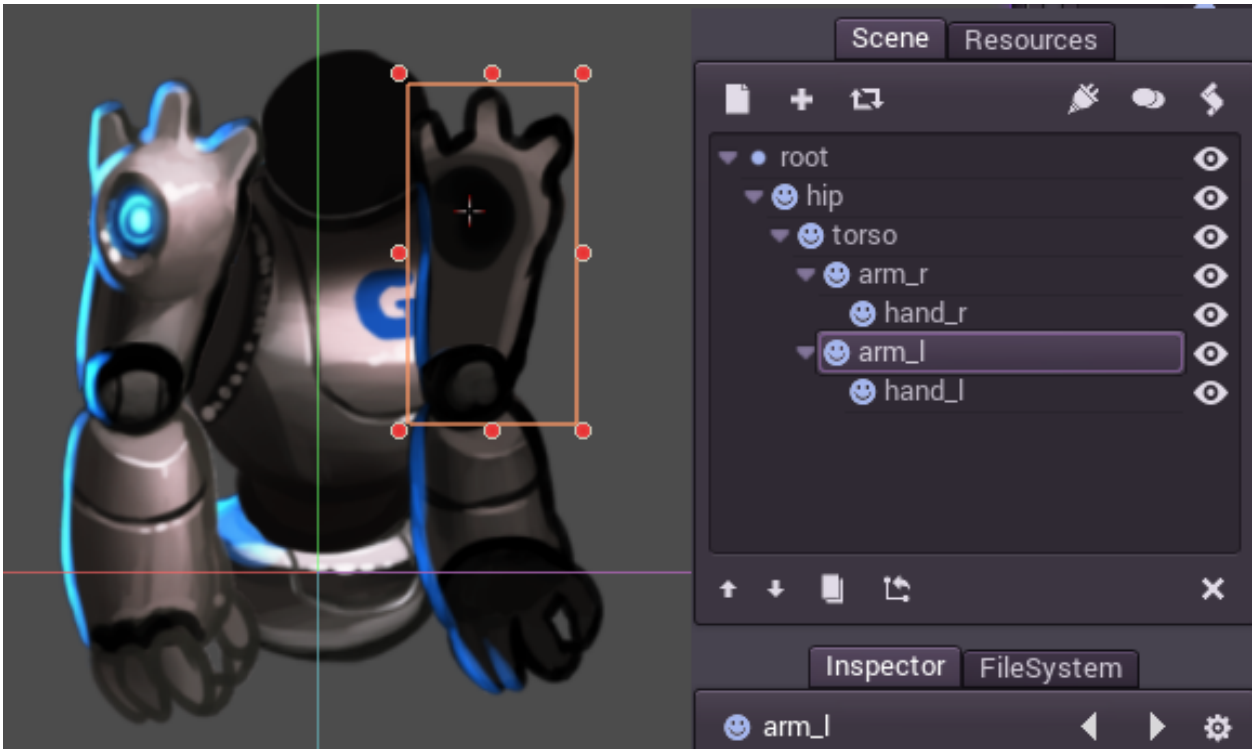
El pivot puede ser ajustado al cambiar la propiedad *offset* en el Sprite:

Sin embargo, hay una forma de hacerlo mas visualmente. Mientras flotas sobre el punto deseado como pivot, simplemente presiona la tecla "v" para mover el pivot allí para el sprite seleccionado. Alternativamente, hay una herramienta en la barra de herramientas que tiene una función similar.

Ahora luce bien! Continuemos agregando piezas del cuerpo, empezando por el brazo derecho. Asegúrate de poner los sprites en jerarquía, así sus rotaciones y traslaciones son relativas al padre:



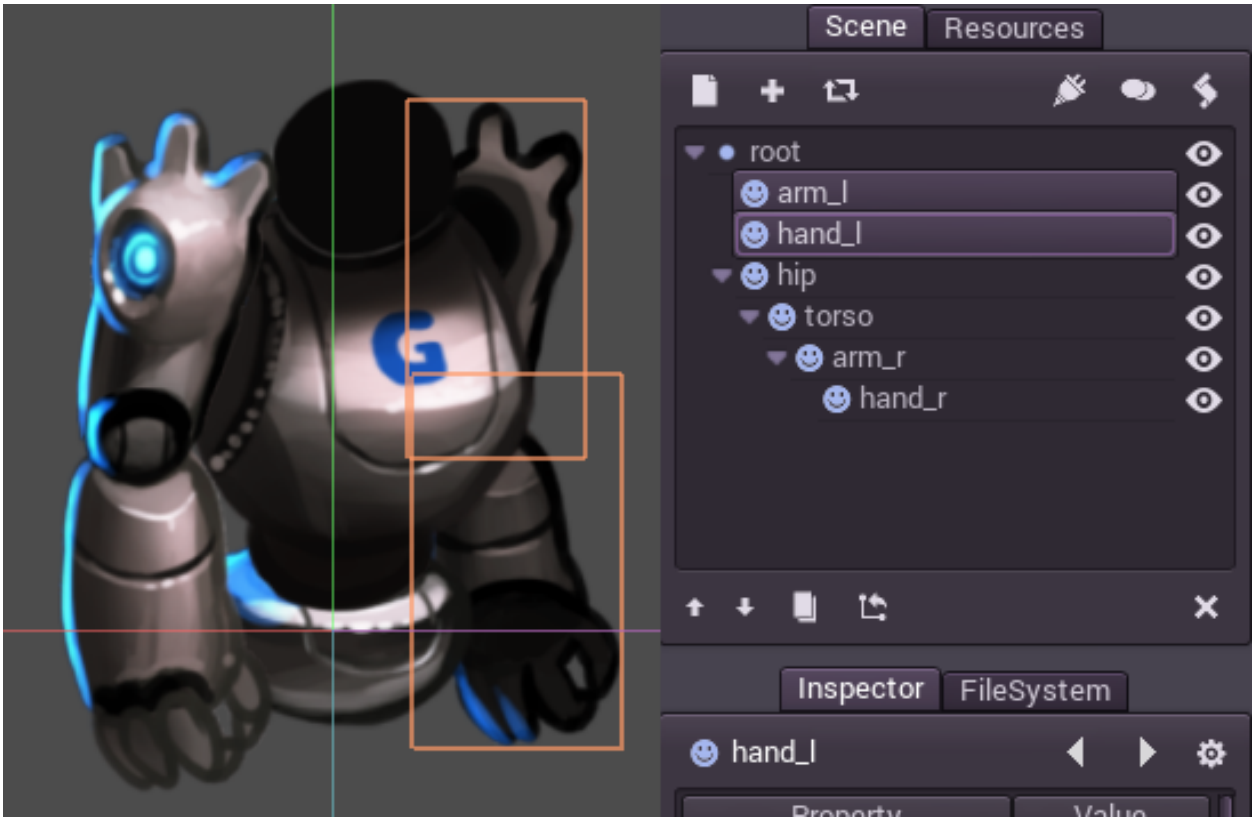
Esto parece fácil, asique continua con el brazo derecho. El resto debería ser simple! O tal vez no:



Bien. Recuerda tus tutoriales, Luke. En 2D, los nodos padre aparecen debajo de los nodos hijos. Bueno, esto apesta.

Parece que Godot no soporta rigs cutout después de todo. Vuelve el año próximo, tal vez para la 3.0.. no espera. Solo bromeaba! Funciona bien.

Pero como puede ser resuelto este problema? Queremos que el brazo izquierdo aparezca detrás de la cadera y el torso. Para esto, podemos mover los nodos detrás de la cadera (ten en cuenta que puedes eludir este paso al ajustar la propiedad Z del Node2D, pero entonces no aprenderías todo esto!):



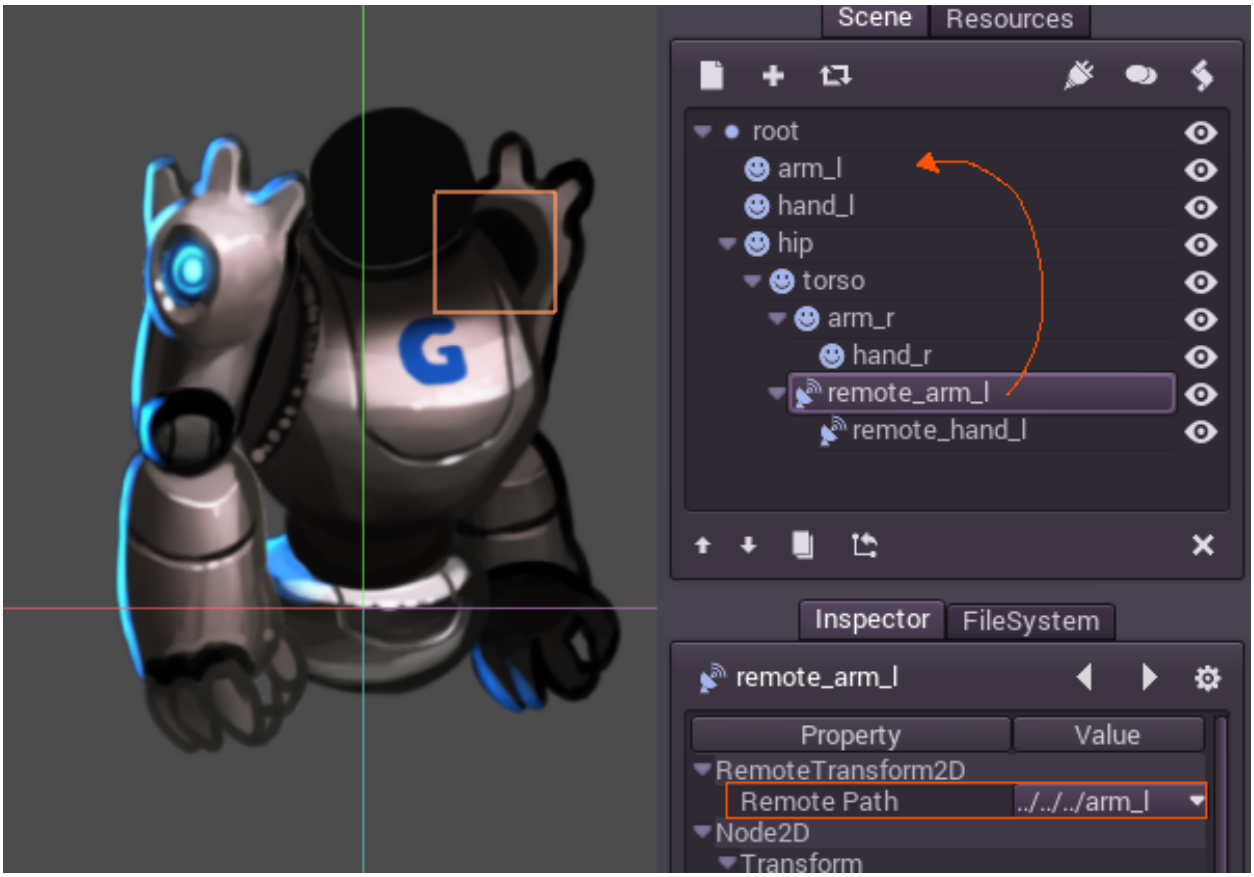
Pero entonces, perdemos el orden de la jerarquia, lo que nos permite controlar el esqueleto como.. un esqueleto. Hay alguna esperanza?.. Por supuesto!

Nodo RemoteTransform2D

Godot provee un nodo especial, *RemoteTransform2D*. Este nodo transformara nodos que están en algún otro lugar en la jerarquía, al aplicar la transformación en los nodos remotos.

Esto permite tener un orden de visibilidad independiente de la jerarquía.

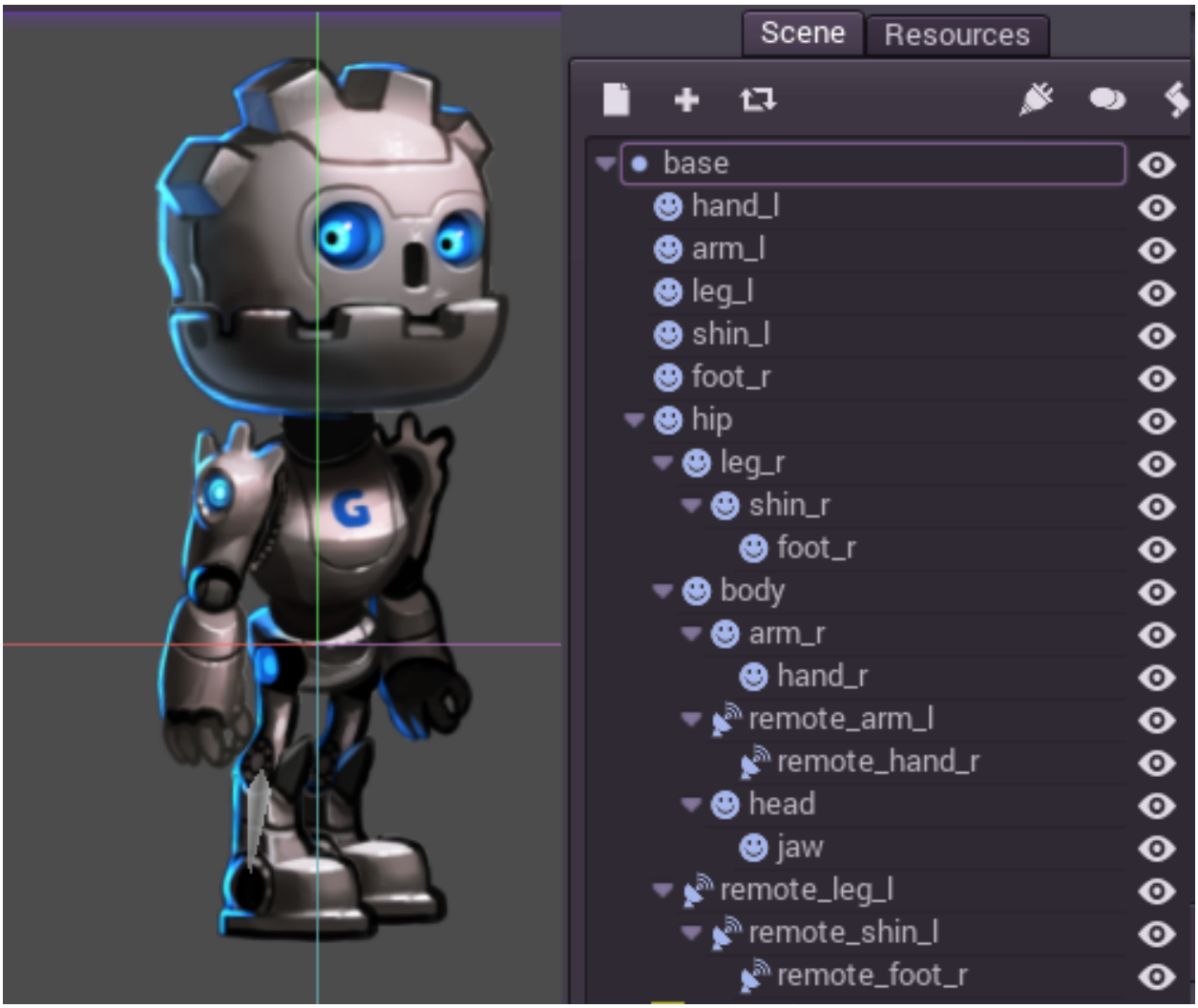
Simplemente crea dos nodos mas como hijos del torso, remote_arm_l y remote_hand_l y vincúlalos a los sprites:



Mover los nodos de transformación remota hará mover los sprites, para posar y animar fácilmente al personaje:

Completando el esqueleto

Completa el esqueleto siguiendo los mismos pasos para el resto de las partes. La escena resultante debería lucir similar a esto:



El rig resultante será fácil de animar. Al seleccionar los nodos y rotarlos puedes animarlo eficientemente usando forward kinematics (FK).

Para objetos y rigs simples esto esta bien, sin embargo los siguientes problemas son comunes:

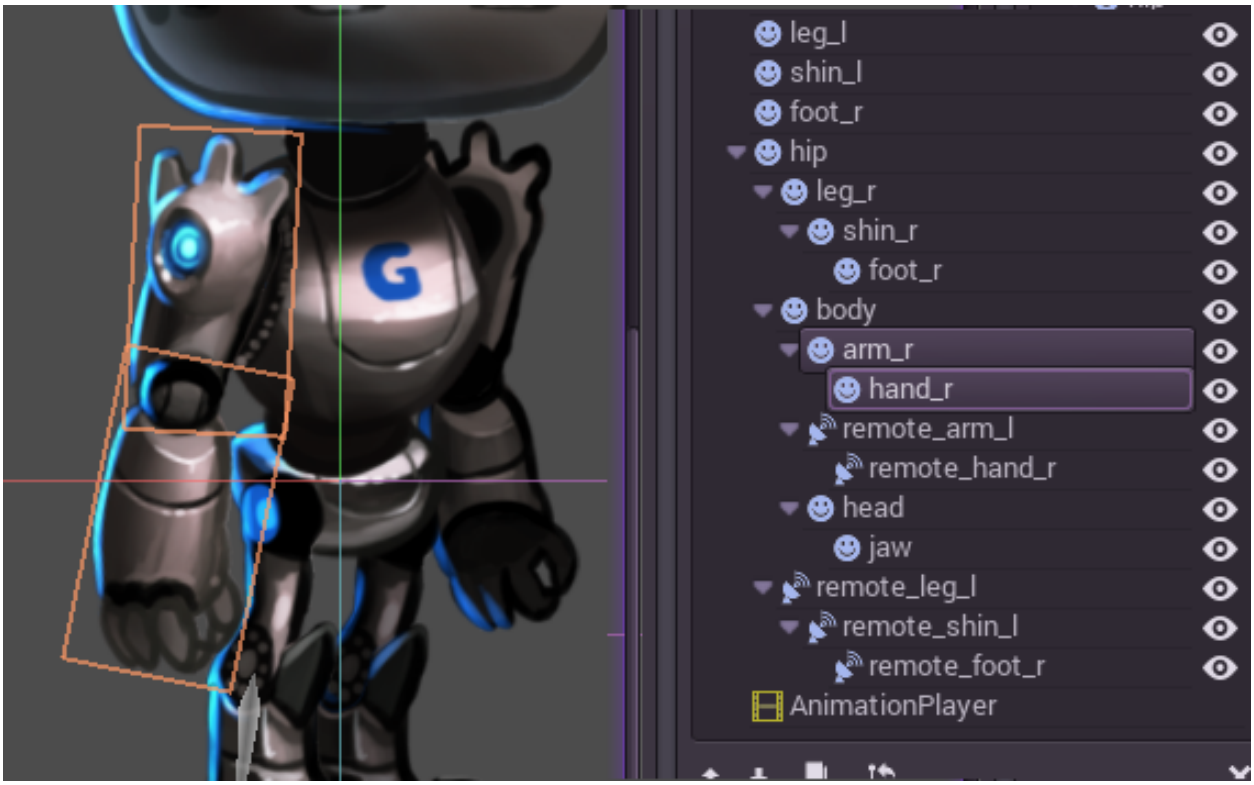
- Seleccionar sprites puede volverse difícil para rigs complejos, y el árbol de escena termina siendo usado debido a la dificultar de hacer clic sobre los sprites adecuados.
- A menudo es deseable usar Inverse Kinematics (IK) para las extremidades.

Para solucionar estos problemas, Godot soporta un método simple de esqueletos.

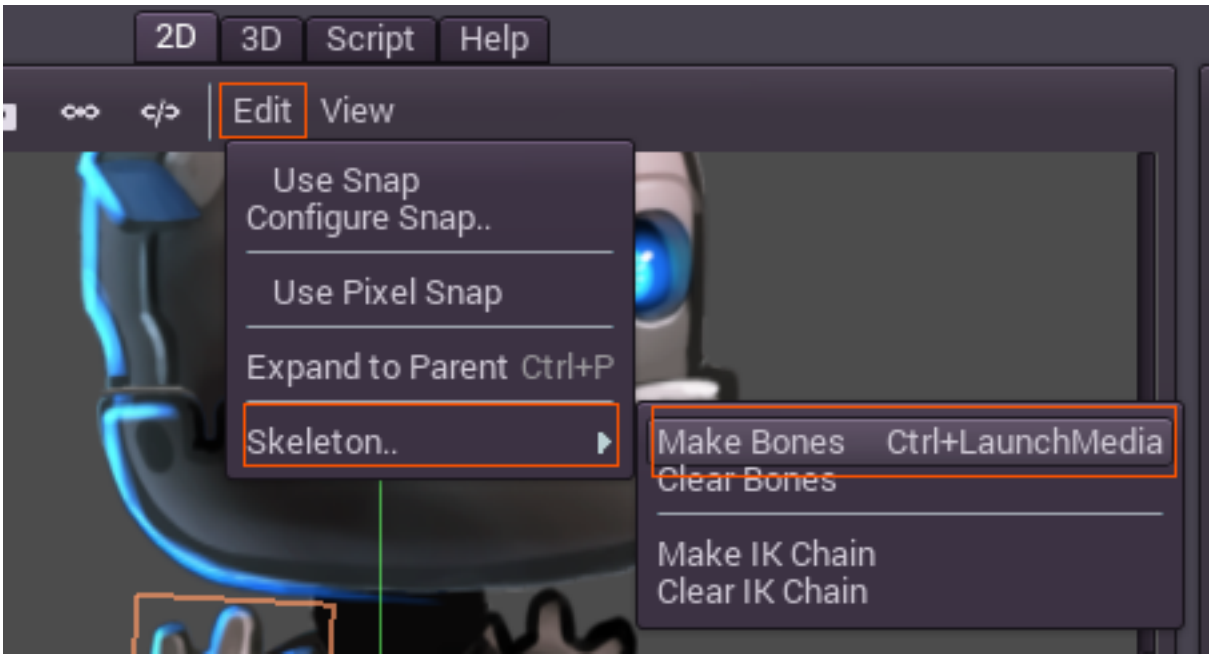
Esqueletos

Godot en realidad no soporta *verdaderos* esqueletos, pero contiene un ayudante (helper) para crear "huesos" entre nodos. Esto es suficiente para la mayoría de los casos, pero la forma como funciona no es completamente obvia.

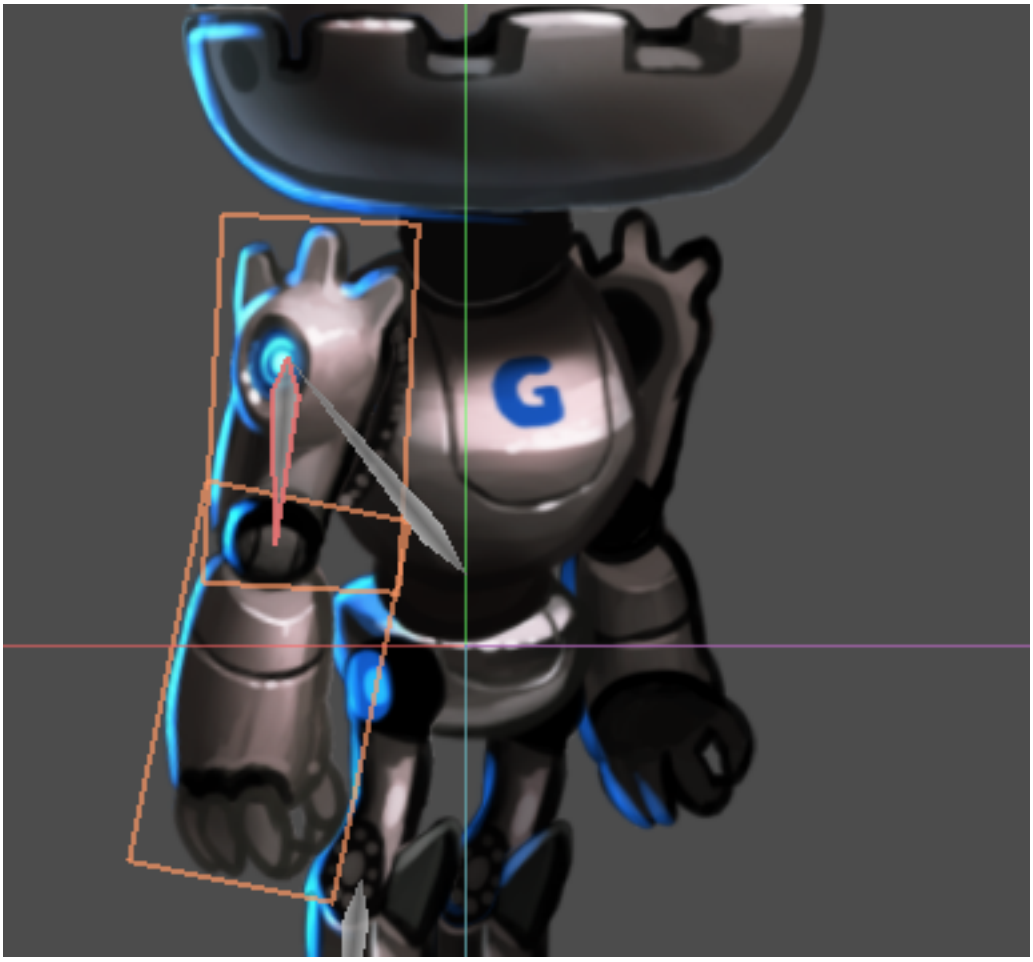
Como ejemplo, vamos a volver un esqueleto el brazo derecho. Para crear esqueletos, una cadena de nodos debe ser seleccionada desde la cima al fondo:



Luego, la opción para crear un esqueleto esta localizada en Editar > Esqueleto > Crear Huesos:



Esto agregara huesos cubriendo el brazo, pero el resultado no es lo que esperabas.



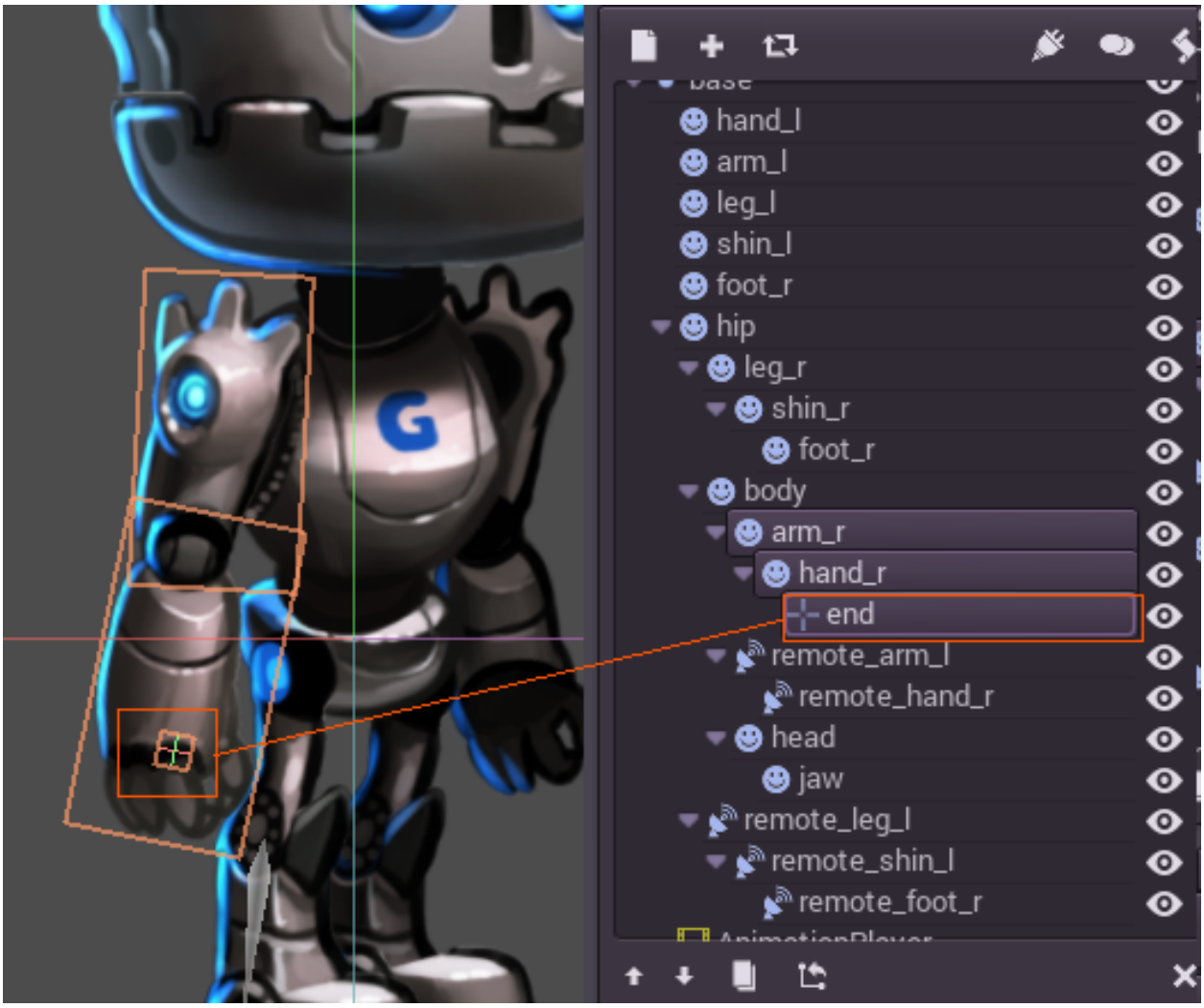
Parece que los huesos fueron desplazados hacia arriba en la jerarquía. La mano se conecta al brazo, y al brazo al cuerpo. Entonces la pregunta es:

- Porque la mano carece de un hueso?
- Porque el brazo se conecta al cuerpo?

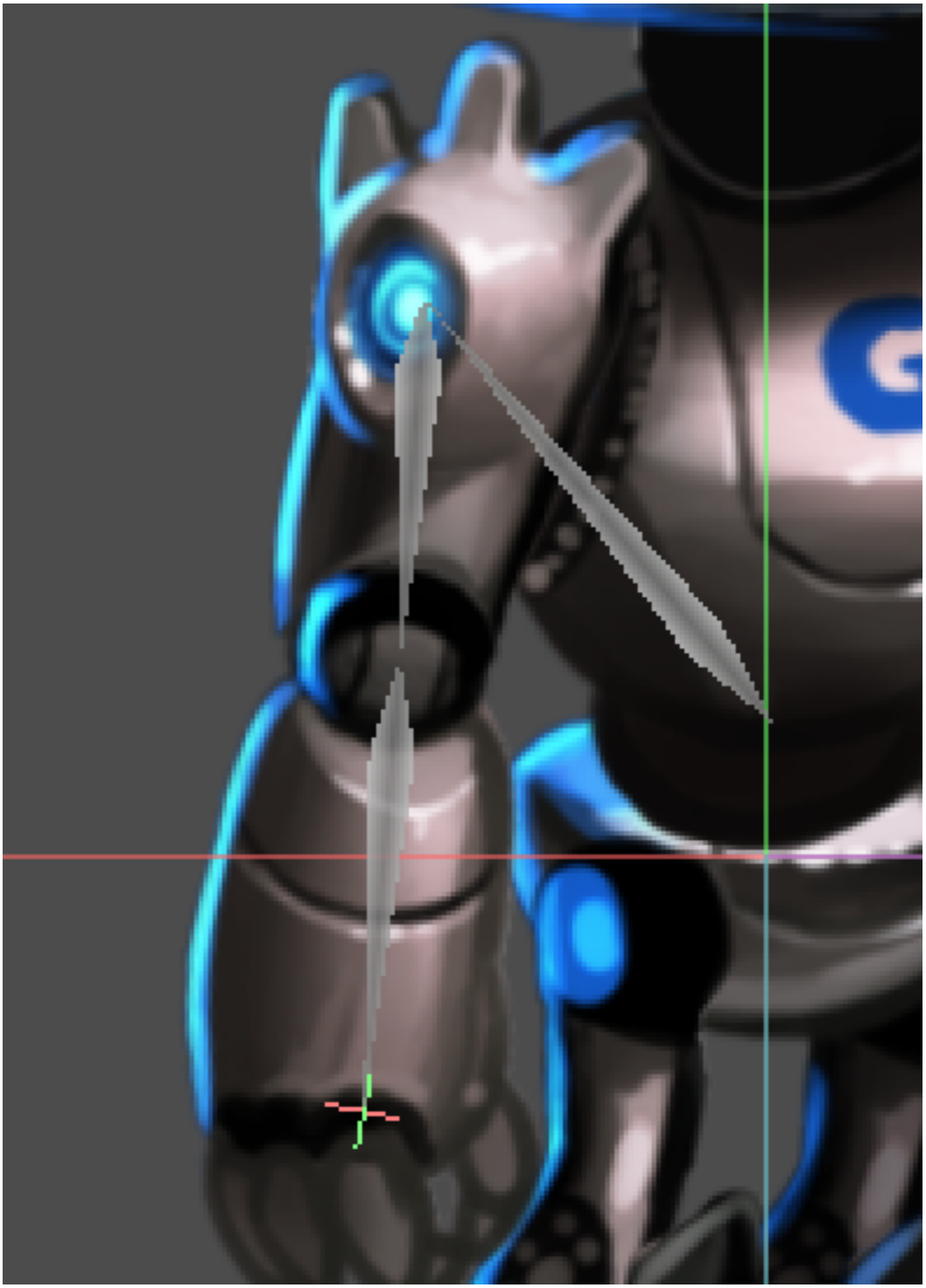
Esto puede ser extraño al comienzo, pero tendrá sentido mas tarde. En sistemas tradicionales de esqueletos, los huesos tienen una posición, una orientación y un largo. En Godot, los huesos son mas que nada ayudantes por lo que conectan al nodo actual con el padre. Por esto, **alternar un nodo como un hueso solo lo conectara con el padre**.

Así que, con este conocimiento. Hagamos lo mismo nuevamente así tenemos un esqueleto útil.

El primer paso es crear no nodo endpoint (punto final). Cualquier tipo de nodo lo hará, pero :ref:‘Position2D <class_Position2D>‘es preferido porque será visible en el editor. El nodo endpoint asegurara que el ultimo nodo tenga orientación.

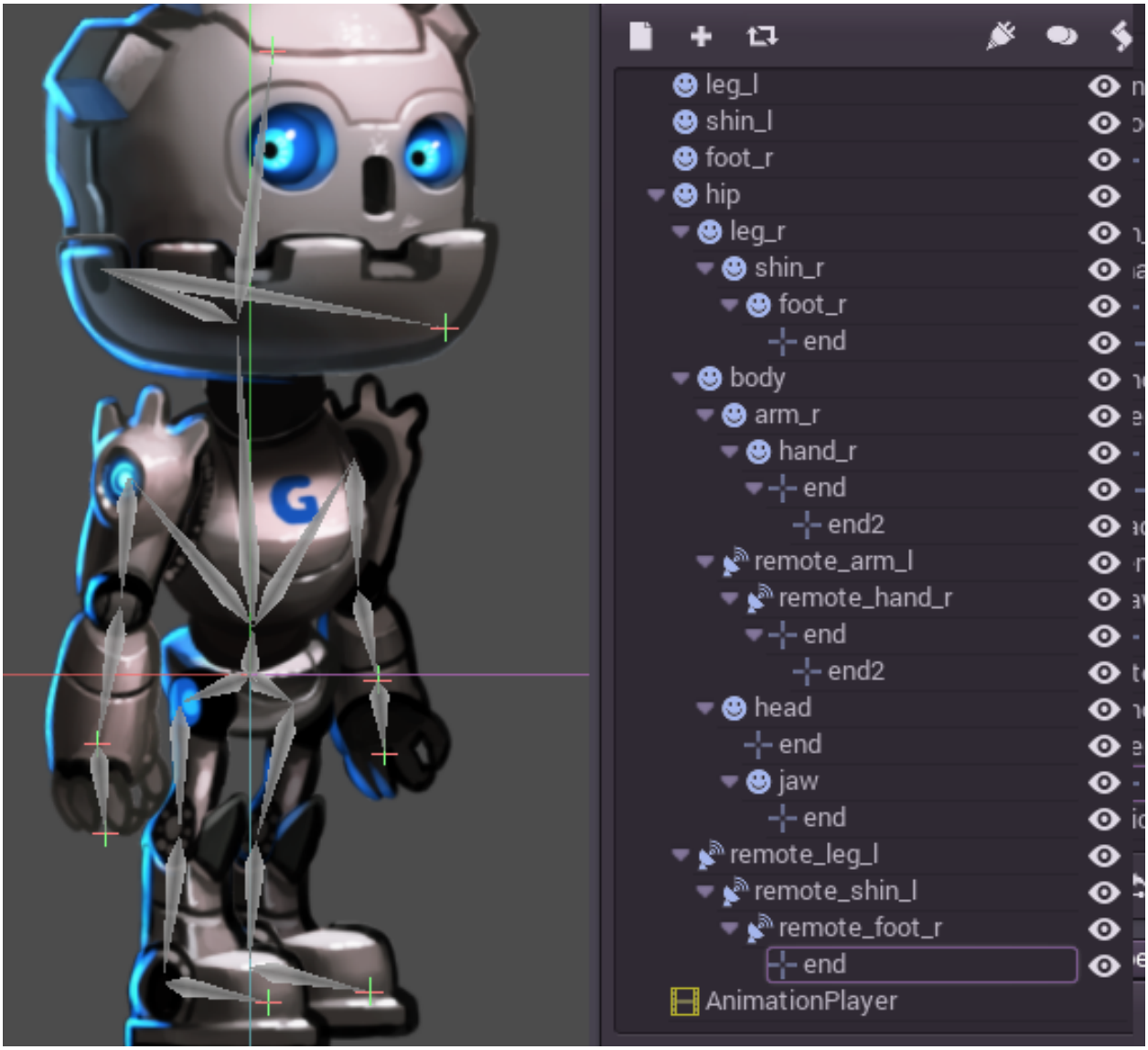


Ahora seleccionar la cadena completa, desde el endpoint hasta el brazo para crear huesos



El resultado se parece mucho mas a un esqueleto, y ahora el brazo y el antebrazo puede ser seleccionado y animado.

Finalmente, crea endpoints en todas las extremidades adecuadas y conecta el esqueleto completo con huesos hasta la cadera:



Al fin! El esqueleto completo fue rigged! Mirando de cerca, se puede ver que hay un segundo conjunto de endpoints en las manos. Esto tendrá sentido pronto.

Ahora que el esqueleto completo fue rigged, el próximo paso es ajustar IK chains. IK chains permiten un control mas natural de las extremidades.

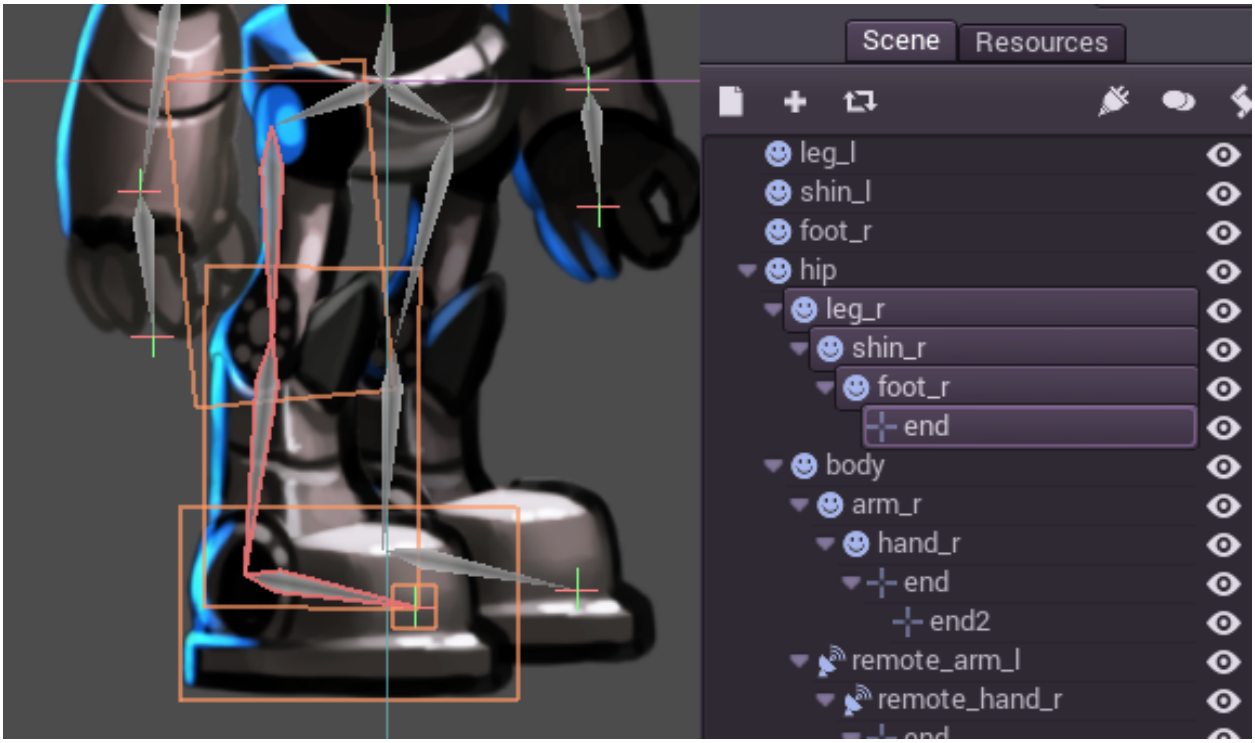
IK chains (cadenas IK)

IK chains es una poderosa herramienta de animación. Imagina que quieres posar el pie de un personaje en una posición especifica en el piso. Sin IK chains, cada movimiento del pise requerirá rotar y posicionar varios huesos mas. Esto seria bastante complejo y nos llevaría a resultados imprecisos.

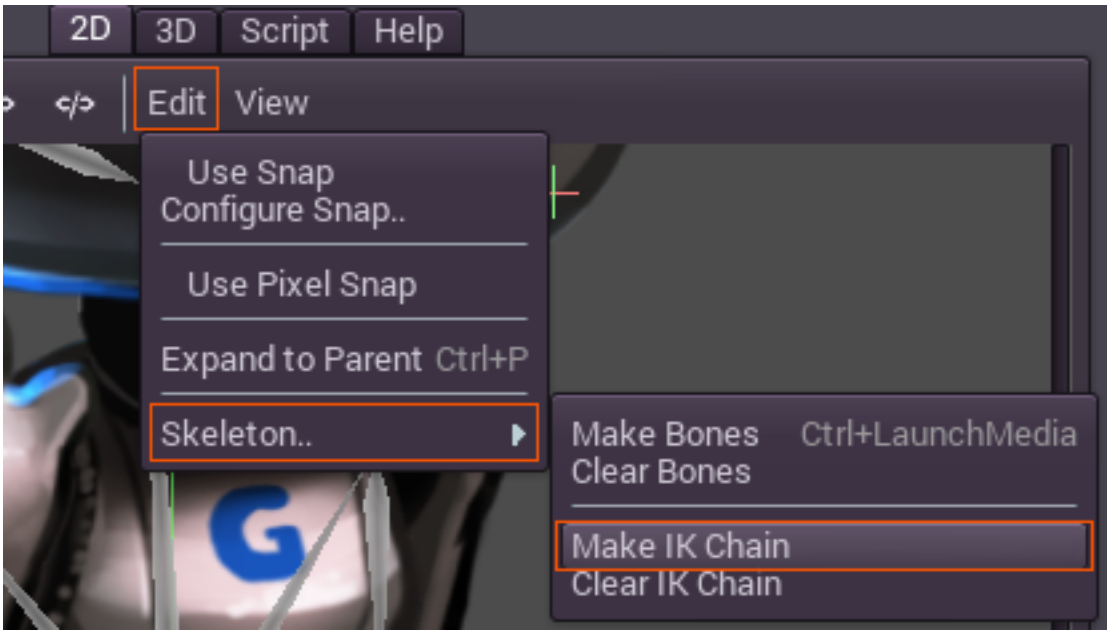
Y si pudiéramos mover el pie y dejar que el resto de la pierna se ajuste sola?

Este tipo de pose se llama IK (Inverse Kinematic - Cinematica Inversa).

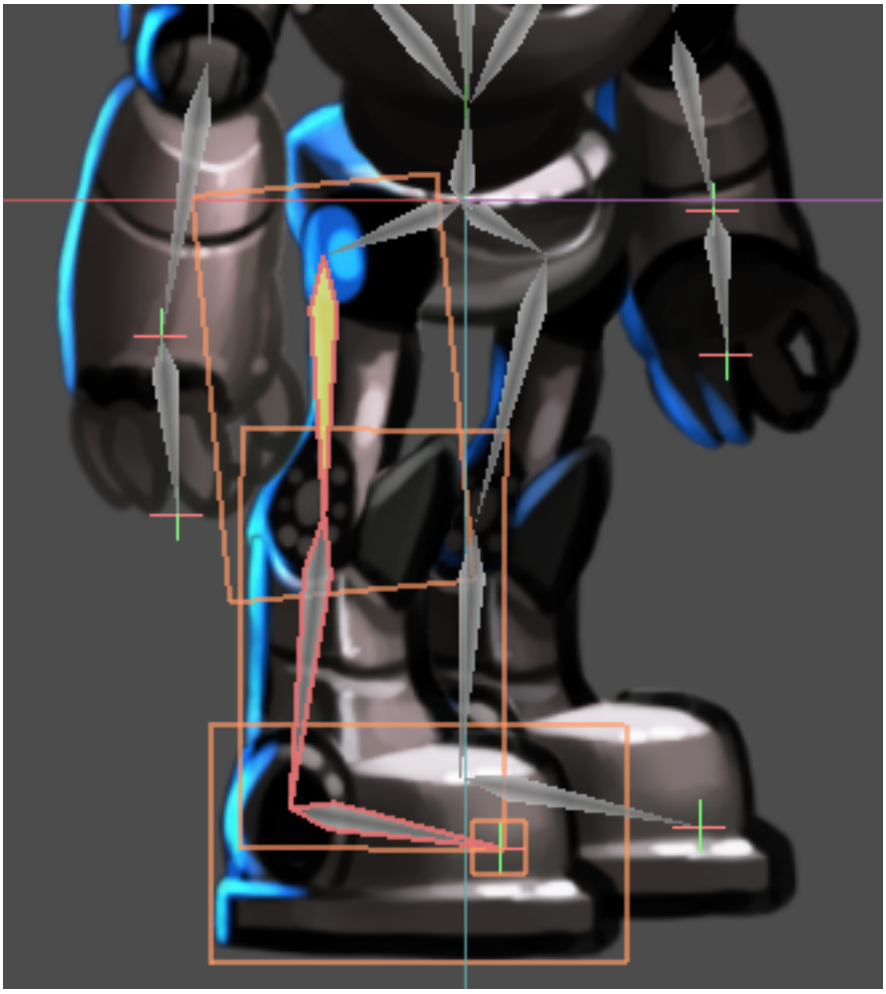
Para crear una cadena IK, simplemente selecciona una cadena de huesos desde el endpoint hasta la base de la cadena. Por ejemplo, para crear una cadena IK para la pierna derecha, selecciona lo siguiente:



Para habilitar esta cadena para IK. Ve a Editar > Esqueleto > Crear Cadena IK.



Como resultado, la base de la cadena se volverá *Amarilla*



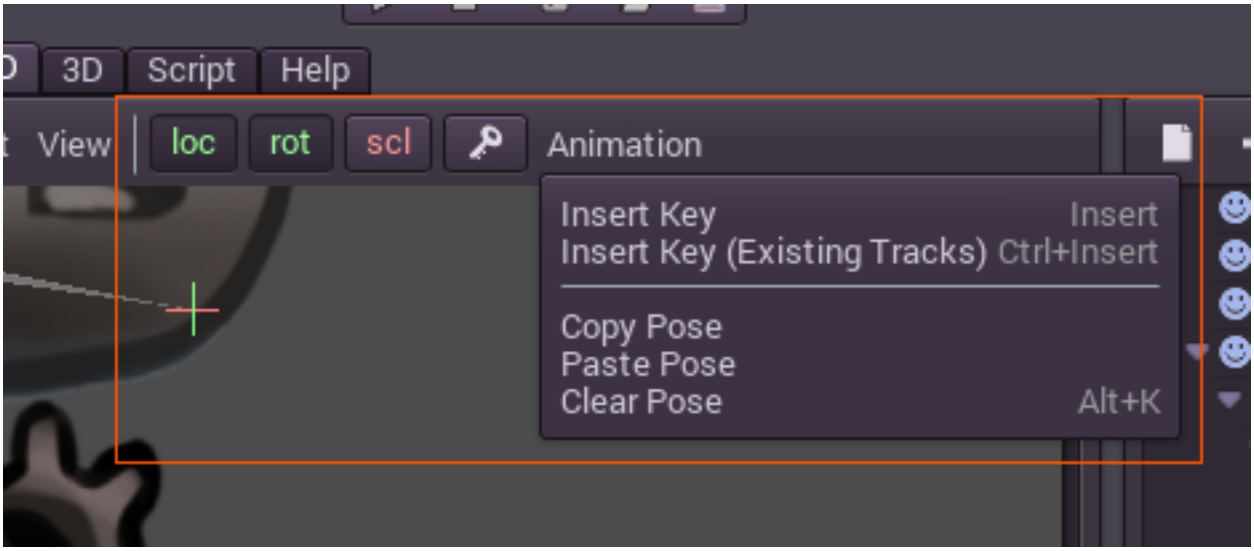
Una vez que la cadena IK ha sido configurada, simplemente toma cualquiera de los huesos en la extremidades, cualquier hijo o nieto de la base de la cadena y trata de moverlo. El resultado será placentero, satisfacción garantizada!

Animación

La siguiente sección será una colección de consejos para crear animaciones para tus rigs. Si no esta seguro sobre como funciona el sistema de animación en godot, refréscalo chequeando nuevamente *Animaciones*.

Animación 2D

Cuando hagas animación en 2D, un ayudante estará presente en el menú superior. Este ayudante solo aparece cuando la ventana del editor de animación este abierta:



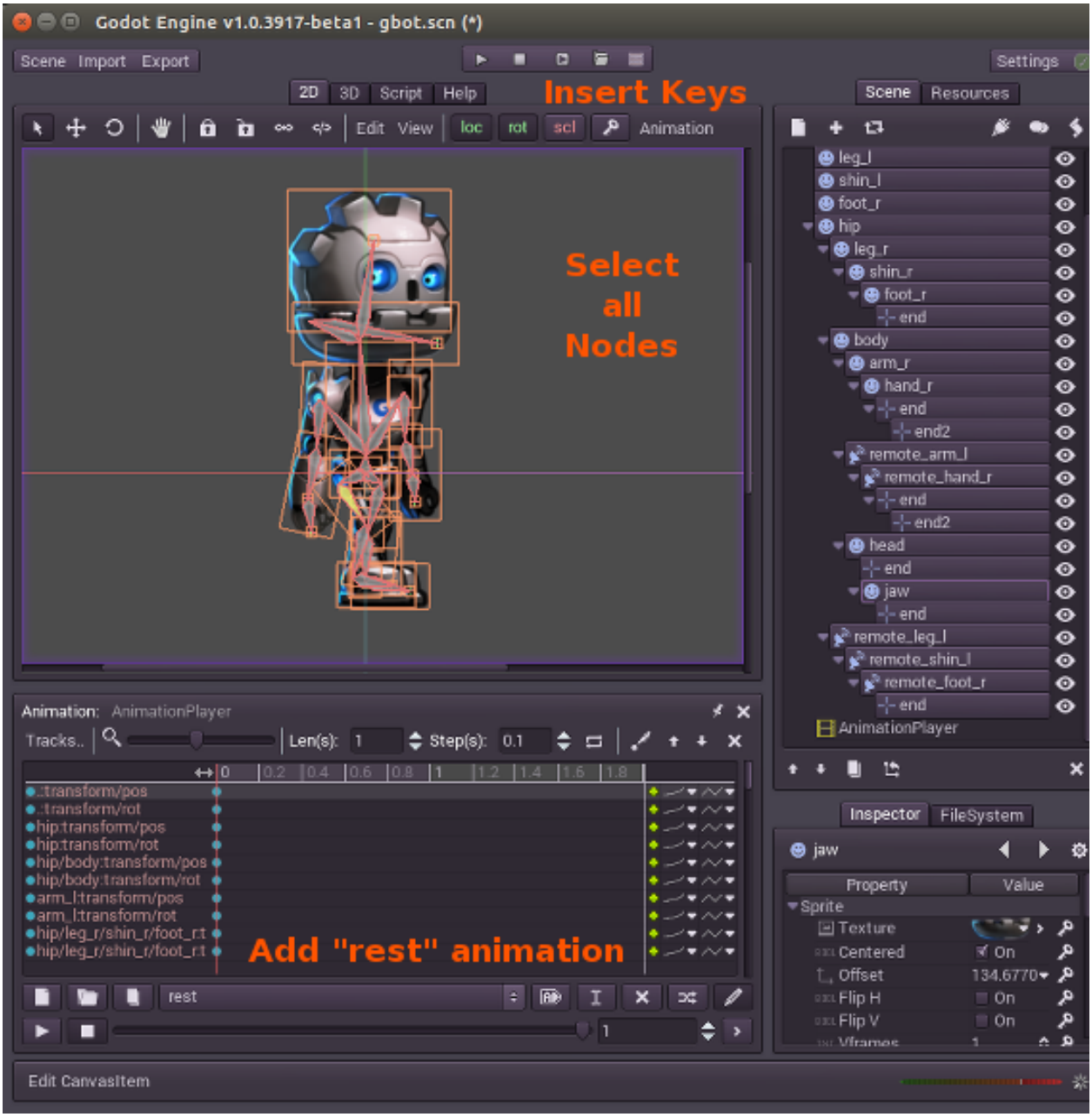
El botón de llave insertara keyframes de posicion(loc)/rotación(rot)/ escala(scl) a los objetos o huesos seleccionados. Esto depende en la mascara habilitada. Los ítems verdes insertaran claves mientras que los rojos no, asique modifica la mascara de inserción para tu preferencia.

Pose de descanso

Este tipo de rigs no tiene una pose de "descanso", así que es recomendado crear una pose de descanso de referencia en una de las animaciones.

Simplemente sigue los siguientes pasos.

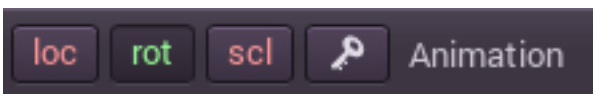
1. Asegúrate que el rig este en "descanso" (sin hacer ninguna pose especifica).
2. Crea una nueva animación, renómbrala a "descanso".
3. Selecciona todos los nodos (la selección de caja debería funcionar bien)
4. Selecciona "loc" y "rot" en el menú superior.
5. Presiona el botón de llave. Las llaves serán insertadas para todo, creando una pose por defecto.



Rotación

Animar estos modelos significa solo modificar la rotación de los nodos. Lugar y escala raramente son usados, con la única excepción de mover el rig entero desde la cadera (la cual es el nodo raíz).

Como resultado, cuando insertas claves, solo el botón “rot” necesita ser presionado la mayoría del tiempo:



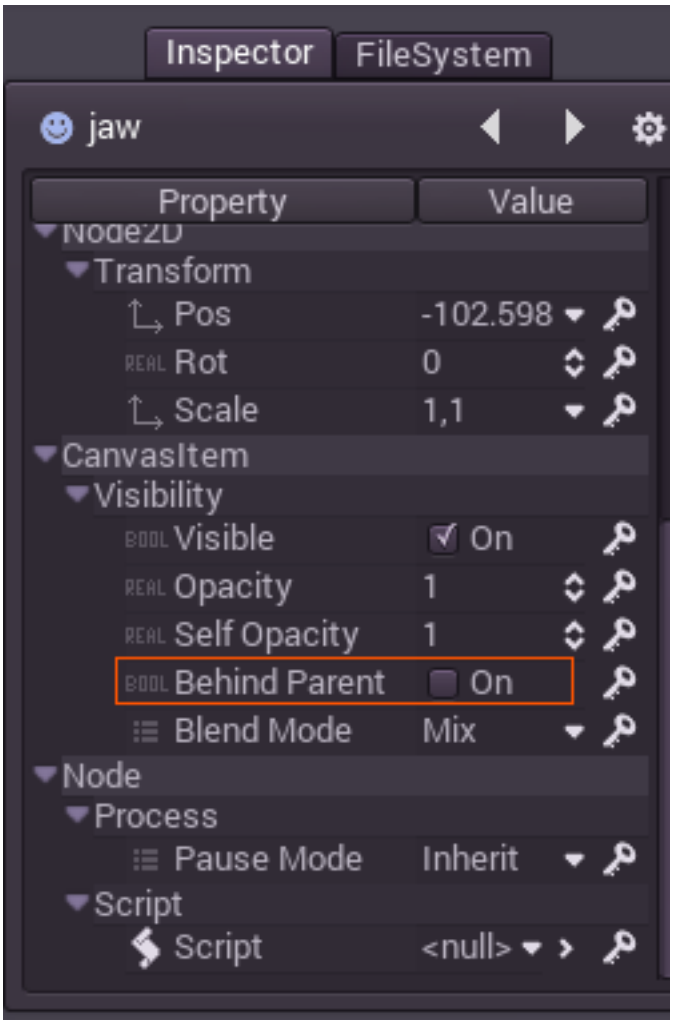
Esto evitara la creación de pistas extra de animación para la posición que no seran utilizadas.

Keyframing IK

Cuando se editan cadenas IK, no es necesario seleccionar la cadena entera para agregar keyframes. Seleccionar el endpoint de la cadena e insertar un keyframe también insertara automáticamente keyframes hasta la base de de la cadena. Esto hace la tarea de animar extremidades mucho mas simple.

Moviendo sprites por delante y detrás de otros.

RemoteTransform2D trabaja en la mayoría de los casos, pero a veces es realmente necesario tener un nodo encima y debajo de otros durante una animación. Para ayudar con esto existe la propiedad "Behind Parent" en cualquier Node2D:



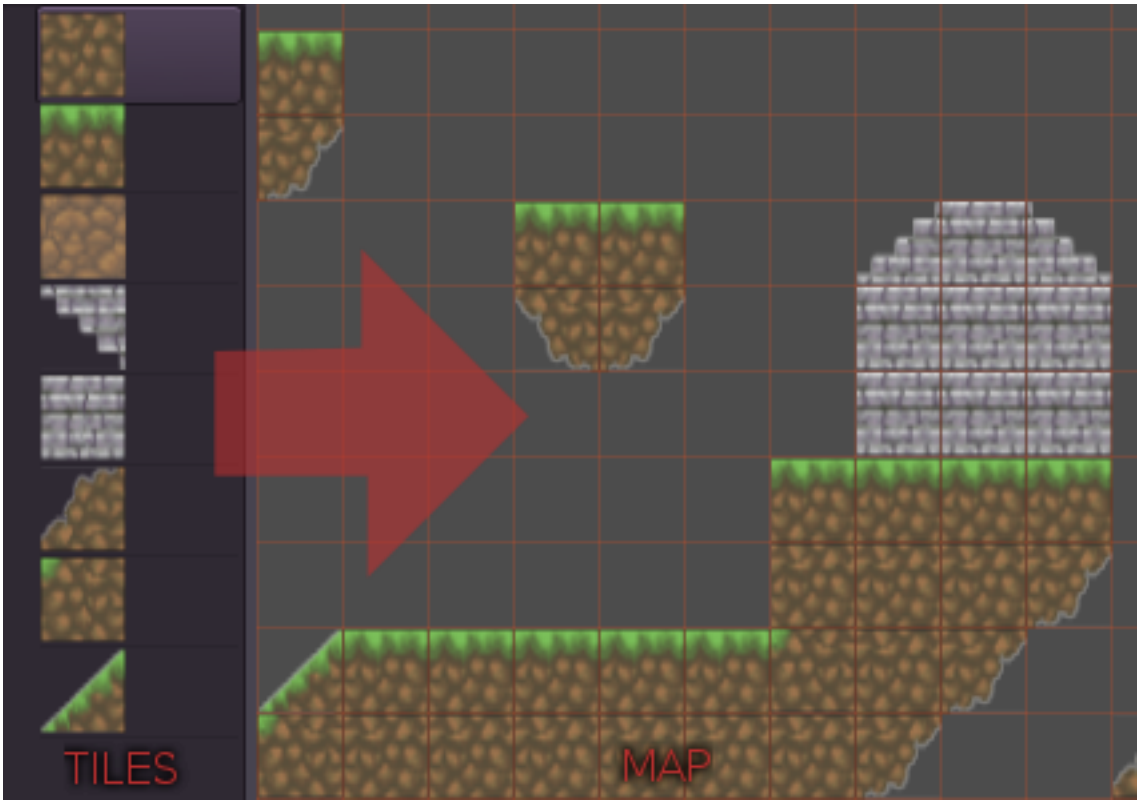
Ajustes de transición de curvas por lotes

Cuando se crean animaciones realmente complejas y se insertan muchos keyframes (cuadros clave), editar las curvas individuales de los keyframes puede volverse una tarea interminable. Para esto, el Editor de Animación tiene un pequeño menú donde es fácil cambiar todas las curvas. Solo selecciona cada uno de los keyframes y (generalmente) aplica la curva de transición "Out-In" para una animación suave:

3.1.7 Usando Tilemaps

Introducción

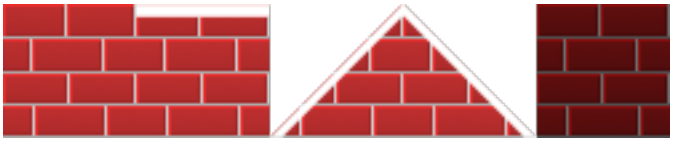
Los Tilemaps son una forma simple y rápida de hacer niveles para juegos 2D. Básicamente, empiezas con un montón de tiles (baldosas, piezas) de referencia que pueden ser puestas en una grilla, tantas veces como se desee:



Las colisiones también pueden ser agregadas a los tiles, permitiendo tanto juegos de desplazamiento lateral (side scrolling) o con vista desde arriba (top down).

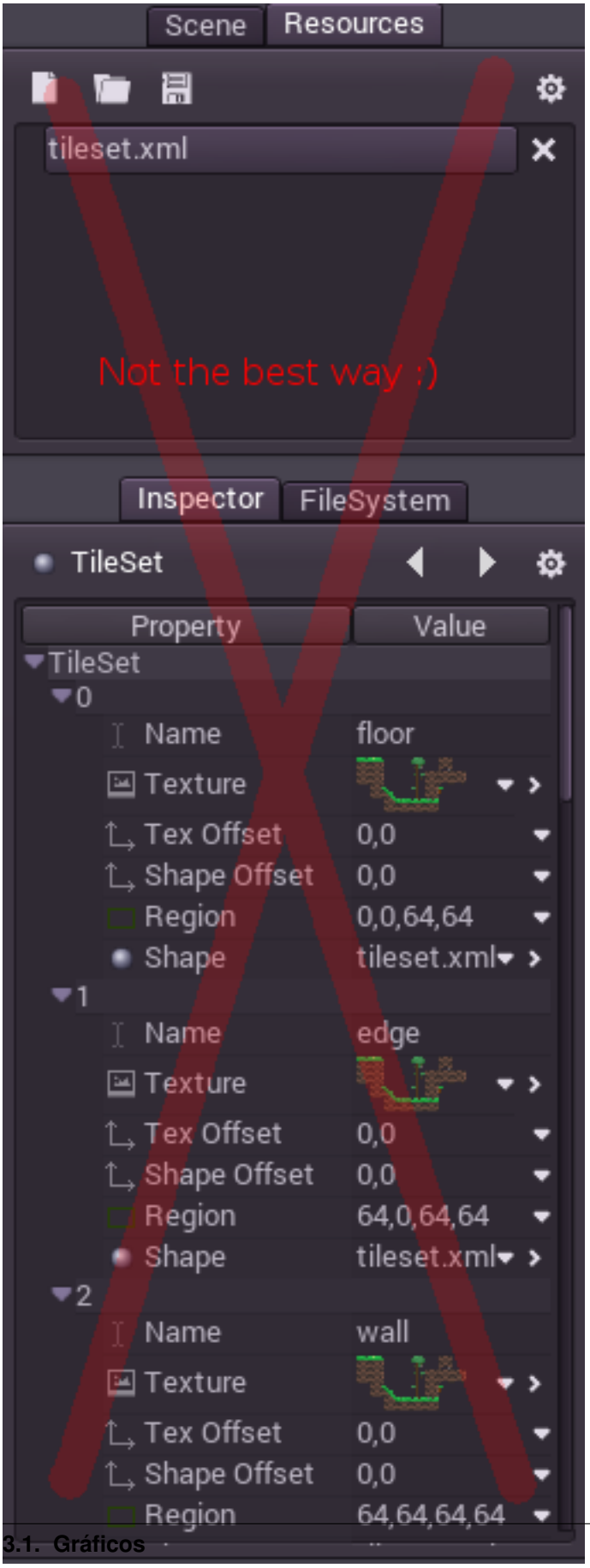
Haciendo un tileset

Para empezar, un tileset tiene que ser hecho. Aquí hay algunos tiles para ello. Están todos en la misma imagen porque los artistas a menudo preferien esto. Tenerlos como imágenes separadas también funciona.



Crea un nuevo proyecto y mueve la imagen png superior a su directorio.

Estaremos creando un recurso *TileSet*. Mientras que este recurso exporta propiedades, es bastante difícil ingresarle datos complejos y mantenerlo:


Scene
Resources
tileset.xml
Not the best way :)
Inspector
FileSystem
TileSet
Property
Value
TileSet
0
Name
floor
Texture
Tex Offset
0,0
Shape Offset
0,0
Region
0,0,64,64
Shape
tileset.xml
1
Name
edge
Texture
Tex Offset
0,0
Shape Offset
0,0
Region
64,0,64,64
Shape
tileset.xml
2
Name
wall
Texture
Tex Offset
0,0
Shape Offset
0,0
Region
64,64,64,64

Hay suficientes propiedades para arreglárselas, y con algo de esfuerzo editando de esta forma puede funcionar, pero la forma mas simple de editar y mantener un tileset es con la herramienta de exportar!

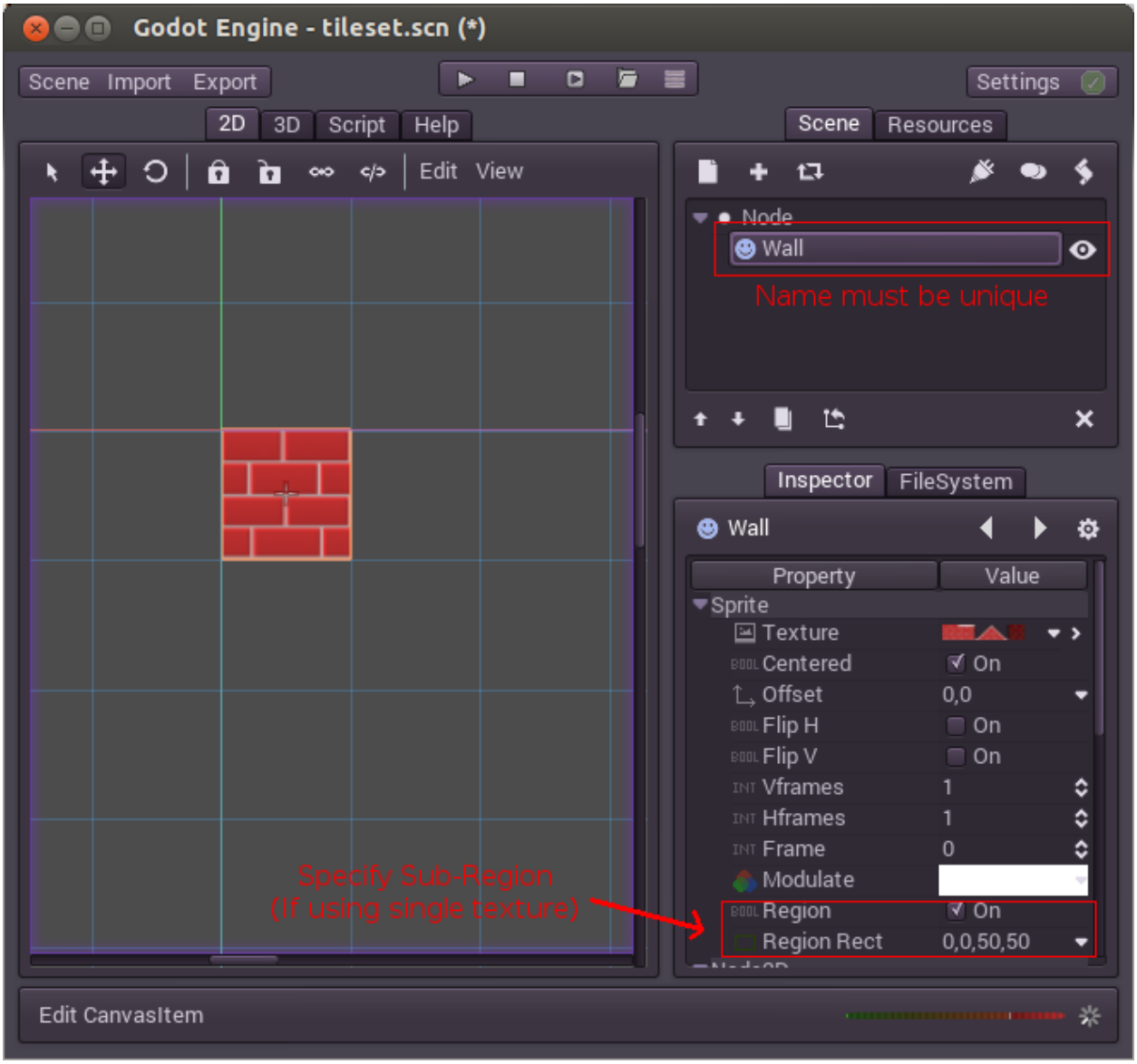
Escena TileSet

Crea una nueva escena con un nodo regular o Node2D como raíz. Para cada tile, agrega un sprite como hijo. Ya que los tiles aquí son de 50x50, habilitar snap puede ser una buena idea (Editar > Usar Snap, Mostrar Grilla y en Configurar Snap, Step Grilla 50 y 50).

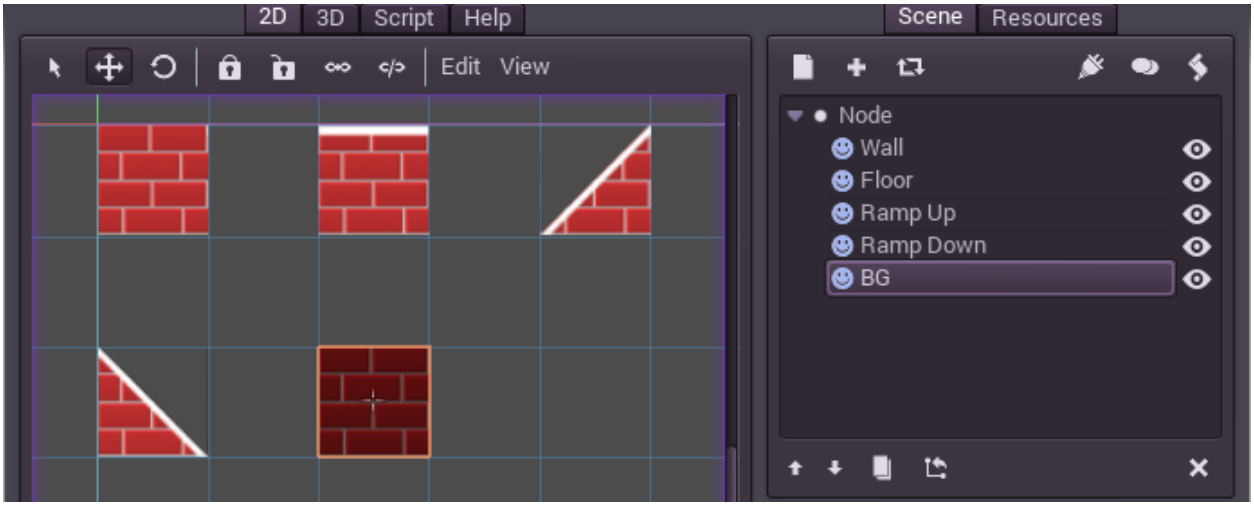
Si mas de un tile esta presente en la imagen de origen, asegúrate de usar la propiedad región del sprite para ajustar cual parte de la textura será usada.

Finalmente, asegúrate de ponerle un nombre correcto a tu sprite, esto para que, en ediciones posteriores al tileset (por ejemplo, si se agrega colisión, se cambia región, etc), el tile igual será **identificado correctamente y actualizado**. Este nombre debe ser único.

Suena como un montón de requerimientos, asique aquí hay un screenshot que muestra donde esta todo lo importante:

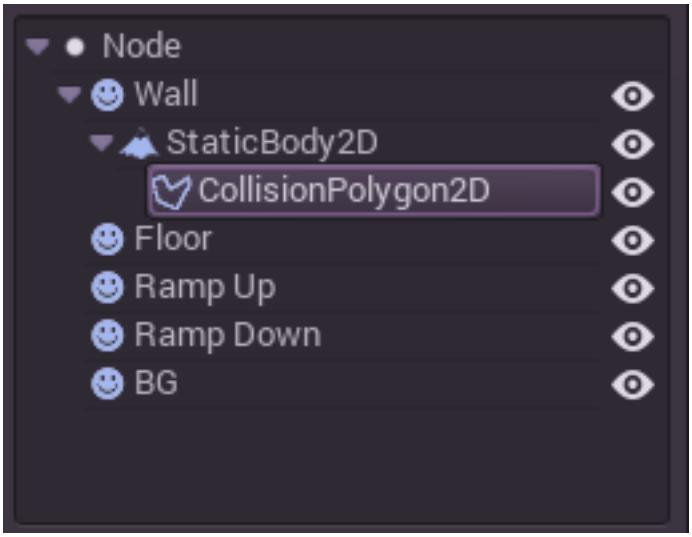


Continua agregando todos los tiles, ajustando los offsets si es necesario (si tiene múltiples tiles en una sola imagen). De nuevo, recuerda que sus nombres deben ser únicos.

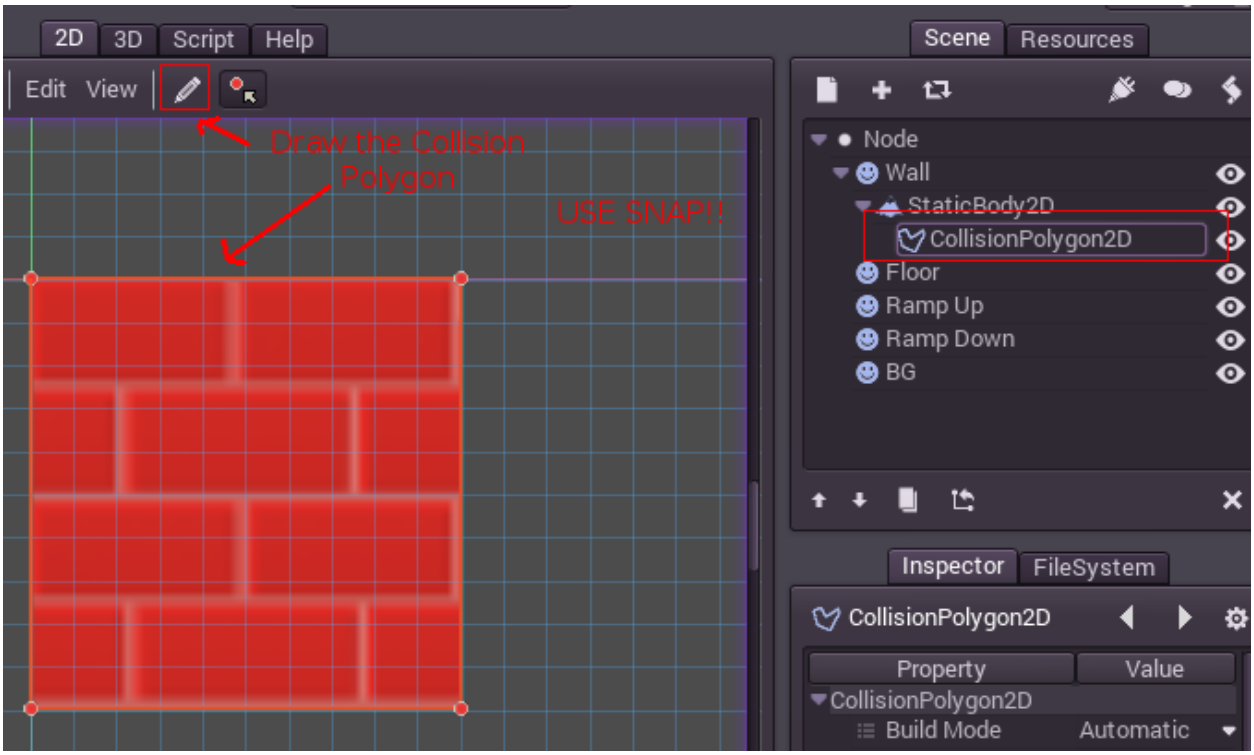


Colisión

Para agregar colisión a un tile, crea un hijo StaticBody2D para cada sprite. Este es un nodo de colisión estática. Luego, como hijo de StaticBody2D, crea una CollisionShape2D o CollisionPolygon. La ultima es recomendada por ser mas sencilla de editar:



Finalmente, edita el polígono, esto le dará una colisión al tile. **Recuerda usar snap!**. Usando snap nos aseguramos que los polígonos de colisiones están alineados correctamente, permitiendo al personaje caminar sin problema de tile a tile. Además **no escales o muevas** la colisión y/o los nodos de polígonos. Déjalos con offset 0,0, con escala 1,1 y rotación 0 respecto al sprite padre.

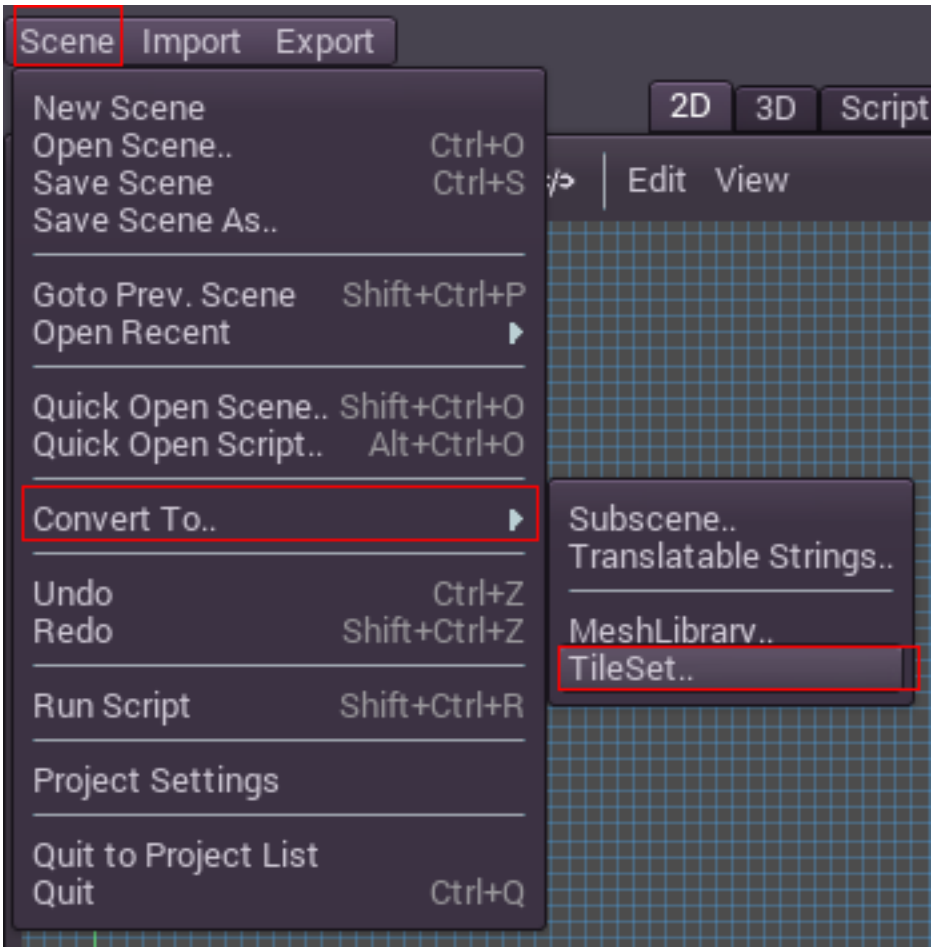


Sigue agregando colisiones a los tiles hasta que este pronto. Ten en cuenta que BG es solo un fondo, por lo que no debe tener colisión.

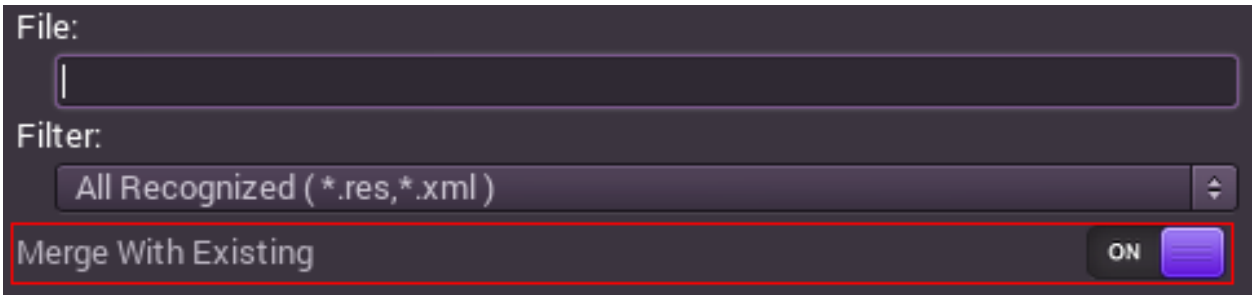
OK! Esta pronto! Recuerdo guardar la escena para ediciones futuras, llámala “tileset_edit.scn” o algo similar.

Exportando un TileSet

Con la escena creada y abierta en el editor, el siguiente paso será crear el tileset. Usa Escena > Convertir A > TileSet desde el menú Escena:



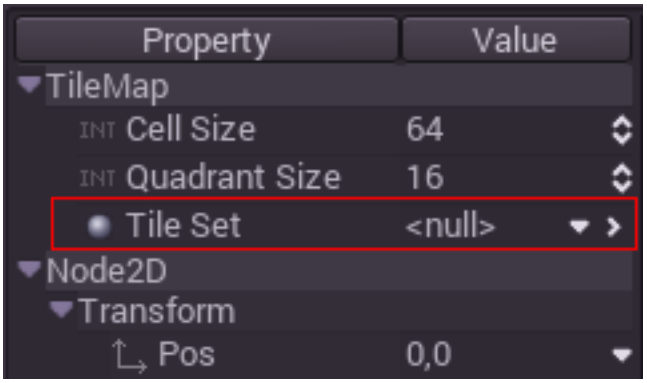
Luego elije el nombre de archivo, como "mistiles.res". Asegúrate que la opción "Mergear con existentes" esta habilitada. De esta forma, cada vez que el recurso de tileset es sobrescrito, los tiles existentes serán mezclados y actualizados (son referenciados por su nombre único, así que nuevamente, **nombre tus tiles adecuadamente**).



Usando el TileSet en un TileMap

Crea una nueva escena, usa cualquier nodo o node2d como raíz, luego crea un *TileMap* como hijo.

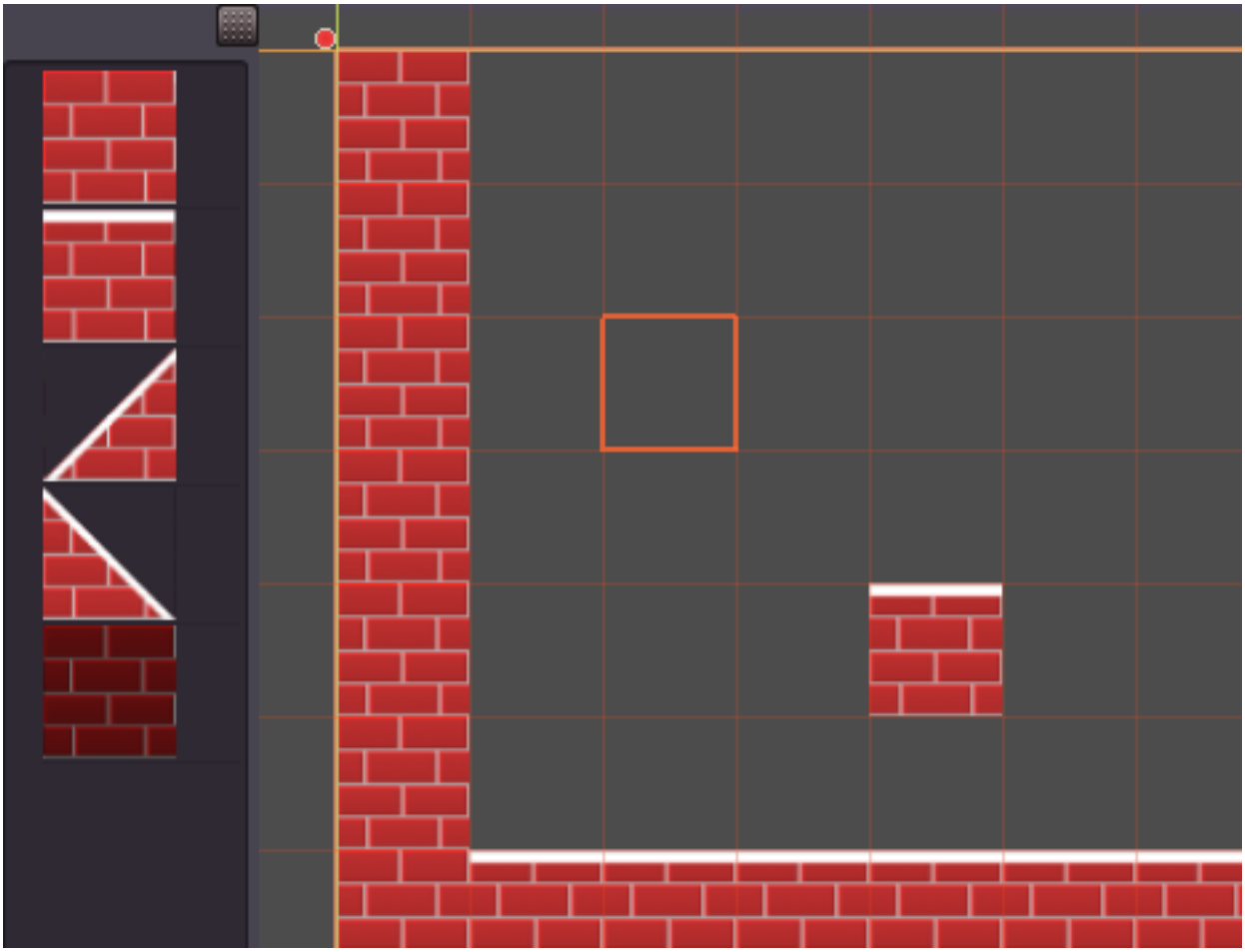
Ve hasta la propiedad tileset de este nodo y asigna el que creamos en los pasos previos:



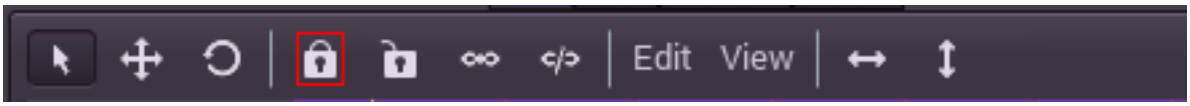
También ajusta el tamaño de cell (celda) a '50', ya que ese es el valor usado por los tiles. Quadrant size es un valor de puesta a punto, que significa que el motor va a dibujar y escoger el tilemap en bloques de baldosas de 16x16. Esta valor suele estar bien y no necesita ser cambiado, pero puede ser usado para poner a punto el rendimiento en casos específicos (si sabes lo q estas haciendo).

Pintando tu mundo

Con todo pronto, asegúrate que el nodo TileMap esta seleccionado. Una grilla roja va a aparecer en la pantalla, permitiendo pintar en ella con el tile seleccionado en la paleta izquierda.

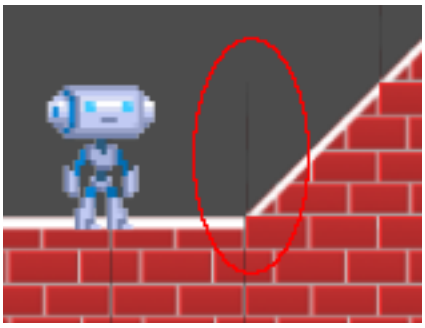


Para evitar mover y seleccionar el nodo TileMap accidentalmente (algo común ya que es un nodo gigante), es recomendable que lo bloquees, usando el botón de candado:



Offset y artefactos de escala

Cuando usas una sola textura para todos los tiles, escalando el tileset (o aun moverlo a un lugar que no esta alineado en pixels) probablemente resultara en artefactos de filtrado como este:



Esto no puede ser evitado, ya que es como funciona el filtro bilinear por hardware. Entonces, para evitar esta situación,

hay algunas soluciones, intenta la que te parezca mejor:

- Usa una sola imagen para cada tile, esto removerá los artefactos pero puede ser pesado de implementar, así que intenta las opciones que están abajo primero.
- Deshabilita el filtrado ya sea para la textura del tileset o el cargador de imágenes entero (ve el tutorial sobre pipeline de assets *Administrando archivos de imagen*).
- Habilita pixel snap (ajusta: "Escena > Configuración de Proyecto" > Display/use_2d_pixel_snap").
- Escalado de Viewport puede a menudo ayudar para encoger el mapa (ve el tutorial *Viewports (Visores)*).

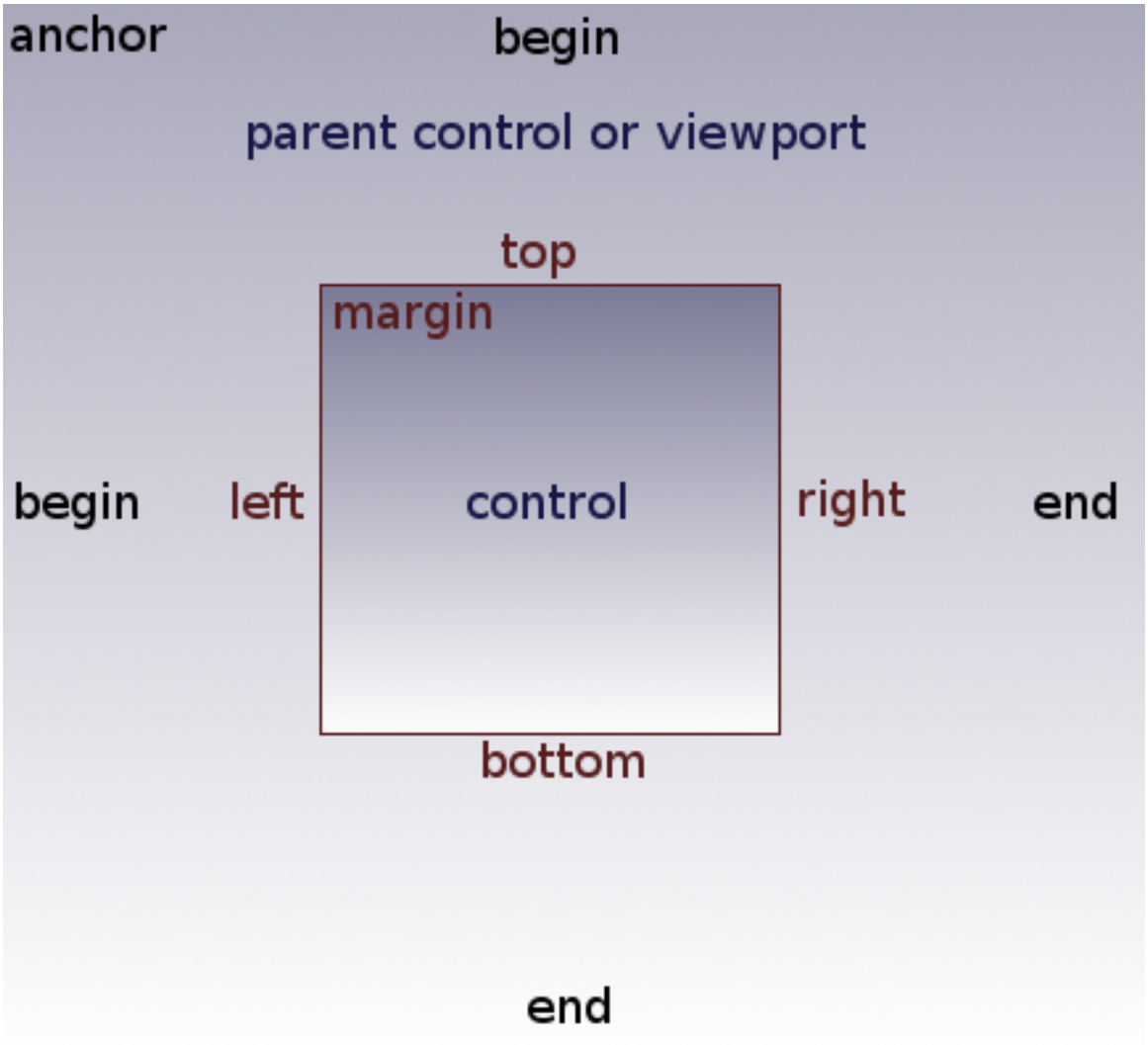
3.2 Interfaz Gráfica de Usuario (GUI)

3.2.1 Tamaño y anclas

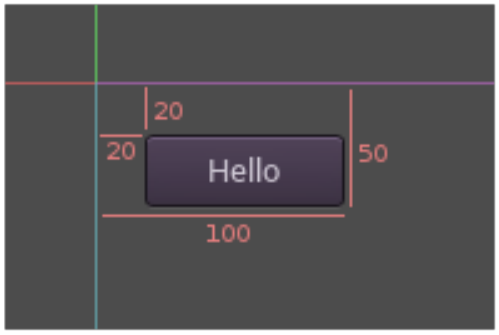
Si un juego siempre se corre en el mismo dispositivo y a la misma resolución, posicionar controles sería tan simple como ajustar la posición y tamaño de cada uno de estos. Desafortunadamente, rara vez es el caso.

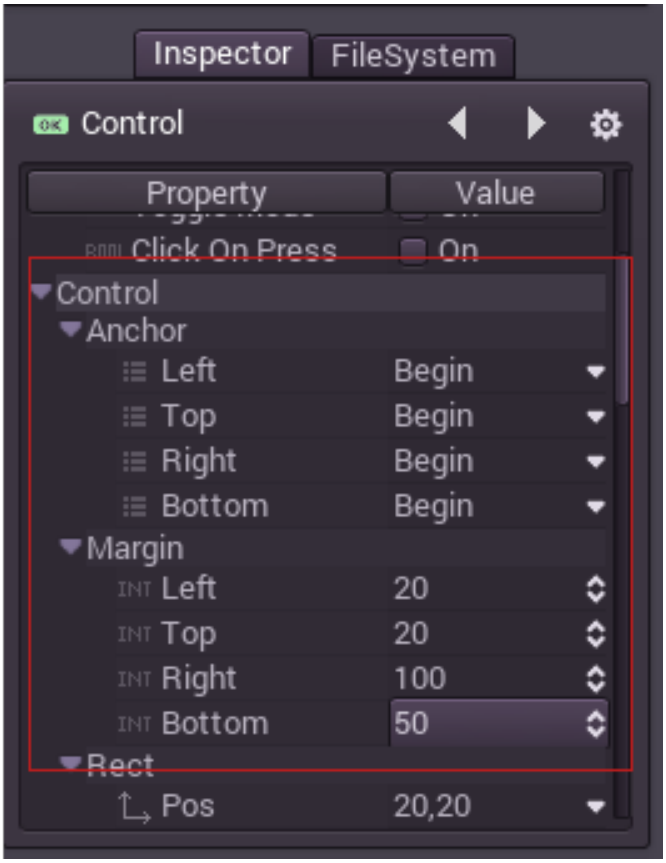
Solo las TVs hoy en día tienen resolución y relación de aspecto standard. Todo lo demás, desde monitores de computadores hasta tablets, consolas portables y teléfonos móviles tienen diferente resolución y relacion de aspecto.

Hay varias formas de manejar esto, pero por ahora vamos a imaginar que la resolución de pantalla cambio y los controles necesitan ser re-posicionados. Algunos necesitan seguir la parte inferior de la pantalla, otros la superior, o tal vez los márgenes derecho o izquierdo.

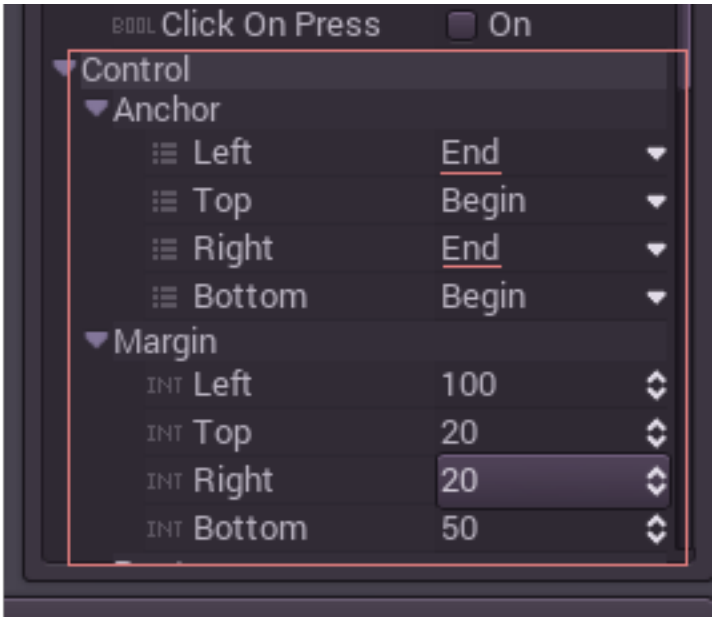


Esto es hecho al editar la propiedad *margin* de los controles. Cada control tiene cuatro márgenes: left(izq), right(der), bottom(inferior) y top(superior). Por defecto todos representan una distancia en pixeles relativa a la esquina superior-izquierda del control padre o (en caso que no haya control padre) el viewport.

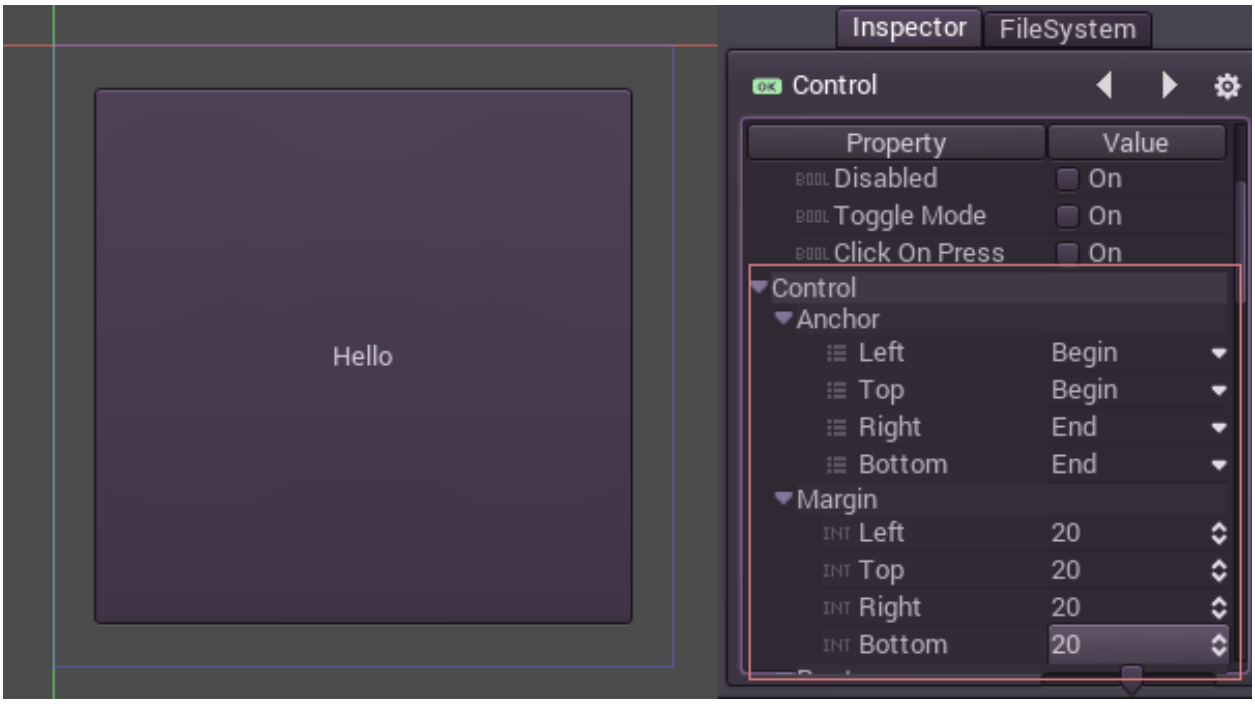




Cuando las anclas horizontales (left, right) y/o vertical (top, bottom) se cambian a END, los valores de márgenes se vuelven relativos a la esquina inferior-derecha del control padre o viewport.



Aquí el control esta ajustado para expandir su esquina inferior-derecha con la del padre, por lo que cuando se cambia de tamaño al padre, el control siempre lo cubrirá, dejando un margen de 20 pixeles:



Finalmente, también esta la opción de ratio (relación), donde 0 implica izquierda, 1 derecha y cualquier valor entre medio es interpolado.

3.2.2 GUI skinning

Oh hermosa GUI!

Este tutorial es sobre skinning avanzado de una interfaz de usuario. La mayoría de los juegos no necesitan esto, ya que terminan usando *Label*, *TextureFrame*, *TextureButton* y *TextureProgress*.

Sin embargo, muchos estilos de juegos a menudo necesitan interfaces de usuario complejas, como MMOs, RPGs tradicionales, simuladores, estrategia, etc. Este tipo de interfaces también son comunes en algunos juegos que incluyen editores para crear contenido, o interfaces para conectividad de red.

La interfaz de usuario de Godot usa este tipo de controles con el tema por defecto, pero pueden ser skinned para lucir como cualquier otra interfaz de usuario.

Tema

La GUI esta skinned con el recurso *Theme*. Los temas contienen toda la información requerida para cambiar el estilo visual de todos los controles. Las opciones de tema son por nombre, por lo que no es obvio que nombre cambia que cosa (especialmente desde código), pero varias herramientas se proveen. El lugar final para fijarse que opción de tema es para que control, y que siempre estará mas actualizado que cualquier documentación es el archivo scene/resources/default_theme/default_theme.cpp. El resto de este documento explicara las diferentes herramientas usadas para personalizar el tema.

Un Tema puede ser aplicado a cualquier control en la escena. Como resultado, todos sus controles hijos y nietos usaran el mismo tema también (a no ser que otro tema sea especificado mas abajo en el árbol). Si un valor no se encuentra en el tema, será buscado en los temas mas arriba en la jerarquía hacia la raíz. Si no se encuentra nada, el tema por defecto es usado. Este sistema permite una sobrescritura flexible de los temas en interfaces de usuario complejas.

Opciones de tema

Cada tipo de opción en un tema puede ser:

- **Un entero constante**: Una única constante numérica. Generalmente usada para definir el espaciado entre componentes o alineación.
- **Un Color**: Un único color, con o sin transparencia. Los colores usualmente se aplican a fuentes e iconos.
- **Una textura**: Una única imagen. Las texturas no se usan a menudo, pero cuando lo son representan manijas para recoger o iconos en un control complejo (como un dialogo de archivo).
- **Una Fuente**: Cada control que usa texto puede tener asignada la fuente usada para dibujar cadenas.
- **Un StyleBox**: StyleBox es un recurso que defino como dibujar un panel en diferentes tamaños (mas información sobre ellos luego).

Toda opción esta asociada a:

- Un nombre (el nombre de la opción)
- Un control (el nombre del control)

Un ejemplo de uso:

```
var t = Theme.new()
t.set_color("font_color","Label",Color(1.0,1.0,1.0))

var l = Label.new()
l.set_theme(t)
```

En el ejemplo de arriba, un nuevo tema es creado. La opcion "font_color" es cambiada y luego aplicada a la etiqueta(label). Como resultado, la etiqueta (y todas las etiquetas hijas y nietas) usaran ese color.

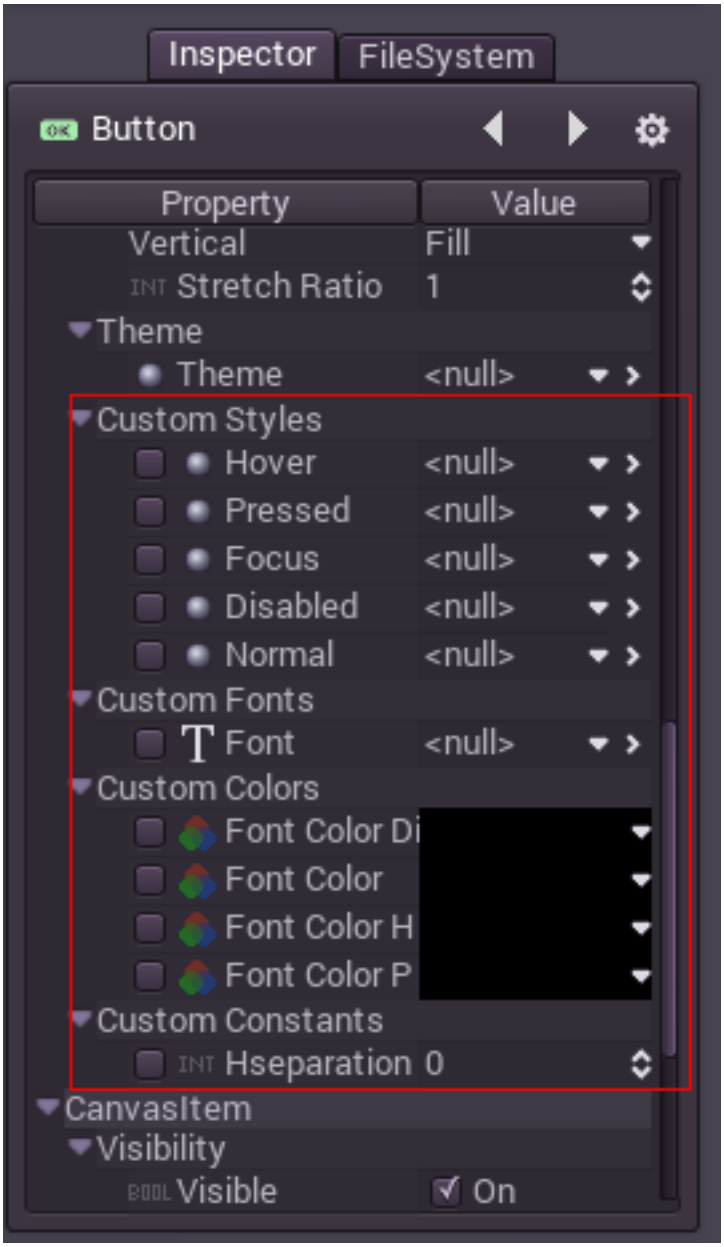
Es posible sobrescribir esas opciones sin usar el tema directamente y solo para un control especifico al usar la API de sobrescritura en *Control.add_color_override()*:

```
var l = Label.new()
l.add_color_override("font_color",Color(1.0,1.0,1.0))
```

En la ayuda integrada de Godot (en la pestaña script) tu puedes chequear que opciones de tema que pueden ser sobrescritas, o chequea la referencia de clase *Control*.

Personalizando un control

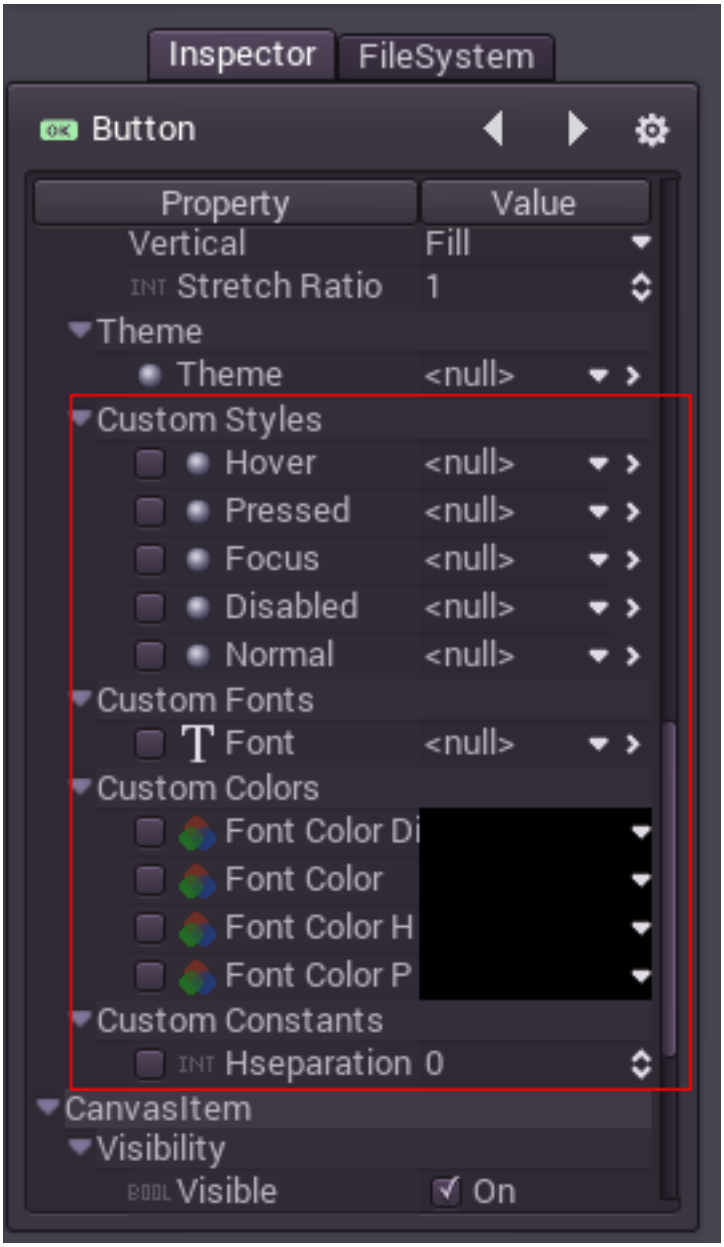
Si solo unos pocos controles necesitan ser skinned, suele no ser necesario crear un nuevo tema. Los controles ofrecen sus propias opciones de tema como propiedades especiales. Si son marcadas, serán sobrescritos:



Como puede verse en la imagen de arriba, las opciones de tema tienen pequeñas check-boxes. Si están marcadas, pueden ser usadas para sobrescribir el valor del tema solo para ese control.

Creando un tema

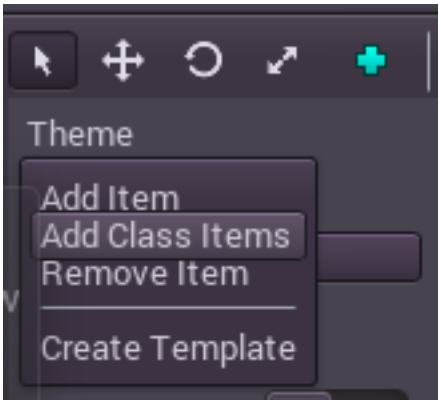
La forma mas simple de crear un tema es editar un recurso theme. Crea un nuevo recurso en el Inspector y selecciona Theme, el editor aparecerá inmediatamente. Luego, guárdalo en el Inspector con el icono de disquette (en, por ejemplo, mytheme.thm):



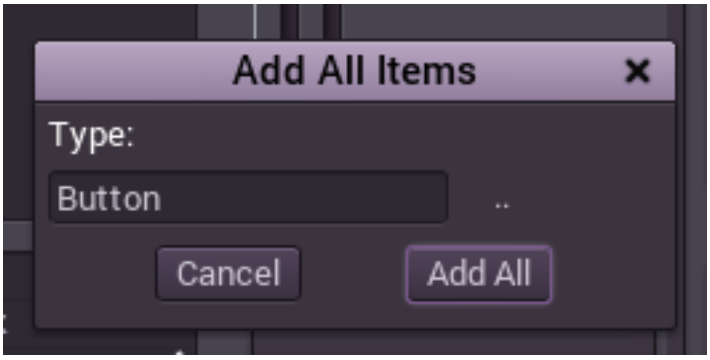
Esto creara un tema vacío que luego puede ser cargado y asignado a los controles.

Ejemplo: aplicando tema a un botón

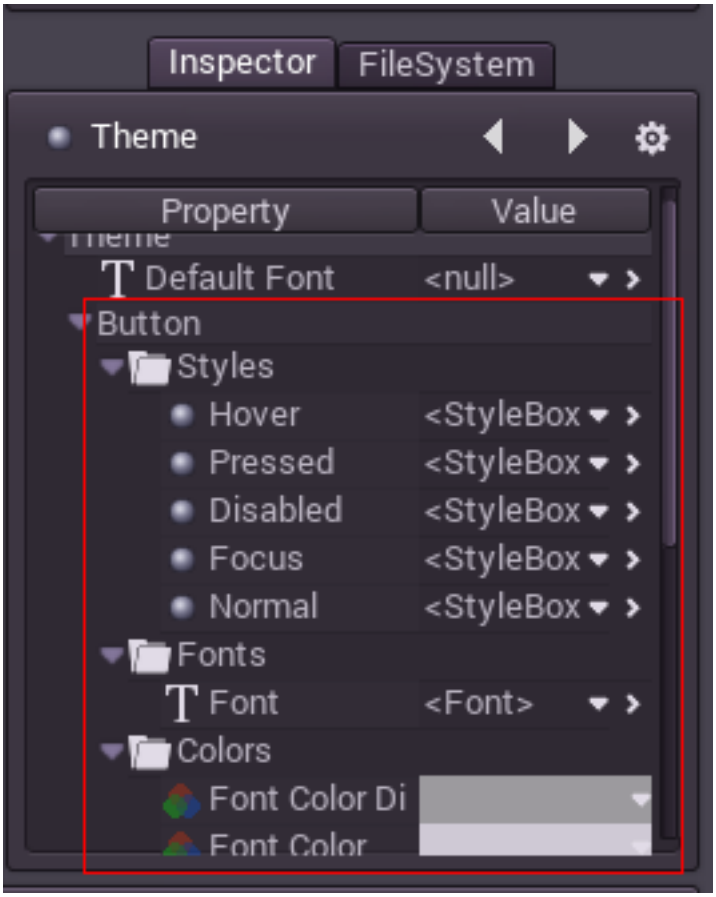
Toma algunos assets (`skin_assets.zip`), ve al menú “theme” y selecciona “Add Class Item”:



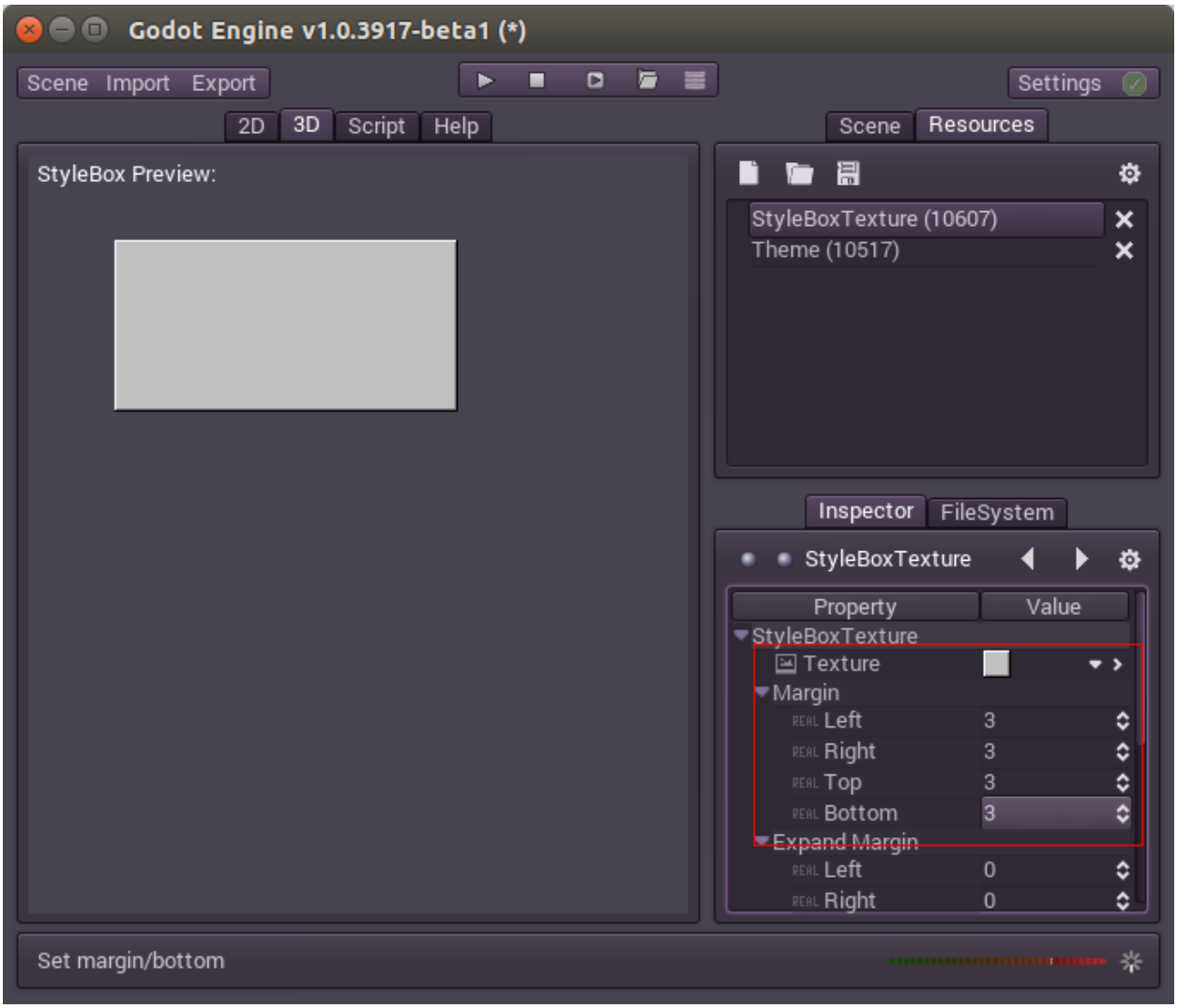
Un menú aparecerá para elegir el tipo de control a crear. Selecciona "Button":



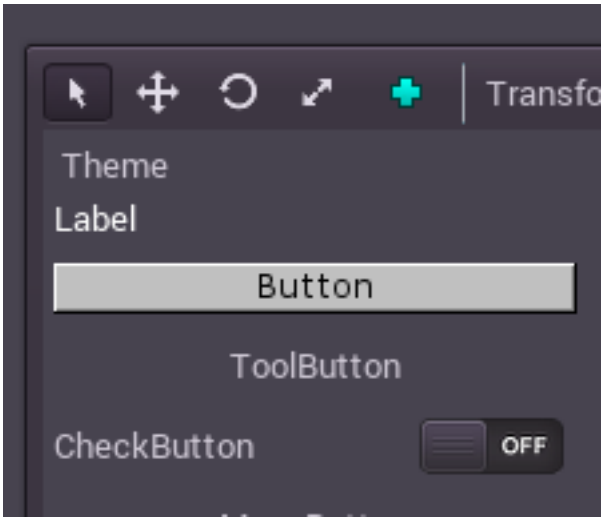
Inmediatamente, todas las opciones de tema para botones aparecerán en las propiedades del editor, donde pueden ser editadas:



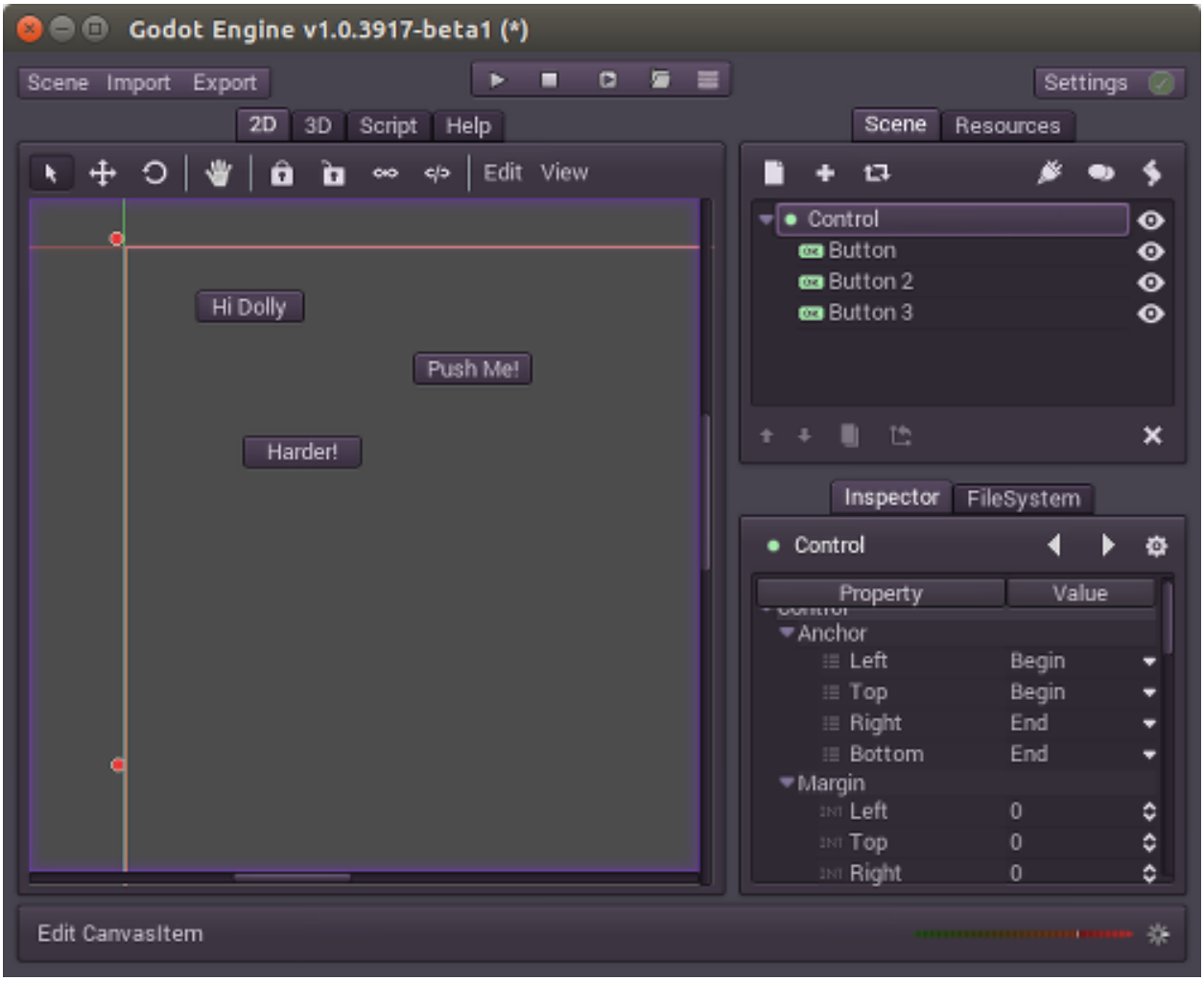
Selecciona el stylebox "normal" y crea un nuevo "StyleBoxTexture", luego edítalo. Una textura stylebox básicamente contiene una textura y el tamaño de los márgenes que no se estiraran cuando la textura sea estirada. Esto se llama stretching "3x3":



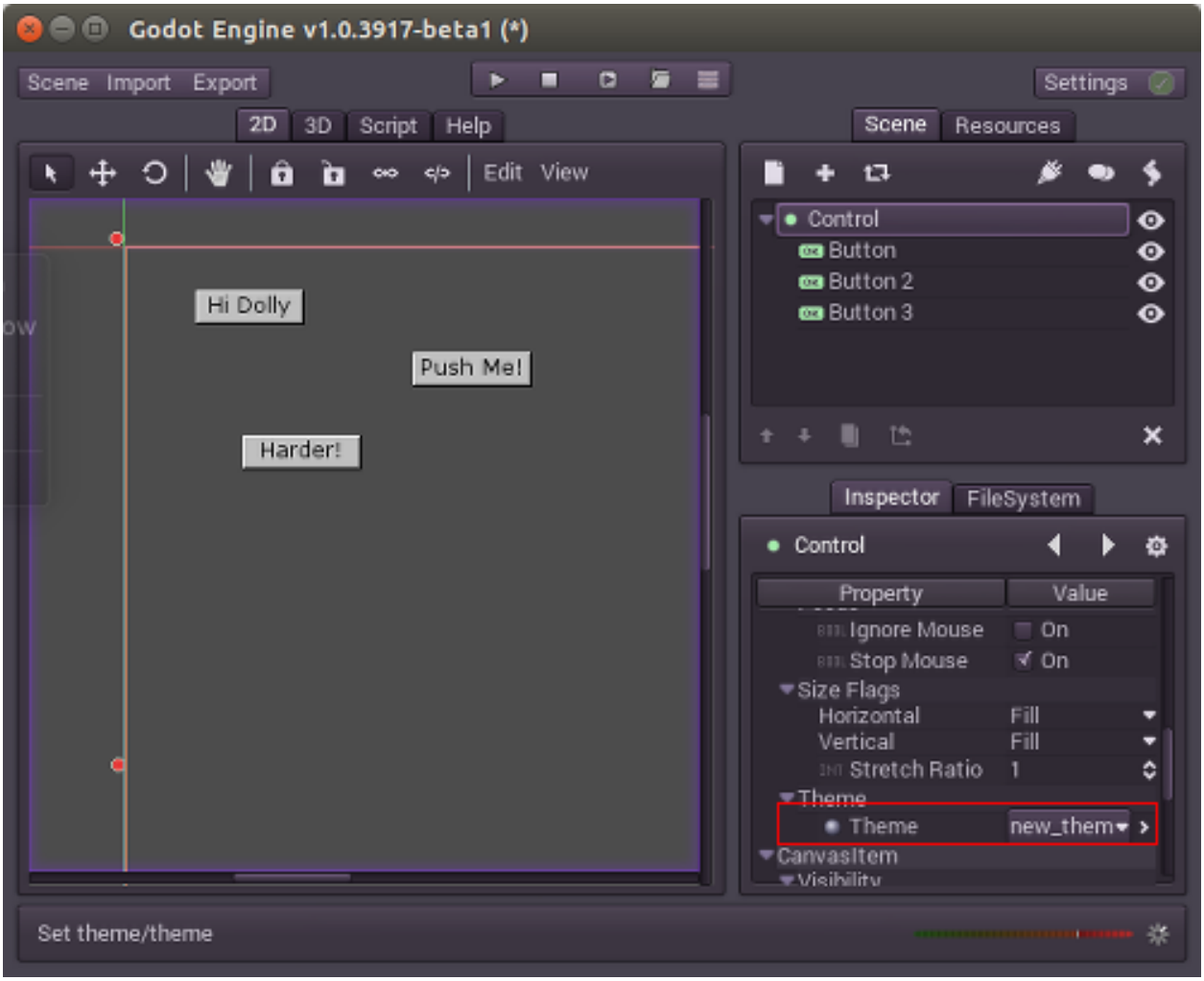
Repite los pasos y agrega otros assets. No hay imagen para los estados hover o deshabilitado en los archivos de ejemplo, por lo que usa el mismo stylebox que normal. Ajusta la fuente proporcionada como la fuente del botón y cambia el color de la fuente a negro. Pronto, tu botón lucirá diferente y retro:



Guarda este tema al archivo .thm. Ve al editor 2D y crea algunos botones:



Ahora, ve al nodo raíz de la escena y encuentra la propiedad "tema", reemplázala por el tema que recién creamos. Debería lucir así:



Felicitaciones! Has creado un tema GUI reusable!

3.2.3 Controles GUI personalizados

Tantos controles...

De todas formas nunca son suficientes. Crear tus propios controles personalizados que actúan exactamente como quieres es una obsesión de casi todo programador GUI. Godot provee muchos, pero puede que no funcionen exactamente en la forma que deseas. Antes de contactar a los desarrolladoras con un pull-request para soportar barras de desplazamiento diagonales, al menos seria bueno saber como crear estos controles fácilmente desde script.

Dibujando

Para dibujar, es recomendado chequear el tutorial *Dibujo personalizado en 2D*. Lo mismo aplica. Vale la pena mencionar algunas funciones debido a su utilidad para dibujar, asique las detallaremos a continuación:

Chequeando el tamaño de un control

A diferencia de los nodos 2D, "size"(tamaño) es muy importante para los controles, ya que ayuda a organizarlos de forma apropiada. Para esto, se provee el método *Control.get_size()*. Chequearlo durante _draw() es vital para asegurar que todo este dentro de los limites.

Chequeando el foco

Algunos controles (como botones o editores de texto) podrían proveer foco de entrada para teclado o joypad. Ejemplos de esto son ingresar texto o presionar un botón. Esto es controlado con la función *Control.set_focus_mode()*. Cuando dibujas, y si el control soporta foco de entrada, siempre es deseable mostrar algún tipo de indicador (highight, box, etc) para indicar que este es el control que tiene el foco. Para chequear por este estatus, existe *Control.has_focus()*.

```
func _draw():
    if (has_focus()):
         draw_selected()
    else:
         draw_normal()
```

Tamaño

Como mencionamos antes, el tamaño es muy importante para los controles. Esto permite distribuirlos adecuadamente, cuando se ajustan en grillas, contenedores, o anclas. Los controles la mayoría del tiempo proveen un *minimum size* para ayudar a distribuirlos adecuadamente. Por ejemplo, si los controles son puestos verticalmente encima uno de otro usando un *VBoxContainer*, el tamaño mínimo nos asegura que tu control personalizado no será aplastado por los demás controles en el contenedor.

Para proveer esta llamada de retorno, solo sobrescribe Control.get_minimum_size(), por ejemplo:

```
func get_minimum_size():
    return Vector2(30,30)
```

O de forma alternativa, ajústalo a través de una función:

```
func _ready():
    set_custom_minimum_size( Vector2(30,30) )
```

Entrada

Los controles proveen algunos ayudantes para hacer la gestión de eventos de entrada mucho mas sencillos que los nodos regulares.

Eventos de entrada

Hay algunos tutoriales sobre entrada antes de este, pero es valioso mencionar que los controles tienen un método especial de entrada que solo funciona cuando:

- El puntero del mouse esta sobre el control.
- El botón fue presionado sobre el control (control siempre captura la entrada hasta que el botón se suelta)
- El control provee foco de teclado/joypad con *Control.set_focus_mode()*.

Esta simple función es *Control._input_event()*. Solo sobrescríbela en tu control. No hay necesidad de ajustar el procesamiento.

```
extends Control

func _input_event(ev):
   if (ev.type==InputEvent.MOUSE_BUTTON and ev.button_index==BUTTON_LEFT and ev.
↪pressed):
       print("Botón izquierdo de mouse presionado!")
```

Para mas información sobre los eventos mismos, chequea el tutorial *InputEvent (Evento de Entrada)*.

Notificaciones

Los controles también pueden tener muchas notificaciones útiles para las cuales no hay llamada de retorno, pero pueden ser chequeadas con la llamada de retorno _notification:

```
func _notification(what):

   if (what==NOTIFICATION_MOUSE_ENTER):
      pass # el mouse entro al área de este control
   elif (what==NOTIFICATION_MOUSE_EXIT):
      pass # el mouse salió del área de este control
   elif (what==NOTIFICATION_FOCUS_ENTER):
      pass # el control tiene foco
   elif (what==NOTIFICATION_FOCUS_EXIT):
      pass # el control perdió el foco
   elif (what==NOTIFICATION_THEME_CHANGED):
      pass # el tema usado para dibujar el control cambio
      # update y redraw son recomendados si se usa un tema
   elif (what==NOTIFICATION_VISIBILITY_CHANGED):
      pass # el control se volvió visible/invisible
      # chequea el nuevo estado si is_visible()
   elif (what==NOTIFICATION_RESIZED):
      pass # el control cambio de tamaño, chequea el nuevo tamaño
      # con get_size()
   elif (what==NOTIFICATION_MODAL_CLOSED):
      pass # para ventanas emergentes modales, notificación
           # que la ventana emergente fue cerrada
```

3.3 Fisica

3.3.1 Introducción a la física

Nuestro mundo esta hecho de materia tangible. En nuestro mundo, un piano no puede ir a través de la pared cuando es ingresado a una casa. Necesita usar la puerta. Los videojuegos a menudo son como el mundo real por lo que Pac-Man no puede ir a través de las paredes de su laberinto (aunque se puede tele transportar desde el lado izquierdo al derecho de la pantalla y viceversa).

Es decir, mover sprites por ahí es lindo pero un día tienen que colisionar adecuadamente, asique vayamos al punto.

Shapes (Formas)

El objeto base que puede colisionar en el mundo 2D de Godot es un *Shape2D*.

- *CircleShape2D*
- *RectangleShape2D*
- *CapsuleShape2D*
- *ConvexPolygonShape2D*
- *ConcavePolygonShape2D*
- etc. (hay mas, chequea la lista de clases).

Shapes pueden ser del tipo *Resource*, pero pueden ser creados desde código fácilmente. Por ejemplo:

```
# Crea un círculo
var c = CircleShape2D.new()
c.set_radius(20)

# Crea una caja
var b = RectangleShape2D.new()
b.set_extents(Vector2(20,10))
```

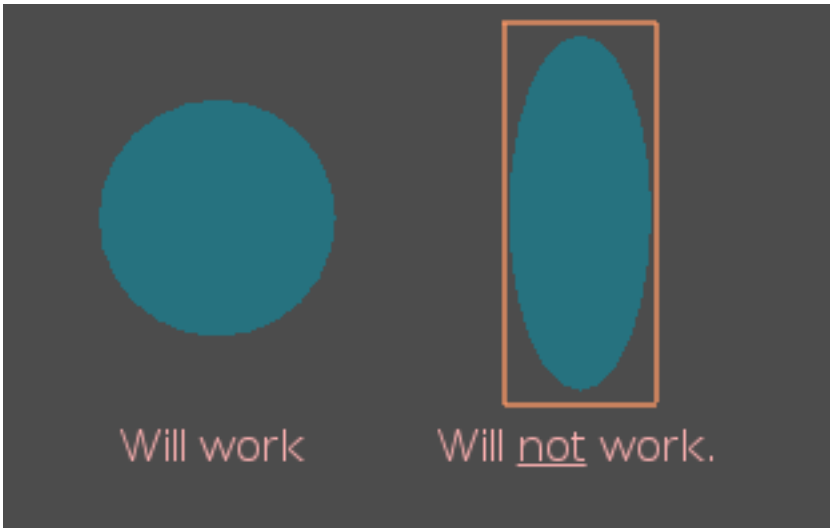
El uso principal de los shapes es chequear colisiones/intersecciones y obtener información de resolución. Son principalmente convexas, (excepto concavepolygon, que no es mas que una lista de segmentos contra la cual chequear colisiones). Este chequeo de colisión es hecho fácilmente con funciones incorporadas como:

```
# Chequea si hay una colisión entre dos formas, cada una tiene un transform
if b.collide(b_xform, a, a_xform):
    print("OMG Colisión!")
```

Godot retornara informació correcta de colisión y de información de colisión desde las diferentes llamadas a la API Shape2D. La colision entre todas las formas y transformaciones pueden ser hechas de esta manera, o aun obtener información de contacto, motion casting, etc.

Transformando shapes

Como vimos antes en las funciones de colisión, los shapes 2D en Godot pueden ser transformados usando una transformación regular *Matrix32*, lo que significa que puede chequear una colisión cuando es escalada, movida o rotada. La única limitación para esto es que las formas con secciones curvas (como circulo y capsula) pueden solo ser escaladas uniformemente. Esto implica que las formas de circulo o capsula escaladas en forma de elipse **no funcionaran adecuadamente**. Esta es una limitación en el algoritmo de colisión usado (SAT), así que asegúrate que tus formas de circulo y capsula siempre se escalen uniformemente!



Cuando comienzan los problemas

Aunque esto suena bien, la realidad es que usualmente la detección de colisiones sola no es suficiente en la mayoría de los escenarios. Muchos problemas aparecen en la medida que el desarrollo del juego progresa:

Demasiadas combinaciones!

Los juegos tienen varias docenas, cientos, miles! de objetos que pueden colisionar y ser colisionados. El enfoque típico es probar todo contra todo en dos bucles for como este:

```
for i in colliders:
    for j in colliders:
        if (i.collides(j)):
            do_collision_code()
```

Pero esto escala realmente mal. Imaginemos que hay solo 100 objetos en el juego. Esto implica que 100*100=10000 colisiones necesitaran ser probadas en cada frame. Esto es mucho!

Ayuda visual

La mayoría del tiempo, crear un shape con código no es suficiente. Necesitamos ubicarlo visualmente sobre nuestro sprite, dibujar un polígono de colisión, etc. Es obvio que necesitamos nodos para crear las formas de colisión apropiadas en una escena.

Resolución de colisión

Imagina que resolvimos el problema de colisión, podemos saber de forma fácil y rápida que formas se superponen. Si muchas de ellas son objetos dinámicos que se mueven por ahí, o que se mueven de acuerdo a física newtoniana, resolver la colisión de muchos objetos puede ser realmente difícil desde código.

Introduciendo... Motor de física de Godot!

Para resolver todos esos problemas, Godot tiene un motor de física y colisión que esta bien integrado en el sistema de escena, y de todas formas permite diferentes niveles y capas de funcionalidad. El motor de física incorporado puede ser usado para:

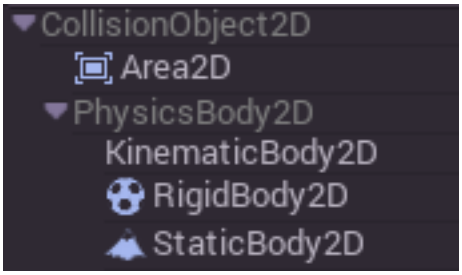
- Detección simple de colisión: Ve la API *Shape2D*
- Cinemática de Escena: Maneja formas, colisiones, broadphase, etc como nodos. Ve *Area2D*.
- Física de Escena: Cuerpos rígidos y restricciones como nodos. Ve *RigidBody2D*, y los nodos joint.

Unidad de medida

A menudo es un problema cuando se integra un motor de física 2D en un juego, que dichos motores están optimizados para trabajar usando metros como unidad de medida. Godot utiliza un motor de física 2D incorporado y personalizado que esta diseñado para funcionar adecuadamente en pixels, así que todas las unidades y valores por defecto usados para estabilización están preparados para esto, haciendo el desarrollo mas directo.

CollisionObject2D

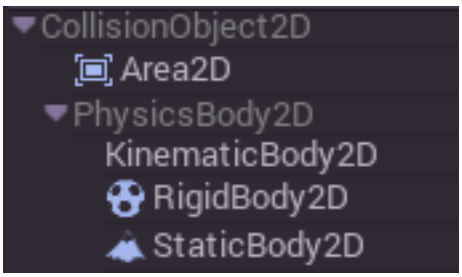
CollisionObject2D es el nodo (virtual) base para todo lo que puede colisionar en 2D. Area2D, StaticBody2D, KinematicBody2D y RigidBody2D todos heredan desde el. Este nodo contiene una lista de formas (Shape2D) y una transformación relativa. Esto significa que todos los objetos colisionables en Godot pueden usar múltiples formas en diferentes transformaciones (offset/scale/rotation). Solo recuerda que, como mencionamos antes, **escalado no uniforma no funcionara para formas de circulo o capsulas**.



StaticBody2D

El nodo mas simple en el motor de física es el StaticBody2D, el cual provee una colisión estática. Esto implica que otros objetos pueden colisionar contra el, pero StaticBody2D no se moverá por si mismo o generara algún tipo de interacción cuando colisione otros cuerpos. Esta allí solo para ser colisionado.

Crear uno de estos cuerpos no alcanza, porque carece de colisión:



Por el punto previo, sabemos que los nodos derivados de ColiisionObject2D tienen una lista interna de formas y transformaciones para colisiones, pero como lo editamos? Hay dos nodos especiales para ello.

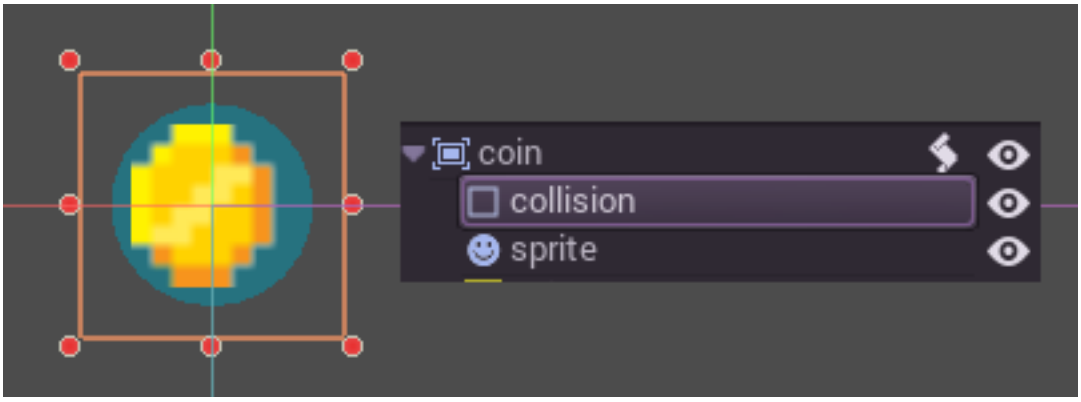
CollisionShape2D

Este nodo es un nodo ayudante. Debe ser creado como un hijo directo de un nodo derivado de CollisionObject2D: *Area2D*, *StaticBody2D*, *KinematicBody2D*, *RigidBody2D*.

Por si mismo no hace nada, pero cuando se crea como hijo de los nodos mencionados arriba, les agrega forma de colisión. Cualquier cantidad de hijos CollisionShape2D pueden ser creados, lo que significa que el objeto padre simplemente tendrá mas formas de colisión. Cuando se agrega/borra/mueve/edita, actualiza la lista de formas en el nodo padre.

En tiempo de ejecución, sin embargo, esto nodo no existe (no puede ser accedido con `get_node()`), ya que solo esta destinado para ser un ayudante en el editor. Para acceder a las formas en tiempo de ejecución, usa la API CollisionObject2D directamente.

Como un ejemplo, aquí esta la escena del juego plataformer (se puede descargar junto con los demás demos desde el sitio oficial de Godot), conteniendo un Area2D con hijo CollisionObject2D como icono de moneda:



Triggers (disparadores)

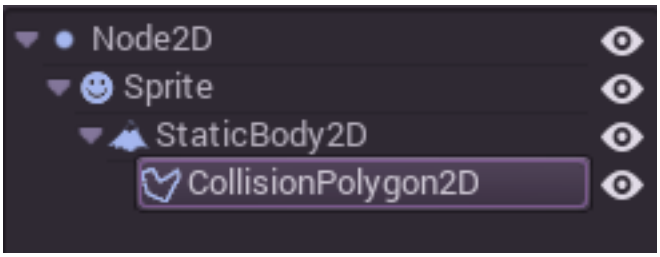
Un CollisionShape2D o CollisionPolygon2D puede ser ajustado como un trigger. Cuando se usa en un RigidBody2D o KinematicBody2D, las formas "trigger" se vuelven no colisionables (los objetos no pueden colisionar contra el). Solo se mueven al rededor de los objetos como fantasmas. Esto los vuelve útiles en dos situaciones:

- Deshabilitar la colisión en una forma especifica.
- Hacer que Area2D dispare una señal body_enter / body_exit con objetos no colisionables (útil en varias situaciones).

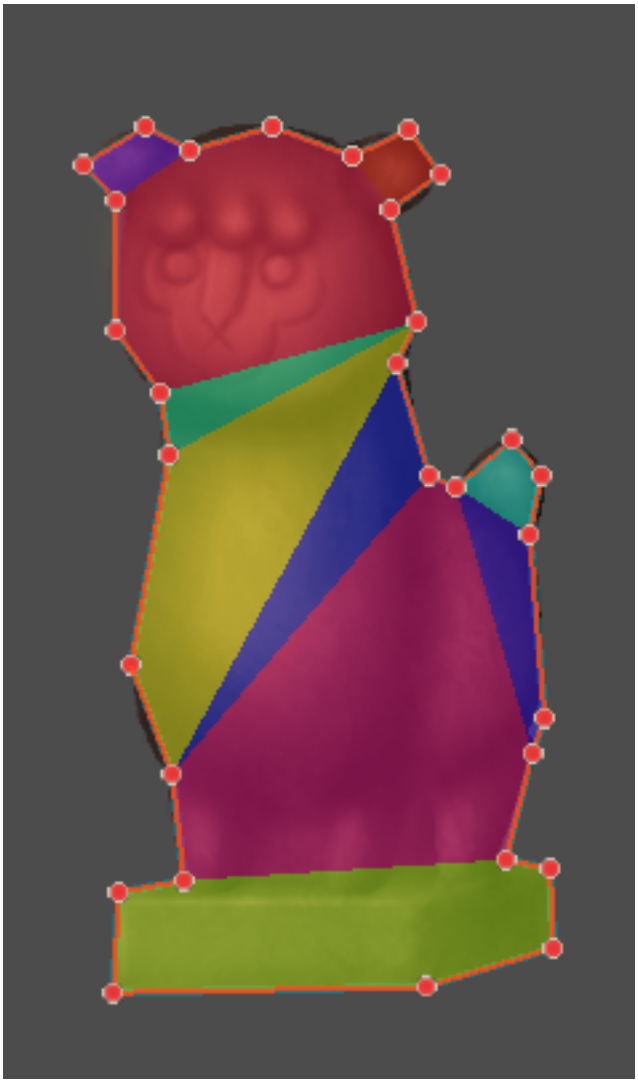
CollisionPolygon2D

Este es similar a CollisionShape2D, excepto que en lugar de asignarse una forma, un polígono puede ser editado (dibujado por el usuario) para determinar la forma. El polígono puede ser convexo o cóncavo, no importa.

Yendo atrás, aquí esta la escena con el StaticBody2D, el static body es hijo de un sprite (lo que significa que si el sprite se mueve, la colisión también lo hace). CollisionPolygon es un hijo de staticbody, lo que implica que le agrega formas de colisión.



De hecho, lo que hace CollisionPolygon es descomponer el polígono en formas convexas (los shapes solo pueden ser convexos, recuerdan?) y agregarla a CollisionObject2D:



KinematicBody2D

KinematicBody2D son tipos especiales de cuerpos que están pensados para ser controlados por el usuario. No son afectados por la física en lo absoluto (para otros tipos de cuerpos, como un character o rigidbody, estos son los mismos que un staticbody). Tienen sin embargo, dos usos principales:

- **Movimiento Simulado**: Cuando estos cuerpos son movidos manualmente, ya sea desde código o desde *AnimationPlayer* (con proccess mode ajustado a fixed!), la física va a computar automáticamente un estimado de su

velocidad linear y angular. Esto los vuelve muy útiles para mover plataformas u otros objetos controlados desde AnimationPlayer (como una puerta, un puente que se abre, etc). Como un ejemplo, el demo 2d/platformer los usa para mover plataformas.

- **Personajes Cinemáticos**: KinematicBody2D también tiene una API para mover objetos (la función move()) mientras realiza pruebas de colisión. Esto los vuelve realmente útiles para implementar personajes que colisionan contra un mundo, pero que no requieren física avanzada. Hay un tutorial sobre esto *Kinematic Character (2D) (Personaje Cinemático)*.

RigidBody2D

Este tipo de cuerpo simula física newtoniana. Tiene masa, fricción, rebote, y las coordenadas 0,0 simulan el centro de masa. Cuando se necesita física real, *RigidBody2D* es el nodo ha usar. El movimiento de este cuerpo es afectado por la gravedad y/u otros cuerpos.

Los Rigid bodies suelen estar activos todo el tiempo, pero cuando terminan en una posición de descanso y no se mueven por un rato, son puestos a dormir hasta que algo los despierte. Esto ahorra una cantidad enorme de CPU.

Los nodos RigidBody2D actualizan sus trasformaciones constantemente, ya que es generada por la simulación desde una posición, velocidad linear y velocidad angular. Como resultado, [STRIKEOUT:this node can't be scaled]. Escalar los nodos hijos debería funcionar sin embargo.

Como un plus, ya que es muy común en los juegos, es posible cambiar un nodo RigidBody2D para que se comporte como un Character (sin rotación), StaticBody o KinematicBody de acuerdo a diferentes situaciones (ejemplo, un enemigo congelado por hielo se vuelve un StaticBody).

La mejor forma de interactuar con un RigidBody2D es durante la llamada de retorno force integration. En este momento exacto, el motor físico sincroniza estado con la escena y permite modificación completa de los parámetros internos (de otra forma, ya que puede estar corriendo en un thread, los cambios no tendrán lugar hasta el siguiente frame). Para hacer esto, la siguiente función debe ser sobrescrita:

```
func _integrate_forces(state):
    [use state to change the object]
```

El parámetro "state" es del tipo *Physics2DDirectBodyState*. Por favor no uses este objeto (estado) fuera de la llamada de retorno ya que resultara en un error.

Reporte de contacto

En general, RigidBody2D no se mantendrá al corriente de los contactos, ya que esto puede requerir una cantidad enorme de memoria si miles de rigid bodies están en la escena. Para tener los contactos reportados, simplemente aumenta la cantidad de la propiedad "contacts reported" desde cero hasta un valor que tenga sentido (dependiendo en cuantos esperas obtener). Los contactos pueden ser las tarde obtenidos con *Physics2DDirectBodyState.get_contact_count()* y las funciones relacionadas.

El monitoreo de contactos a través de señales también esta disponible (las señales son similares a las de Area2D, descritas mas abajo) con propiedades booleanas.

Area2D

Las áreas en la física de Godot tienen tres roles principales:

1. Sobrescribir los parámetros de espacio para objetos que entran a ellas (ejemplo: gravedad, dirección de gravedad, tipo de gravedad, densidad, etc).

2. Monitorear cuando rigid o kinematic bodies entrar o salen del área.

3. Monitorear otras áreas (esta es la forma mas simple de probar la superposición)

La segunda función es la mas común. Para que funcione, la propiedad "monitoring" debe estar habilitada (lo esta por defecto). Hay dos tipos de señales emitidas por este nodo:

```
# Notificación simple, de alto nivel
body_enter(body:PhysicsBody2D)
body_exit(body:PhysicsBody2D)
area_enter(area:Area2D)
area_exit(body:Area2D)

# Notificación de bajo nivel basada en shape
# Notifica específicamente cual shape en ambos, cuerpo y área, están en contacto
body_enter_shape(body_id:int,body:PhysicsBody2D,body_shape_index:int,area_shape_
↪index:idx)
body_exit_shape(body_id:int,body:PhysicsBody2D,body_shape_index:int,area_shape_
↪index:idx)
area_enter_shape(area_id:int,area:Area2D,area_shape_index:int,self_shape_index:idx)
area_exit_shape(area_id:int,area:Area2D,area_shape_index:int,self_shape_index:idx)
```

Por defecto, las áreas también reciben entrada de mouse/touchscreen, proveyendo una forma de mas bajo nivel que controla la implementación de este tipo de entrada en un juego.

En caso de overlap, quien recibe la información de colisión?

Recuerda que no toda combinación de dos cuerpos puede hacer "report" de contactos. Los cuerpos Static son pasivos y no reportaran contacto cuando son tocados. Los cuerpos Kinematic reportaran contactos pero solo contra cuerpos Rigid/Character. Area2D reportara overlap (no el contacto detallado) con cuerpos y otras áreas. La siguiente tabla debería hacerlo mas visual:

Tipo	*RigidBody*	*CharacterBody*	*KinematicBody*	*StaticBody*	*Area*
RigidBody	Both	Both	Both	Rigidbody	Area
CharacterBody	Both	Both	Both	CharacterBody	Area
KinematicBody	Both	Both	None	None	Area
StaticBody	RigidBody	CharacterBody	None	None	None
Area	Area	Area	Area	None	Both

Variables globales de física

Algunas variables globales pueden ser modificadas en la configuración de proyecto para ajustar como funciona la física 2D:

Dejarlos sin tocar es lo mejor (excepto para la gravedad, que necesita ser ajustada en la mayoría de los juegos), pero hay un parámetro especifico que puede necesitar ser ajustado "cell_size". El motor físico de Godot usa por defecto un algoritmo de space hashing que divide el espacio en celdas para computar pares de colisiones cercanas mas eficientemente.

Si un juego utiliza varios colisionadores que son realmente pequeños y ocupan una pequeña porción de la pantalla, puede ser necesario encoger ese valor (siempre a una potencia de 2) para mejorar la eficiencia. De la misma manera si un juego usa pocos colisionadores grandes que cubren un mapa gigante (del tamaño de varias pantallas), aumentar ese valor un poco puede ayudar a salvar recursos.

Llamada de retorneo fixed process

El motor físico puede iniciar múltiples threads para mejorar el rendimiento, para que puede utilizar hasta un frame completo para procesar física. Por este motivo, cuando se acceden variables de física como la posición, velocidad linear, etc. pueden no ser representativas de lo que esta sucediendo en el frame actual.

Para resolver esto, Godot tiene una llamada de retorno fixed process, que es como process pero es llamada una vez por frame de física (por defecto 60 veces por segundo). Durante este tiempo, el motor de física esta en estado de *sincronización* y puede ser accedido directamente sin demoras.

Para habilitar una llamada de retorno fixed process, usa la función `set_fixed_process()`, ejemplo:

```
extends KinematicBody2D

func _fixed_process(delta):
    move(direction * delta)

func _ready():
    set_fixed_process(true)
```

Casting rays y consultas de movimiento

Es muy a menudo necesario "explorar" el mundo al rededor desde nuestro código. Emitir rays (rayos) es la forma mas común de hacerlo. La forma mas simple de hacer esto es usando el nodo RayCast2D, el cual va a emitir un rayo en cada frame y grabar la intersección.

Por el momento no hay una API de alto nivel para esto, así que el servidor de física debe ser usado directamente. Para esto, la clase *Physics2DDirectspaceState* debe ser usada. Para obtenerla, los siguiente pasos deben ser tomados:

1. Debe ser usada dentro de la llamada de retorno `_fixed_process()`, o en `_integrate_forces()`
2. Los RIDs 2D para el servidor de espacio y física deben ser obtenidos.

El siguiente código debería funcionar:

```
func _fixed_process(delta):
    var space = get_world_2d().get_space()
    var space_state = Physics2DServer.space_get_direct_state(space)
```

Disfruta haciendo consultas de espacio!

3.3.2 Kinematic Character (2D) (Personaje Cinemático)

Introducción

Si, el nombre suena extraño. "Kinematic Character". Que es? La razón es que cuando salieron los motores de física, se llamaban motores "Dynamics"(Dinámicos) (porque se encargaban principalmente de las respuestas de las colisiones). Muchos intentos tuvieron lugar para crear un character controller usando los motores dinámicos pero no era tan simple como parecía. Godot tiene una de las mejores implementaciones de dynamic character controller que puedes encontrar

(basta chequear el demo 2d/platformer), pero usarlo requiere un nivel considerable de habilidad y entendimiento de los motores de física (o un montón de paciencia con prueba y error).

Algunos motores de física como Havok parecen jurar que los dynamic character controllers son la mejor alternativa, mientras otros (PhysX) prefiere promover los de tipo Kinematic.

Entonces, cual es realmente la diferencia?:

- Un **dynamic character controller** usa un rigid body con tensor inercial infinito. Básicamente, es un rigid body que no puede rotar. Los motores de fisica siempre permiten que los objetos colisionen, luego resuelven las colisiones todas juntas. Esto hace que los dynamic character controllers sean capaces de interactuar con otros objetos físicos sin problemas (como se ve en el demo platformer), sin embargo estas interacciones no siempre son predecibles. Las colisiones también pueden tomar mas de un frame para ser resueltas, por lo que algunas colisiones puede parecer que se desplazan un poquito. Esos problemas pueden ser arreglados, pero requieren una cierta cantidad de habilidad.
- Un **kinematic character controller** se asume que siempre empieza en un estado de no-colisionar, y siempre se moverá a un estado de no-colisionar. Si empieza en un estado de colisión, tratara de liberarse (como hacen los rigid bodies) pero esta es la excepción, no la regla. Esto vuelve su control y movimiento mucho mas predecible y fácil de programar. Sin embargo, como contrapartida, no pueden interactuar directamente con otros objetos de física (a no ser que se haga a mano en el codigo).

Este corto tutorial se enfocara en los kinematic character controllers. Básicamente, la forma antigua de manejar colisiones (que no es necesariamente mas simple bajo el capó, pero bien escondida y presentada como una linda y simple API).

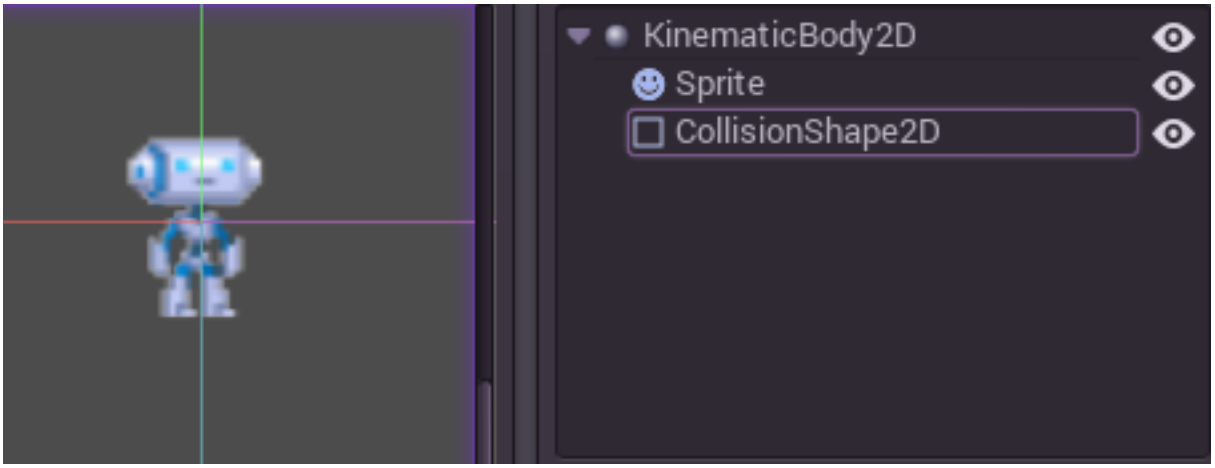
Fixed process

Para manejar la lógica de cuerpos o personajes kinematic, siempre es recomendable usar fixed process, el cual es llamado la misma cantidad de veces por segundo, siempre. Esto hace que los cálculos de física y movimiento funcionen de una forma mas predecible que usando el process regular, el cual puede tener picos o perder precisión si el frame rate (cuadros por segundo) es demasiado alto o demasiado bajo.

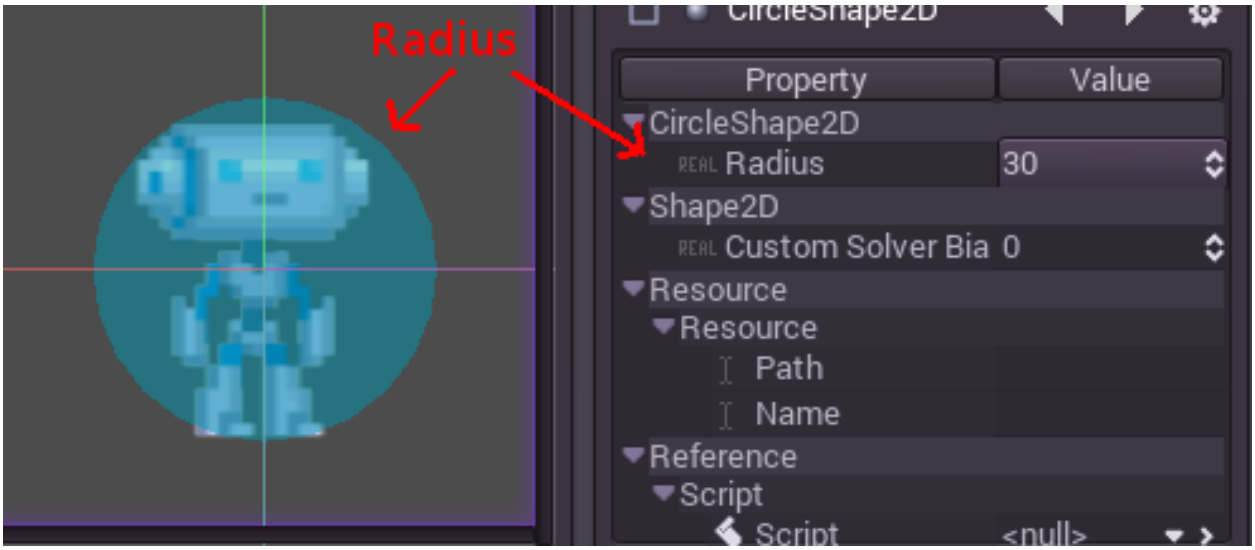
```
extends KinematicBody2D

func _fixed_process(delta):
    pass

func _ready():
    set_fixed_process(true)
```

Configuración de la escena

Para tener algo que probar, aquí esta la escena (del tutorial de tilemap): `kbscene.zip`. Vamos a estar creando una nueva escena para el personaje. Usa los sprites del robot y crea una escena como esta:



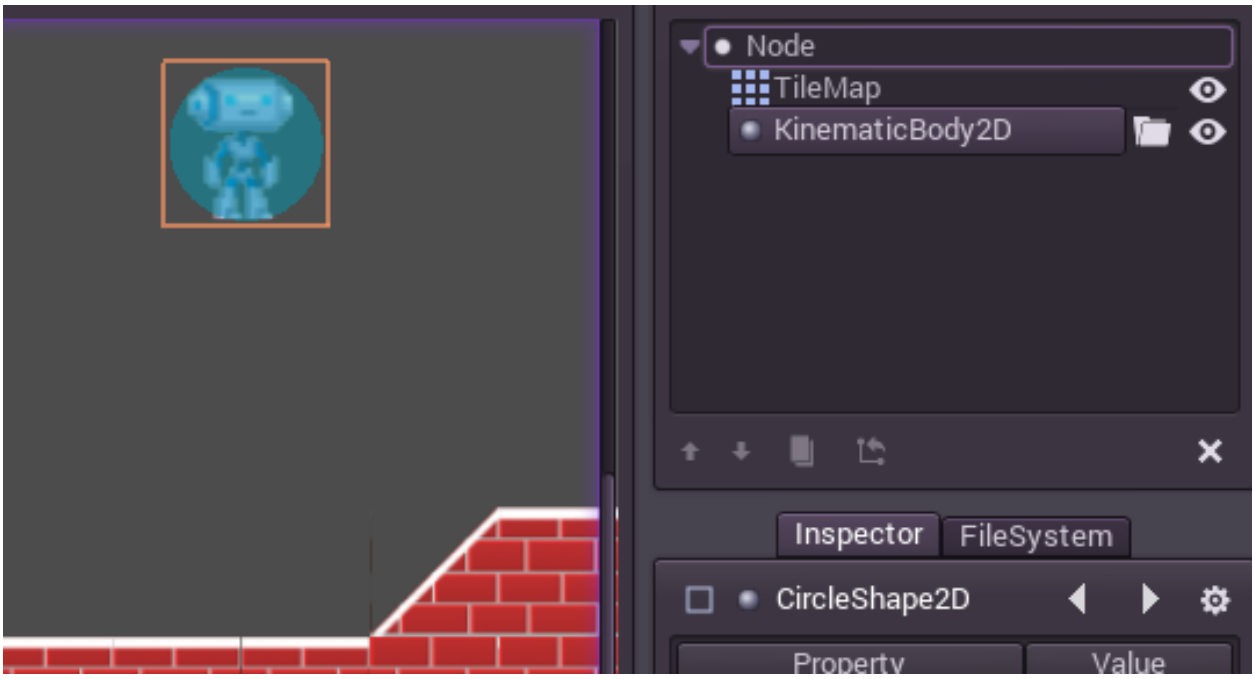
Agreguemos una forma de colisión circular al collision body, crea una nueva CircleShape2D en la propiedad shape de CollisionShape2D. Ajusta el radio a 30:



Nota: Como se menciono antes en el tutorial de física, el motor de física no puede manejar escala en la mayoría de las formas (solo los polígonos de colisión, planos y segmentos parecen funcionar), así que siempre cambia los parámetros (como el radio) en la forma en lugar de escalarlo. Lo mismo es cierto para los cuerpos kinematic/rigid/static, porque su escala afecta la escala de la forma.

Ahora crea un script para el personaje, el que se uso como un ejemplo arriba debería funcionar como base.

Finalmente, instancia esa escena de personaje en el tilemap, y haz que la escena del mapa sea la principal, asi corre cuando tocamos play.



Moviendo el Kinematic character

Ve nuevamente a la escena del personaje, y abre el script, la magia empieza ahora! Kinematic body no hará nada por defecto, pero tiene una función realmente util llamada *KinematicBody2D.move()*. Esta función toma un *Vector2* como argumento, e intenta aplicar ese movimiento al kinematic body. Si una colisión sucede, se detiene justo en el momento de la colisión.

Entonces, vamos a mover nuestro sprite hacia abajo hasta que toque el piso:

```
extends KinematicBody2D

func _fixed_process(delta):
    move( Vector2(0,1) ) #mover abajo 1 pixel por frame de física

func _ready():
    set_fixed_process(true)
```

El resultado es que el personaje se va a mover, pero se detendrá justo cuando toque el piso. Copado, eh?

El siguiente paso será agregar gravedad a la mezcla, de esta forma se comporta un poco mas como un personaje de juego:

```
extends KinematicBody2D

const GRAVITY = 200.0
var velocity = Vector2()

func _fixed_process(delta):

    velocity.y += delta * GRAVITY

    var motion = velocity * delta
    move( motion )
```

```
func _ready():
    set_fixed_process(true)
```

Ahora el personaje cae suavemente. Hagamos que camine hacia los costados, izquierda y derecha cuando se tocan las teclas direccionales. Recuerda que los valores siendo usados (al menos para la velocidad) son pixels/segundo.

Esto agrega soporte simple para caminar cuando se presiona izquierda y derecha:

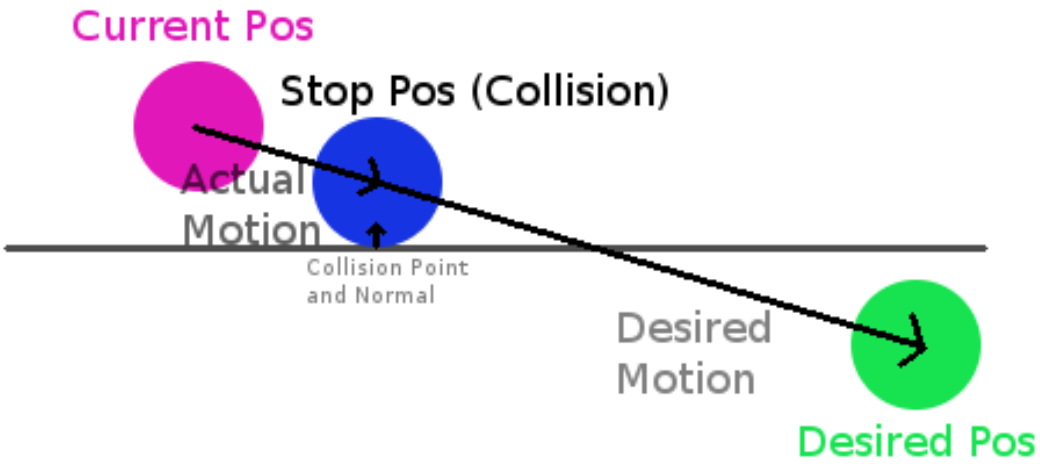
```
extends KinematicBody2D

const GRAVITY = 200.0
const WALK_SPEED = 200

var velocity = Vector2()

func _fixed_process(delta):

    velocity.y += delta * GRAVITY

    if (Input.is_action_pressed("ui_left")):
        velocity.x = -WALK_SPEED
    elif (Input.is_action_pressed("ui_right")):
        velocity.x =  WALK_SPEED
    else:
        velocity.x = 0

    var motion = velocity * delta
    move(motion)

func _ready():
    set_fixed_process(true)
```

Y dale una prueba.

Problema?

Y... no funciona muy bien. Si vas hacia la izquierda contra una pared, se queda trancado a no ser que sueltes la flecha. Una vez en el piso, también se queda trancado y no camina. Que sucede??

La respuesta es, lo que parece que debería ser simple, no es tan simple en realidad. Si el movimiento no puede ser completado, el personaje dejara de moverse. Es así de sencillo. Este diagrama debería ilustrar mejor que esta sucediendo:



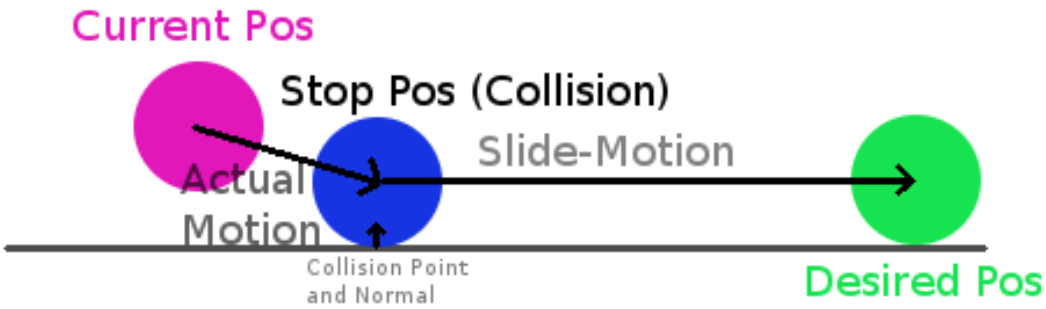
Básicamente, el movimiento vectorial deseado nunca se completara porque golpea el piso y la pared demasiado temprano en la trayectoria de movimiento y eso hace que se detenga allí. Recuerda que aunque el personaje esta en el piso, la gravedad siempre esta empujando el vector movimiento hacia abajo.

Solución!

La solución? Esta situación se resuelve al "deslizar" por la normal de la colisión. KinematicBody2D provee dos funciones útiles:

- *KinematicBody2D.is_colliding()*
- *KinematicBody2D.get_collision_normal()*

Así que lo que queremos hacer es esto:



Cuando colisiona, la función `move()` retorna el "remanente" del vector movimiento. Esto significa, que si el vector de movimiento es 40 pixels, pero la colisión sucedió a los 10 pixeles, el mismo vector pero con 30 pixels de largo es retornado.

La forma correcta de resolver el movimiento es, entonces, deslizarse por la normal de esta forma:

```
func _fixed_process(delta):

    velocity.y += delta * GRAVITY
    if (Input.is_action_pressed("ui_left")):
        velocity.x = -WALK_SPEED
    elif (Input.is_action_pressed("ui_right")):
        velocity.x =  WALK_SPEED
    else:
        velocity.x = 0

    var motion = velocity * delta
    motion = move(motion)

    if (is_colliding()):
        var n = get_collision_normal()
        motion = n.slide(motion)
        velocity = n.slide(velocity)
        move(motion)


func _ready():
    set_fixed_process(true)
```

Observa que no solo el movimiento ha sido modificado pero también la velocidad. Esto tiene sentido ya que ayuda a mantener la nueva dirección también.

La normal también puede ser usada para detectar que el personaje esta en el piso, al chequear el ángulo. Si la normal apunta hacia arriba (o al menos, con cierto margen), se puede determinar que el personaje esta allí.

Una demo mas completa puede encontrarse en el zip de demos distribuidos con el motor, o en https://github.com/

godotengine/godot/tree/master/demos/2d/kinematic_char.

3.3.3 Ray-casting (Emision de rayos)

Introducción

Una de las tareas mas comunes en el desarrollo de juegos es emitir un rayo (o un objeto con una forma personalizada) y chequear que golpea. Esto permite que comportamientos complejos, AI, etc. tengan lugar. Este tutorial va a explicar como hacer esto en 2D y 3D.

Godot guarda toda la información de bajo nivel del juego en servidores, mientras que la escena es solo una interfaz. Como tal, ray casting en general es una tarea de bajo-nivel. Para raycasts simples, nodos como *RayCast* y *RayCast2D* funcionaran, ya que regresaran en cada frame cual es el resultado del raycast.

Muchas veces, sin embargo, ray-casting necesita ser un proceso mas interactivo por lo que una forma de hacer esto por código debe existir.

Espacio

En el mundo físico, Godot guarda todas las colisiones de bajo nivel e información física en un *space*. El actual espacio 2D (para física 2D) puede ser obtenido al llamar *CanvasItem.get_world_2d().get_space()*. Para 3D, es *Spatial.get_world().get_space()*.

El espacio resultante *RID* puede ser usado en *PhysicsServer* y *Physics2DServer* para 3D y 2D respectivamente.

Accediendo al espacio

La física de Godot corre por defecto en el mismo thread que la lógica del juego, pero puede ser ajustada para correr en un thread separado para funcionar mas eficientemente. Debido a esto, el único momento donde acceder al espacio es seguro es durante la llamada de retorno *Node._fixed_process()*. Accediéndola desde fuera de esta función puede resultar en un error debido a que el espacio esta *locked*(bloqueado).

Para realizar cónsulas en espacio físico, la clase *Physics2DDirectSpaceState* y ref:*PhysicsDirectSpaceState <class_PhysicsDirectSpaceState>* deben ser usadas.

En código, para 2D spacestate, este código debe ser usado:

```
func _fixed_process(delta):
    var space_rid = get_world_2d().get_space()
    var space_state = Physics2DServer.space_get_direct_state(space_rid)
```

Por supuesto, hay un atajo mas simple:

```
func _fixed_process(delta):
    var space_state = get_world_2d().get_direct_space_state()
```

Para 3D:

```
func _fixed_process(delta):
    var space_state = get_world().get_direct_space_state()
```

Consulta Raycast

Para realizar una consulta raycast 2D, el método *Physics2DDirectSpaceState.intersect_ray()* debe ser usado, por ejemplo:

```
func _fixed_process(delta):
    var space_state = get_world().get_direct_space_state()
    # usa coordenadas globales, no locales al nodo
    var result = space_state.intersect_ray( Vector2(0,0), Vector2(50,100) )
```

El resultado es un diccionario. Si el rayo no pego contra nada, el diccionario estará vacío. Si pego algo entonces tendrá la información de colisión:

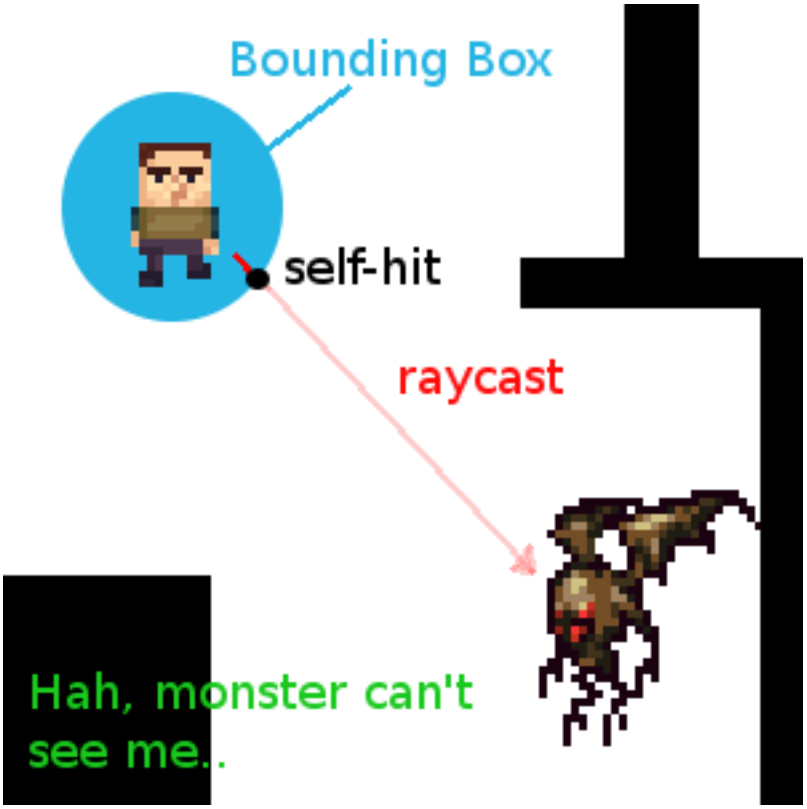
```
if (not result.empty()):
    print("Golpe en el punto: ",result.position)
```

El diccionario de resultado de colisión, cuando pega en algo, tiene este formato:

```
{
   position:Vector2 # punto en el world space para la colisión
   normal:Vector2 # normal en el world space para colisión
   collider:Object # Objeto colisionado o null (si no esta asociado)
   collider_id:ObjectID # Objeto contra el que colisiono
   rid:RID # RID contra el que colisiono
   shape:int # indice de forma del colisionador
   metadata:Variant() # metadata del colisionador
}

# En caso de 3D, Vector3 es regresado.
```

Excepciones de colisiones

Es un caso muy común el intentar emitir un rayo desde un personaje u otra escena del juego para tratar de inferir propiedades de el mundo a su al rededor. El problema con esto es que el mismo personaje tiene un colisionador, por lo que el rayo nunca deja el origen (se mantendrá golpeando su propio colisionador), como queda en evidencia en la siguiente imagen.



Para evitar la intersección propia, la función intersect_ray() puede tomar un tercer parámetro opcional que es un arreglo de excepciones. Este es un ejemplo de como usarlo desde un KinematicBody2D o cualquier otro nodo basado en CollisionObject:

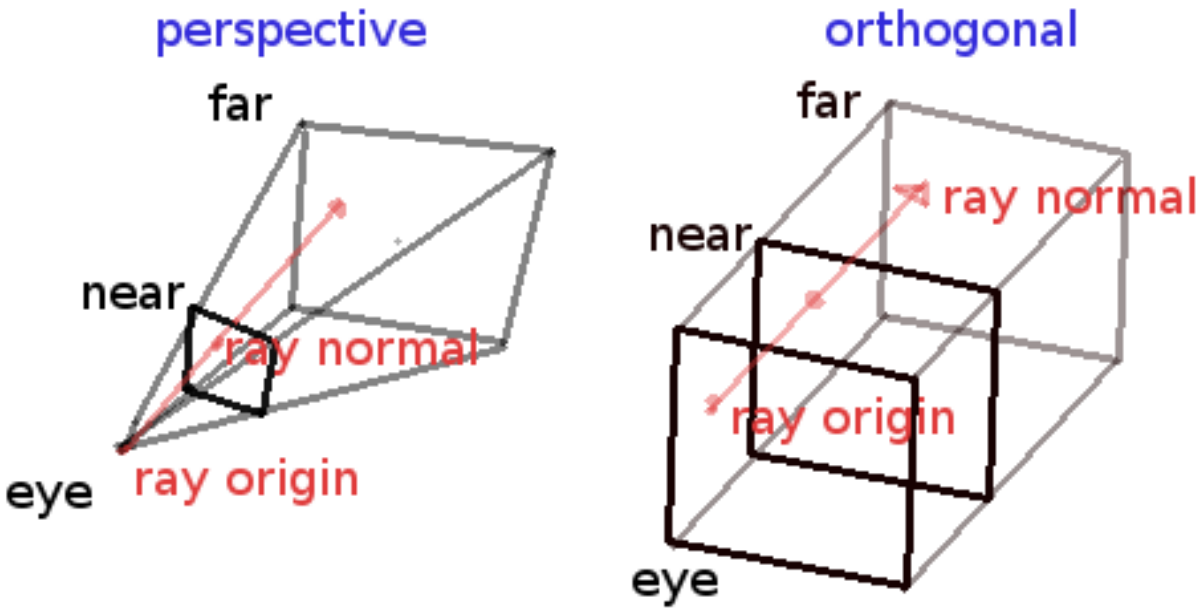
```
extends KinematicBody2D

func _fixed_process(delta):
    var space_state = get_world().get_direct_space_state()
    var result = space_state.intersect_ray( get_global_pos(), enemy_pos, [ self ] )
```

El argumento extra es una lista de excepciones, pueden ser objetos o RIDs.

Ray casting 3D desde pantalla.

Emitir un rayo desde pantalla a un espacio físico 3D es útil para tomar objetos. No hay mucha necesidad de hacer esto porque *CollisionObject* tiene una señal "input_event" que te permite saber cuando es pinchada, pero en caso que haya deseo de hacerlo manualmente, aquí esta como.

Para emitir un rayo desde la pantalla, el nodo *Camera* es necesario. La cámara puede estar en dos modos de proyección, perspectiva y ortogonal. Por este motivo, ambos el origen del rayo y la dirección deben obtenerse. (el origen cambia en ortogonal, mientras que la dirección cambia en perspectiva):



Para obtenerlo usando una cámara, el siguiente código puede ser usado:

```
const ray_length = 1000

func _input(ev):
    if ev.type==InputEvent.MOUSE_BUTTON and ev.pressed and ev.button_index==1:

          var camera = get_node("camera")
          var from = camera.project_ray_origin(ev.pos)
          var to = from + camera.project_ray_normal(ev.pos) * ray_length
```

Por supuesto, recuerda que durante `_input()`, el espacio puede estar locked, así que guarda tu consulta para `_fixed_process()`.

CAPÍTULO 4

Tutoriales 3D

4.1 Gráficos

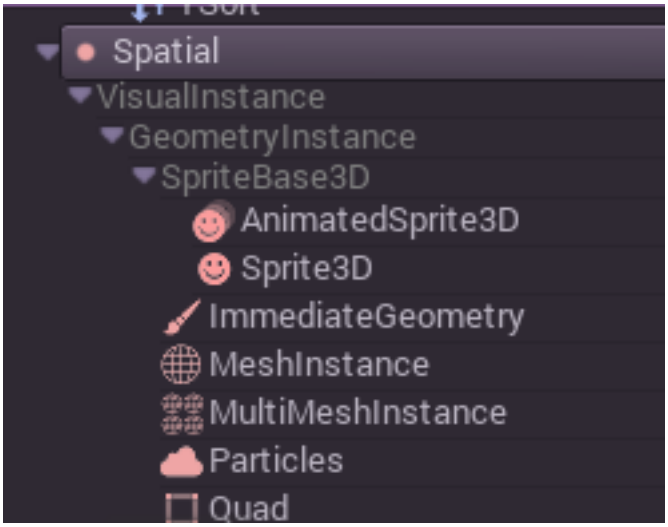
4.1.1 Introducción a 3D

Crear un juego 3D puede ser desafiante. La coordenada Z extra hace que muchas de las técnicas comunes que ayudan a hacer juegos 2D simplemente no sirvan mas. Para ayudar en esta transición, vale mencionar que Godot usa APIs muy similares para 2D y 3D. La mayoría de los nodos son iguales y están presentes tanto en versiones 2D como 3D. De hecho, es recomendable chequear el tutorial del platformer 3D, o los tutoriales de kinematic 3D, los cuales son casi idénticos a su contraparte 2D.

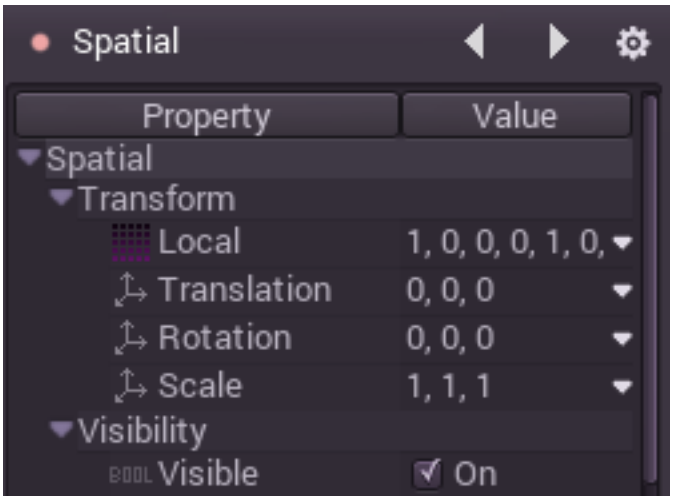
En 3D, la matemática es un poco mas compleja que en 2D, por lo que chequear :ref:‘doc_vector_math‘(que fue creado especialmente para desarrolladores de juegos, no matemáticos ni ingenieros) ayudara a cementar el camino hacia el desarrollo de juegos 3D eficientes.

Nodo Spatial

Node2D es el nodo base para 2D. *Control* es el nodo base para todo lo relacionado con GUI. Siguiendo este razonamiento, el motor 3D usa el nodo *Spatial* para todo lo que sea 3D.



Los nodos Spatial tienen transformaciones locales, las cuales son relativas al nodo padre (en la medida que el nodo padre también sea o **herede** el tipo Spatial). Esta transformación puede ser accedida como un *Transform* 4x3, o como 3 miembros *Vector3* representando posición, rotación Euler (ángulos x, y, z) y escala.



Contenido 3D

A diferencia de 2D, donde cargar contenido de imágenes y dibujar es sencillo, en 3D se vuelve mas difícil. El contenido necesita ser creado con herramientas 3D especiales (generalmente referidas como DCCs) y exportadas hacia un formato de archivo de intercambio para ser importadas en Godot (los formatos 3D no son tan estandarizados como las imágenes).

Modelos creados por DCC

Hay dos pipelines (conductos) para importar modelos 3D en Godot. La primera y mas común es a través del importador *Importando escenas 3D*, el cual permite importar escenas enteras (como lucen en el DCC), incluyendo animación, skeletal rigs, blend shapes, etc.

El segundo pipeline es a través del importador *Importando mallas 3D*. Este segundo método permite importar archivos simples .OBJ como recursos mesh (malla), los cuales luego pueden ser puestos dentro de un nodo *MeshInstance* para visualización.

Geometría generada

Es posible crear geometría personalizada usando el recurso *Mesh* directamente, solo crea tus arreglos y usa la función *Mesh.add_surface()*, que provee una API mas directa y ayudantes para indexar, generar normales, tangentes, etc.

En todo caso, este método esta destinado para generar geometráa estática (modelos que no serán actualizados a menudo), ya que crear arreglos de vértices y enviarlos a la API 3D tiene un costo de rendimiento significativo.

Geometría Inmediata

Si, por otro lado, hay un requerimiento de generar geometría simple que va a ser actualizada a menudo, Godot provee un nodo especial, *ImmediateGeometry* que provee una API immediate-mode de estilo OpenGL 1.x para crear puntos, líneas triángulos, etc.

2D en 3D

Aunque Godot provee un motor 2D poderoso, muchos tipos de juegos usan 2D en un entorno 3D. Al usar una cámara fija (ortogonal o perspectiva) que no rota, nodos como *Sprite3D* y *AnimatedSprite3D* pueden ser usados para crear juegos 2D que toman ventaja de la mezcla con fondos 3D, parallax mas realista, efectos de luz y sombra, etc.

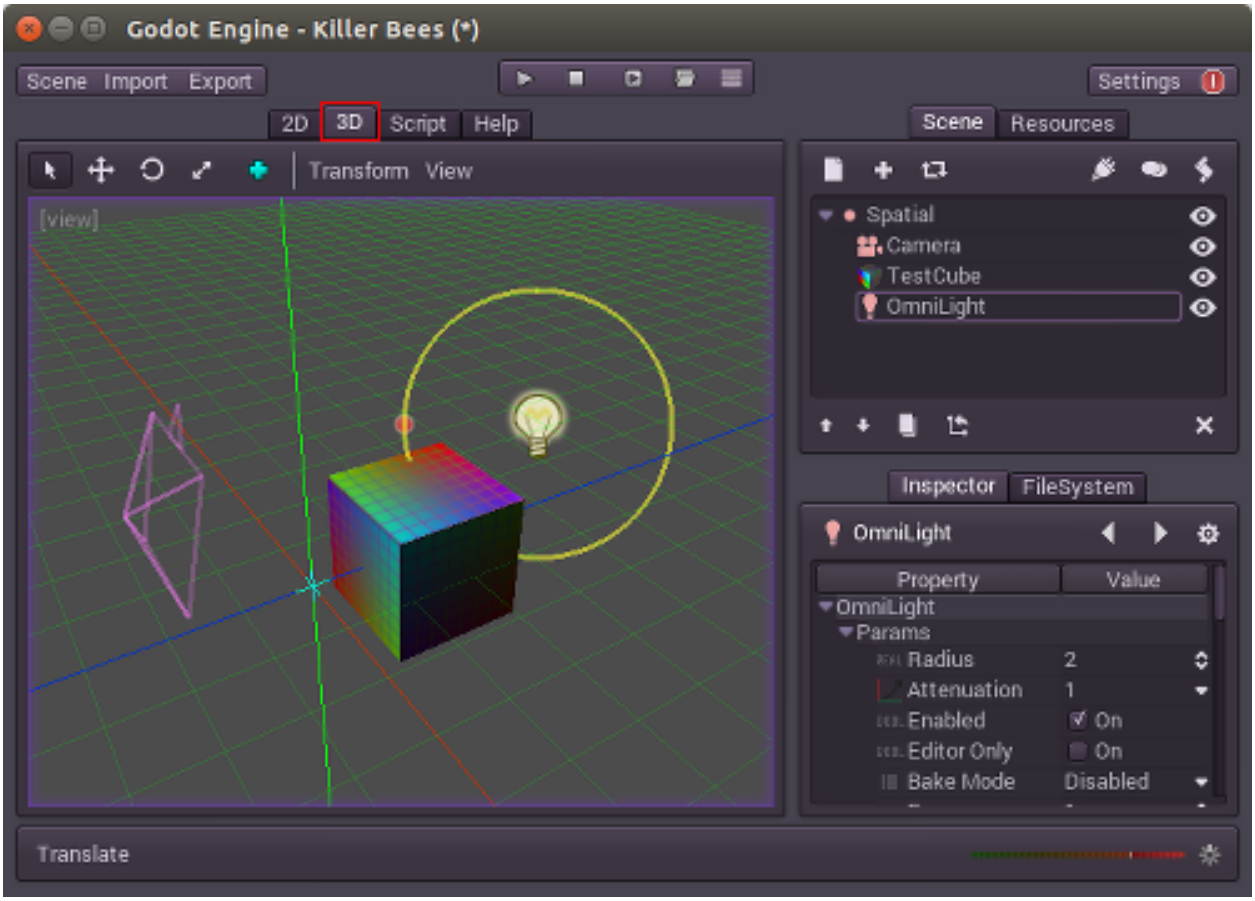
La desventaja es, por supuesto, esa complejidad agregada y un rendimiento reducido en comparación con 2D puro, así como la falta de referencia de trabajar en pixels.

Entorno

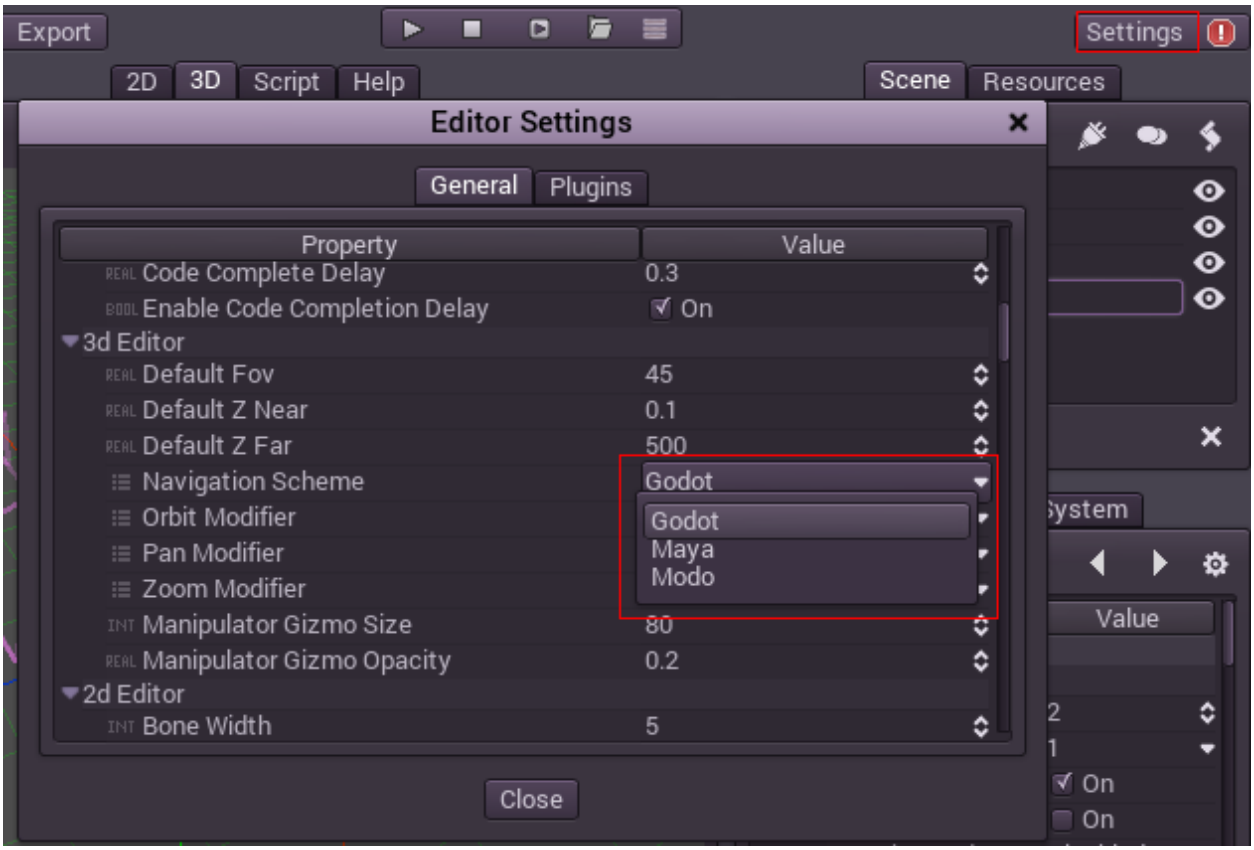
Mas allá de editar una escena, es a menudo común editar el entorno. Godot provee un nodo *WorldEnvironment* que permite cambiar el color de fondo, modo (como en, poner un skybox), y aplicar varios tipos de efectos post-processing incorporados. Los entornos también pueden ser sobrescritos en la Cámara.

Viewport 3D

Editar escenas 3D es hecho en la pestaña 3D. Esta pestaña puede ser seleccionada manualmente, pero será automáticamente habilitada cuando se selecciona un nodo Spatial.



Los controles por defectos para la navegación de escena 3D es similar a Blender (apuntando a tener algún tipo de consistencia en el pipeline de software libre..), pero hay opciones incluidas para personalizar los botones de mouse y el comportamiento para ser similar a otras herramientas en la Configuración del Editor:



Sistema de coordenadas

Godot usa el sistema métrico para todo. La física 3D y otras áreas estas ajustadas para esto, así que intentar usar una escala diferente suele ser una mala idea (a no ser que sepas lo que estas haciendo).

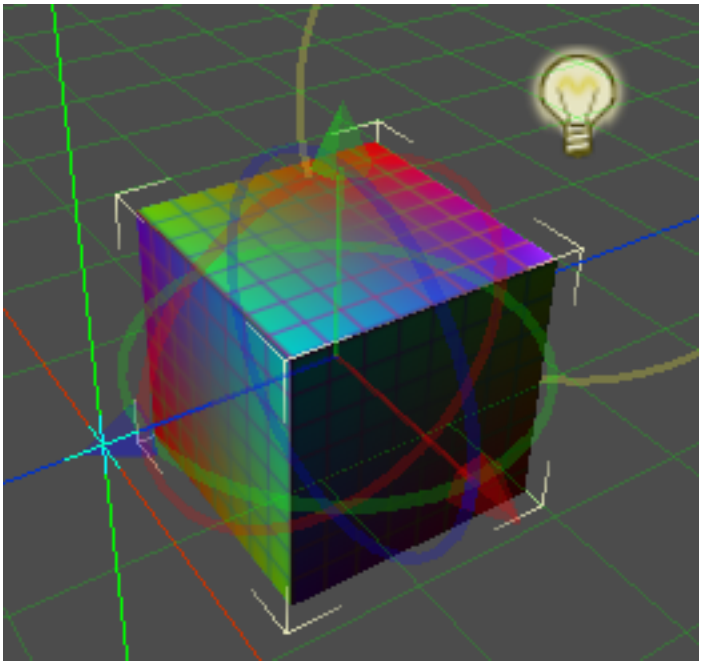
Cuando se trabaja con assets 3D, siempre es mejor trabajar en la escala correcta (ajusta tu DCC a métrico). Godot permite escalar luego de importar y, mientras que funciona en la mayoría de los casos, en situaciones raras puede introducir problemas de precisión de punto flotante (y por lo tanto, glitches y artefactos) en áreas delicadas como renderizacion o física. Asique, asegurate que tus artistas siempre trabajen en la escala correcta!

La coordenada Y es usada para "arriba", aunque en la mayoría de los objetos que requieren alineación (como luces, cámaras, colisionadores de capsula, vehículos, etc.), el eje Z es usado como la dirección "hacia donde apuntan". Esta convención a grandes líneas implica que:

- **X** son los lados
- **Y** es arriba/abajo
- **Z** es adelante/atras

Espacio y manipulación de gizmos

Mover objetos en la vista 3D es hecho a través de gizmos de manipulación. Cada eje es representado por un color: Rojo, Verde, Azul representa X, Y, Z respectivamente. Esta convención se aplica a la grilla y otros gizmos también (ademas en el lenguaje de shader, orden de componentes para Vector3, Color, etc.).



Algunos a atajos de teclado útiles:

- Para aplicar snap a movimiento o rotación, presiona la tecla “s” mientras mueves, escalas o rotas.
- Para centrar la vista en el objeto seleccionado, presiona la tecla “f”.

Menú de vista

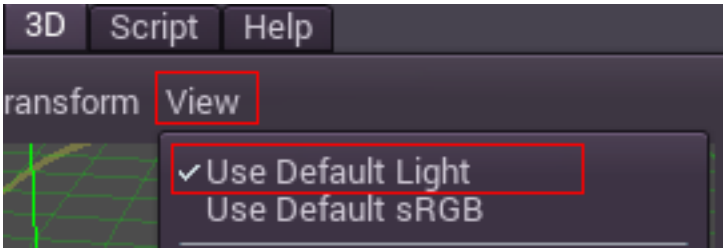
Las opciones de vista con controladas por el menú vista. Presta atención a este pequeño menú dentro de la ventana porque a menudo es pasado por alto!

Iluminación por defecto

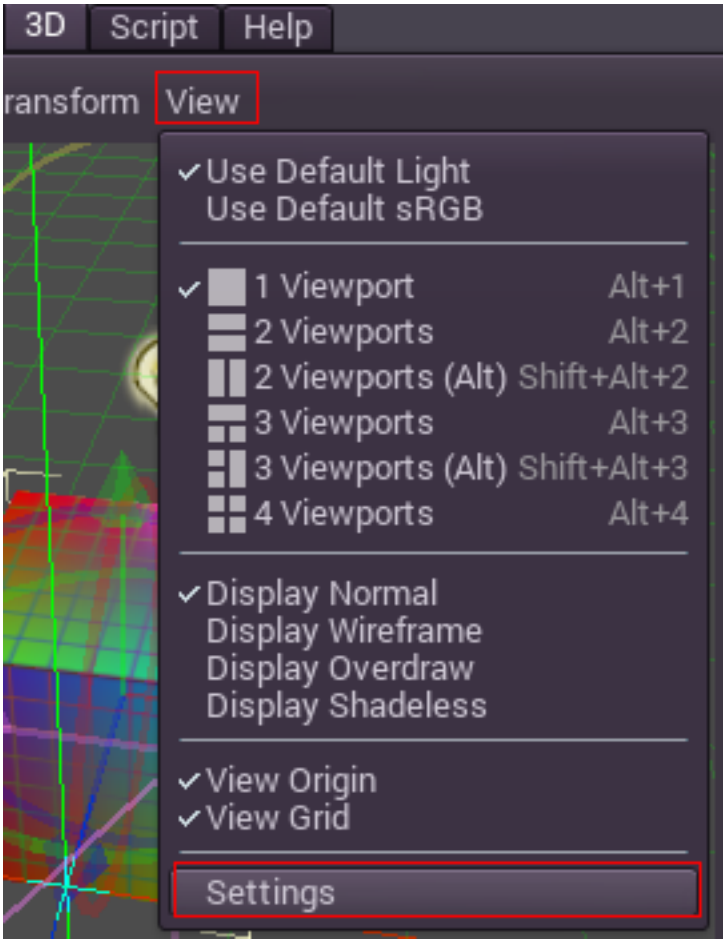
La vista 3D tiene algunas opciones por defecto en iluminación:

- Hay una luz direccional que vuelve los objetos visibles mientras se esta editando por defecto. No es visible cuando se corre el juego.
- Hay una luz sutil de entorno por defecto para iluminar lugares donde no llega luz y que permanezcan visibles. Tampoco es visible cuando se corre el juego (y cuando la luz por defecto esta apagada).

Estas pueden ser apagadas al alternar la opción "Usar luz por defecto":



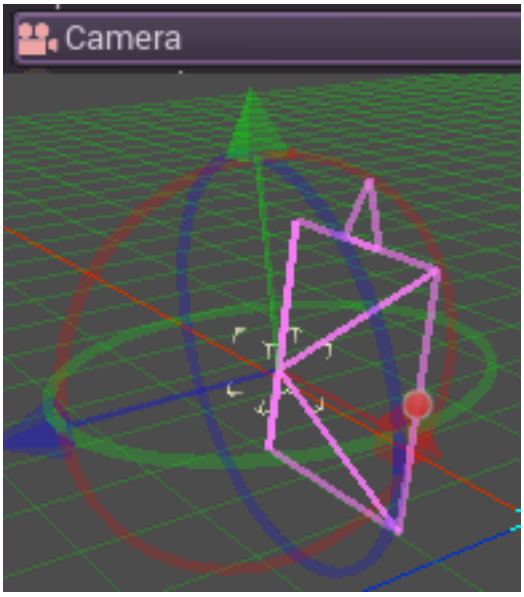
Personalizar esto (y otros valores de vista por defecto) también es posible desde el menú de configuración:



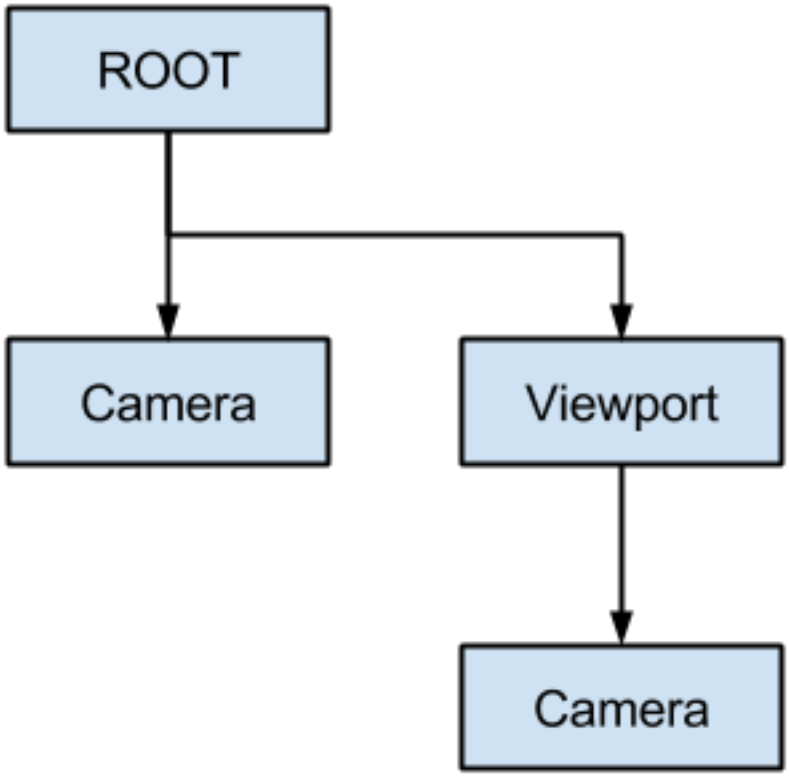
Que abre esta ventana, permitiendo personalizar el color de la luz ambiente y la dirección de la luz por defecto

Camarás

No importa cuantos objetos son ubicados en el espacio 3D, nada va a ser mostrado a no ser que *Camera* también este agregado a la escena. Las cámaras pueden trabajar ya sea en proyección ortogonal o perspectiva:



Las cámaras están asociadas y solo despliegan hacia un viewport padre o abuelo. Ya que la raíz del árbol de escena es un viewport, las cámaras van a mostrarse en el por defecto, pero si se desean sub_viewports (tanto como render targets o picture-in-picture), necesitan sus propias cámaras hijo para desplegar.



Cuando se manejen muchas cámaras, las siguientes reglas se siguen para cada viewport:

- Si no hay cámaras presentes en el árbol de escena, la primera que entre se volverá la cámara activa. Las siguientes cámaras que entren a la escena serán ignoradas (a no ser que se ajusten a *current*).

- Si una cámara tiene la propiedad *current* habilitada, será usada no importa si existen otras cámaras en la escena. Si la propiedad se ajusta, se volverá activa, reemplazando la cámara previa.
- Si una cámara activa deja el árbol de escena, la primer cámara en el orden de árbol tomara su lugar.

Luces

No hay limitacion en el numero de luces ni de los tipos de luces en Godot. Tantas como se quieran pueden ser agregadas (mientras que el rendimiento lo permita). Los Shadow maps son, sin embargo, limitados. Cuanto mas se usan, menor es la calidad total.

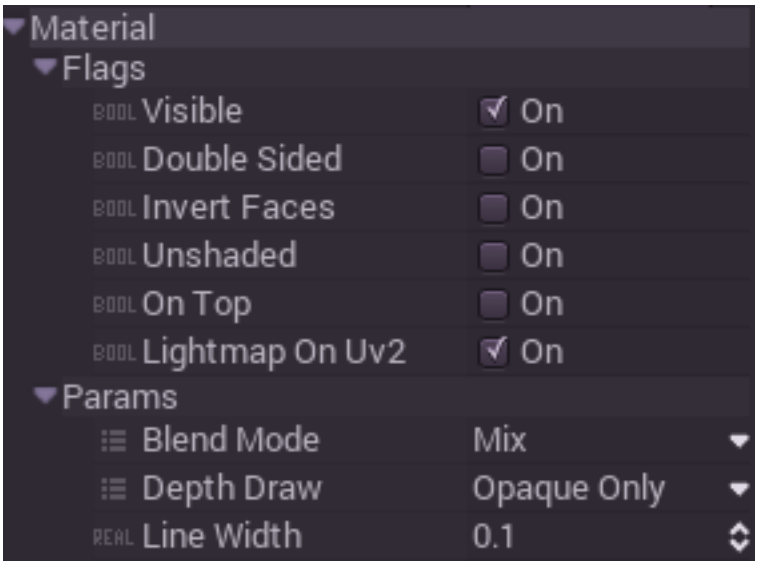
Es posible usar doc_light_baking, para evitar usar una gran cantidad de luces en tiempo real y mejorar el rendimiento.

4.1.2 Materiales

Introducción

Los materiales pueden ser aplicados a la mayoría de los objetos 3D, son básicamente una descripción de como la luz reacciona con ese objeto. Hay muchos tipos de materiales, pero los principales son *FixedMaterial* y *ShaderMaterial*. Existen tutoriales para cada uno de ellos: *Fixed Materials (Materiales Fijos)* and *Shader materials (Materiales Shader)*.

Este tutorial es sobre las propiedades básicas compartidas entre ellos.



Flags (banderas)

Los materiales, no importa del tipo que sean, tienen un grupo de flags asociados. Cada una tiene un uso diferente y será explicado a continuación.

Visible

Alterna si el material es visible. Si no esta marcado, el objeto no será mostrado.

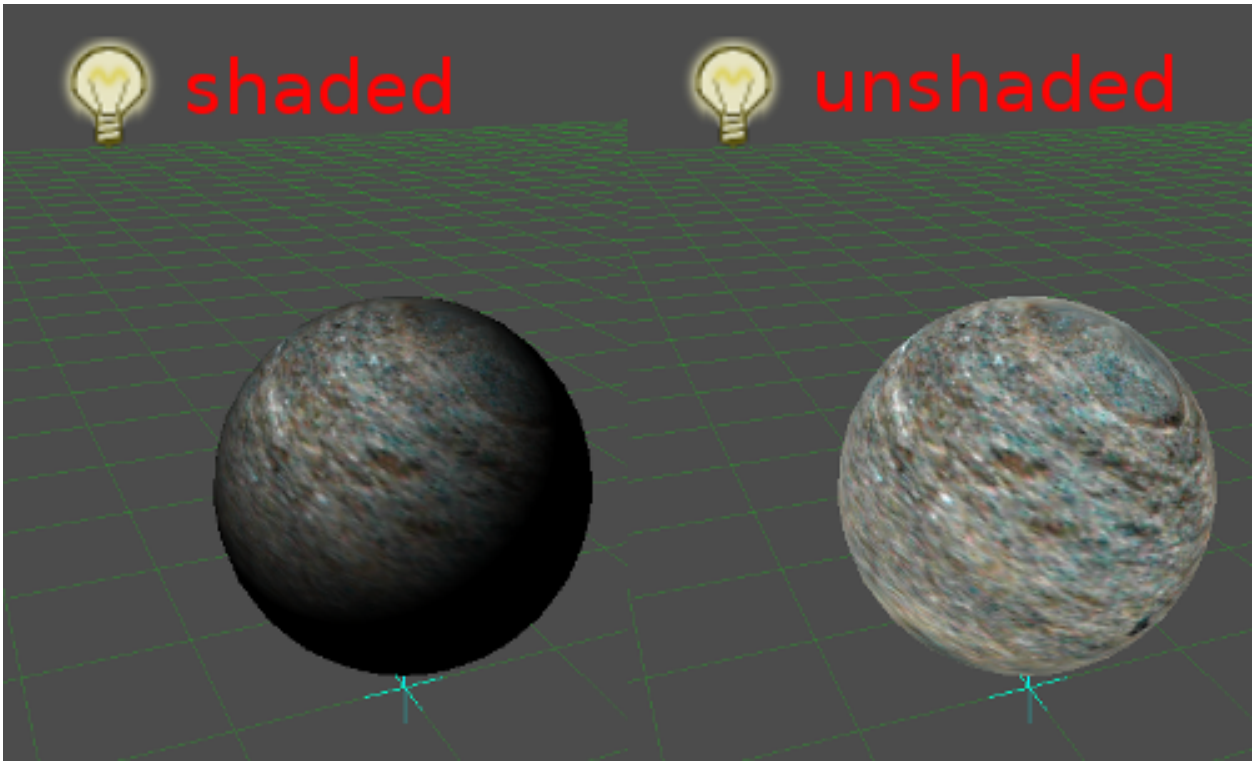
Double sided & invert faces

Godot por defecto solo muestra las caras de la geometría (triángulos) cuando están frente a la cámara. Para hacer esto necesita que estén en un orden de agujas de reloj. Esto ahorra un montón de trabajo a la GPU al asegurarse que los triángulos no visibles no sean dibujados.

Algunos objetos planos pueden necesitar ser dibujados todo el tiempo sin embargo, para esto el flag "double sided" se asegurara que no importa hacia donde este mirando, el triángulo siempre será dibujado. También es posible invertir este chequeo y dibujar caras en un orden contrario a las agujas de reloj, aunque no es muy útil excepto para algunos casos (como dibujar outlines).

Unshaded

Los objetos siempre son negros a no ser que la luz los afecta, y el shading (sombreado) cambia de acuerdo al tipo y dirección de las luces. Cuando este flag es prendido, el color diffuse (difuso) es mostrado tal cual aparece en la textura o parámetro.



On top

Cuando este flag esta encendido, el objeto será dibujado luego que todo lo demás ha sido dibujado y sin una prueba de profundidad. Esto en general solo es útil para efectos de HUD o gizmos.

Ligthmap on UV2

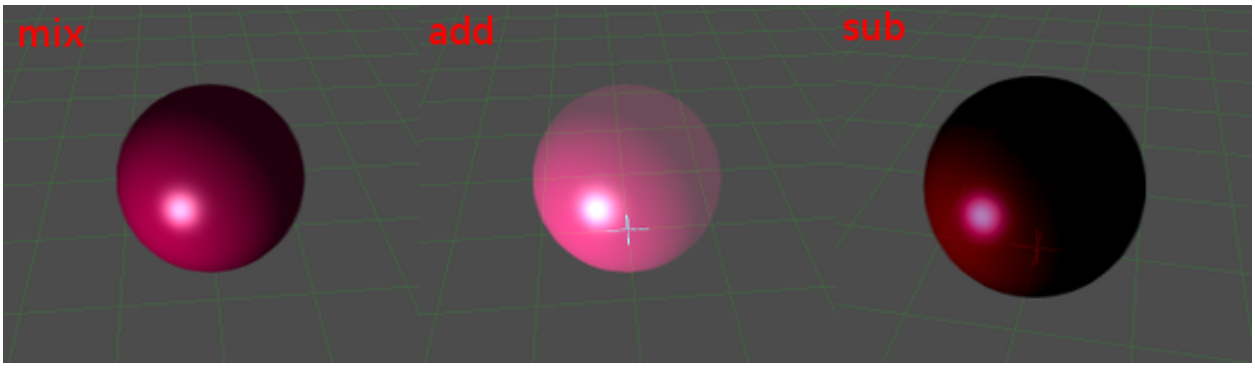
Cuando usas lightmapping (ve el tutorial doc_light_baking), esta opción determina que el lightmap debería ser accedido en el arreglo UV2 en lugar del UV regular.

Parámetros

Algunos párametros también existen para controlar el dibujado y mezclado:

Blend mode

Los objetos son usualmente mezclados en modo Mix. Otros modos de blend (Add and Sub) existen para casos especiales (usualmente efectos de partículas, rayos de luz, etc.) pero se pueden ajustar los materiales a ellos:



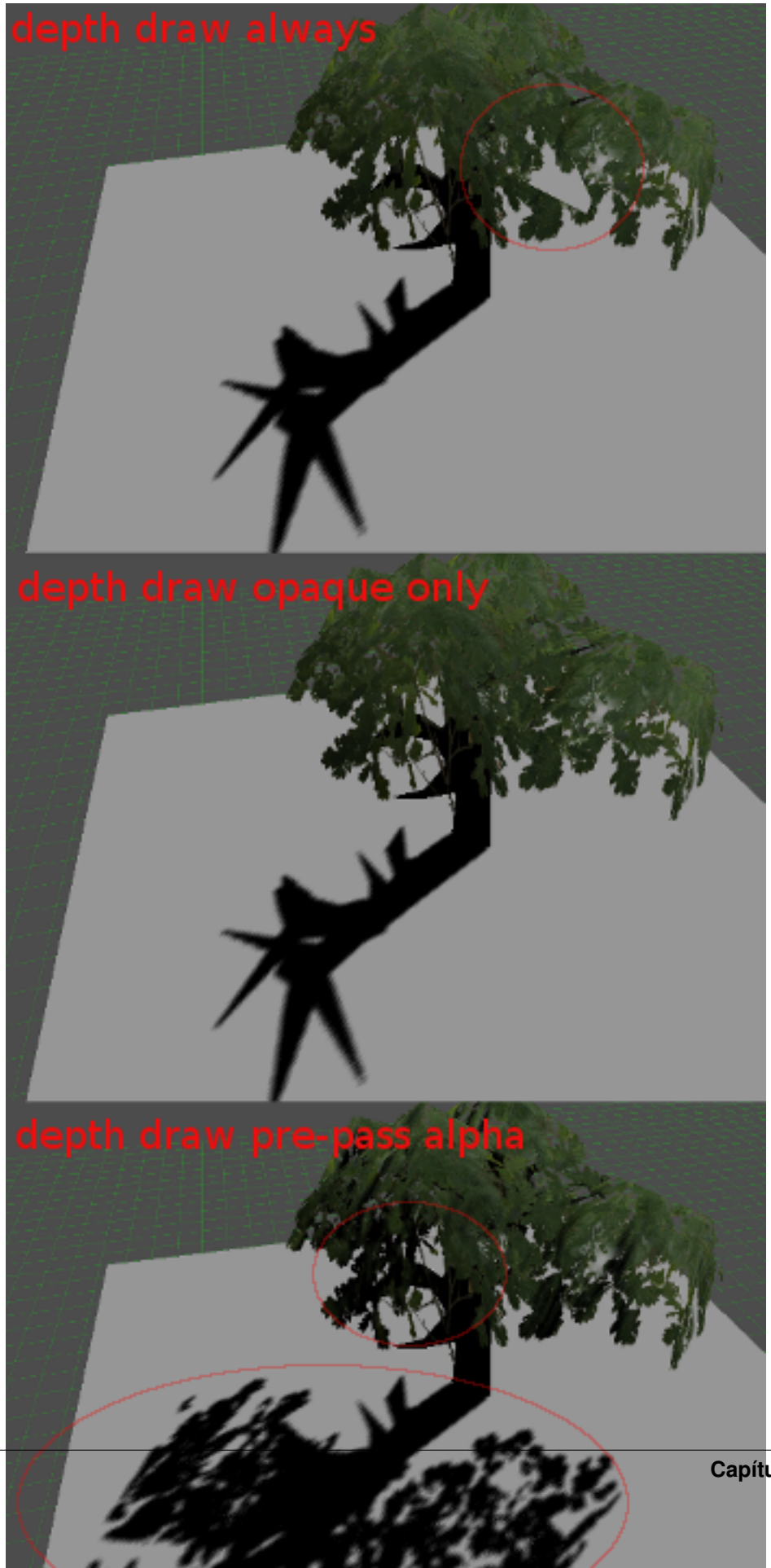
Line width

Cuando dibujas líneas, el tamaño de ellas puede ser ajustado aquí para cada material.

Depth draw mode

Este es un ajuste difícil pero muy útil. Por defecto, los objetos opacos son dibujados usando el depth buffer y los objetos translucientes no (pero son ordenados por profundidad). Este comportamiento puede ser cambiado aquí. Las opciones son:

- **Always**: Dibuja objetos con profundidad siempre, aun aquellos con alpha. Esto a menudo resulta en glitches como el de la primer imagen (motivo por el cual no es el valor por defecto)
- **Opaque Only**: Dibuja objetos con profundidad solo cuando son opacos, y no ajusta la profundidad para alpha. Este es el valor por defecto porque es rápido, pero no es el ajuste mas correcto. Los objetos con transparencia que se auto-intersectan siempre lucirán mal, especialmente aquellos que mezclan áreas opacas y transparentes, como pasto, hojas de árboles, etc. Los objetos con transparencia tampoco pueden emitir sombras, esto es evidente en la segunda imagen.
- **Alpha Pre-Pass**: Lo mismo que el anterior, pero una pasada de profundidad es realizada para las áreas opacas de los objetos con transparencia. Esto hace que los objetos con transparencia luzcan mucho mejor. En la tercer imagen es evidente como las hojas emiten sombras entre ellas y hacia el piso. Este ajuste es apagado por defecto ya que, mientras en PC no es muy costoso, los dispositivos móviles sufren un montón cuando es habilitado, así que úsalo con cuidado.
- **Never**: Nunca usar el buffer de profundidad para este material. Esto es mas útil en combinación con la bandera "On Top" explicada mas arriba.


depth draw always
depth draw opaque only
depth draw pre-pass alpha

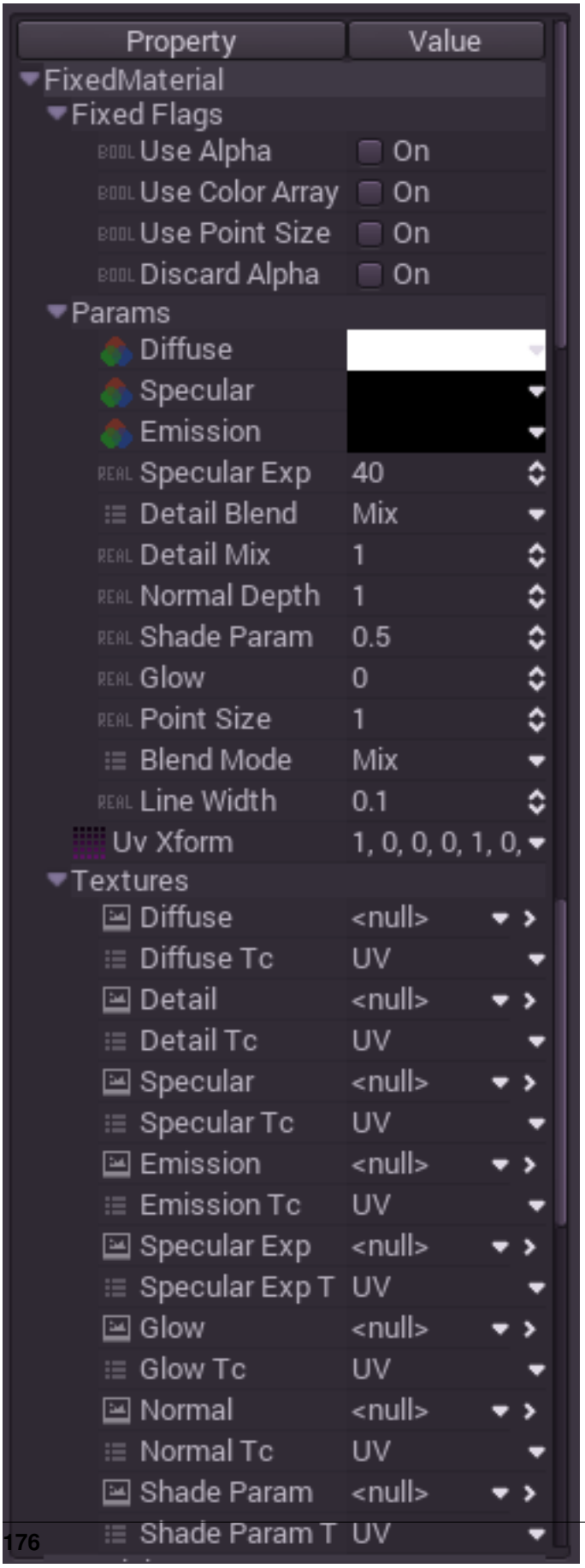
4.1.3 Fixed Materials (Materiales Fijos)

Introducción

Fixed materials (originalmente Fixed Pipeline Materials) son el tipo mas común de materiales, usando las opciones mas comunes de materiales encontradas en DCCs 3D (Como Maya, 3DS Max, o Blender). La gran ventaja de usarlos es que los artistas 3D están muy familiarizados con este diseño. También permiten probar diferentes cosas rápidamente sin la necesidad de escribir shaders. Fixed Materials heredan desde *Material*, que también tiene varias opciones. Si no lo has leido antes, es recomendable leer el tutorial *Materiales*.

Opciones

Aquí hay una lista de todas las opciones disponibles para fixed materials:


Property
Value
FixedMaterial
Fixed Flags
Use Alpha
On
Use Color Array
On
Use Point Size
On
Discard Alpha
On
Params
Diffuse
Specular
Emission
Specular Exp
40
Detail Blend
Mix
Detail Mix
1
Normal Depth
1
Shade Param
0.5
Glow
0
Point Size
1
Blend Mode
Mix
Line Width
0.1
Uv Xform
1, 0, 0, 0, 1, 0,
Textures
Diffuse
<null>
Diffuse Tc
UV
Detail
<null>
Detail Tc
UV
Specular
<null>
Specular Tc
UV
Emission
<null>
Emission Tc
UV
Specular Exp
<null>
Specular Exp T
UV
Glow
<null>
Glow Tc
UV
Normal
<null>
Normal Tc
UV
Shade Param
<null>
Shade Param T
UV

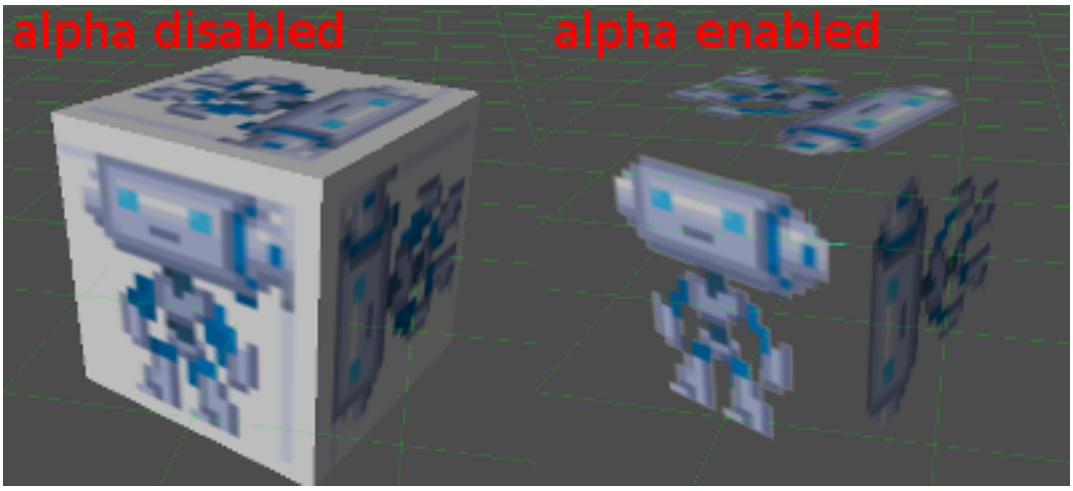
A partir de este punto, toda opción se explicara en detalle:

Flags fijas

Este es un conjunto de flags (banderas) que controlan aspectos generales del material.

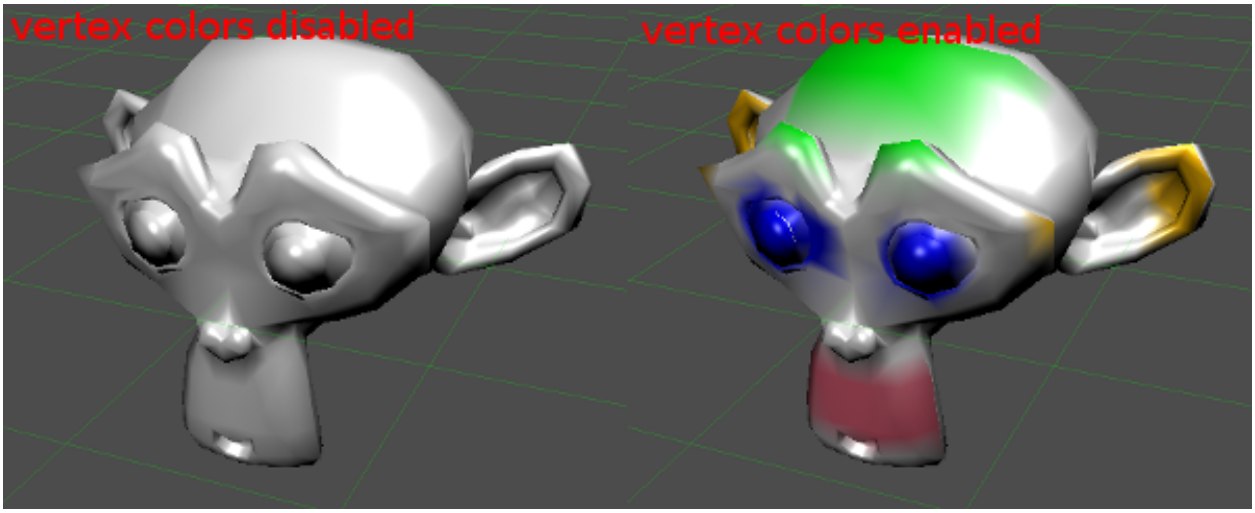
Use alpha

Este flag necesita estar activo para que materiales transparentes se mezclen con lo que esta detrás, de otra forma siempre se desplegara de forma opaca. No habilites este flag a no ser que el material realmente lo necesite, porque puede afectar severamente la performance y calidad. Los materiales con transparencia no proyectaran sombras (a no ser que contengan áreas opacas y que el hint "opaque pre-pass" este habilitado, ve el tutorial *Materiales* para mas información)



Use vertex colors

Pintar por vertex color es una técnica muy común de agregar detalle a la geometría. Todos los DDCs 3D soportan esto, y puede que hasta soporten baking occlusion. Godot permite que esta información sea usada en el fixed material al modular el color diffuse cuando se habilita.



Point size

Point size es usado para ajustar el tamaño del punto (en pixels) para cuando se renderizan puntos. Esta característica es mas que nada usada para herramientas y HUDs.

Discard alpha

Cuando alpha esta habilitado (ve arriba) los pixels invisibles son mezclados con lo que hay detrás de ellos. En algunas combinaciones (de usar alpha para renderizar profundidad) puede suceder que pixels invisibles cubran otros objetos.

Si este es el caso, habilita esta opción para el material. Esta opción es a menudo usada en combinación con el hint "opaque pre-pass" (ve el tutorial *Materiales* para mas información).

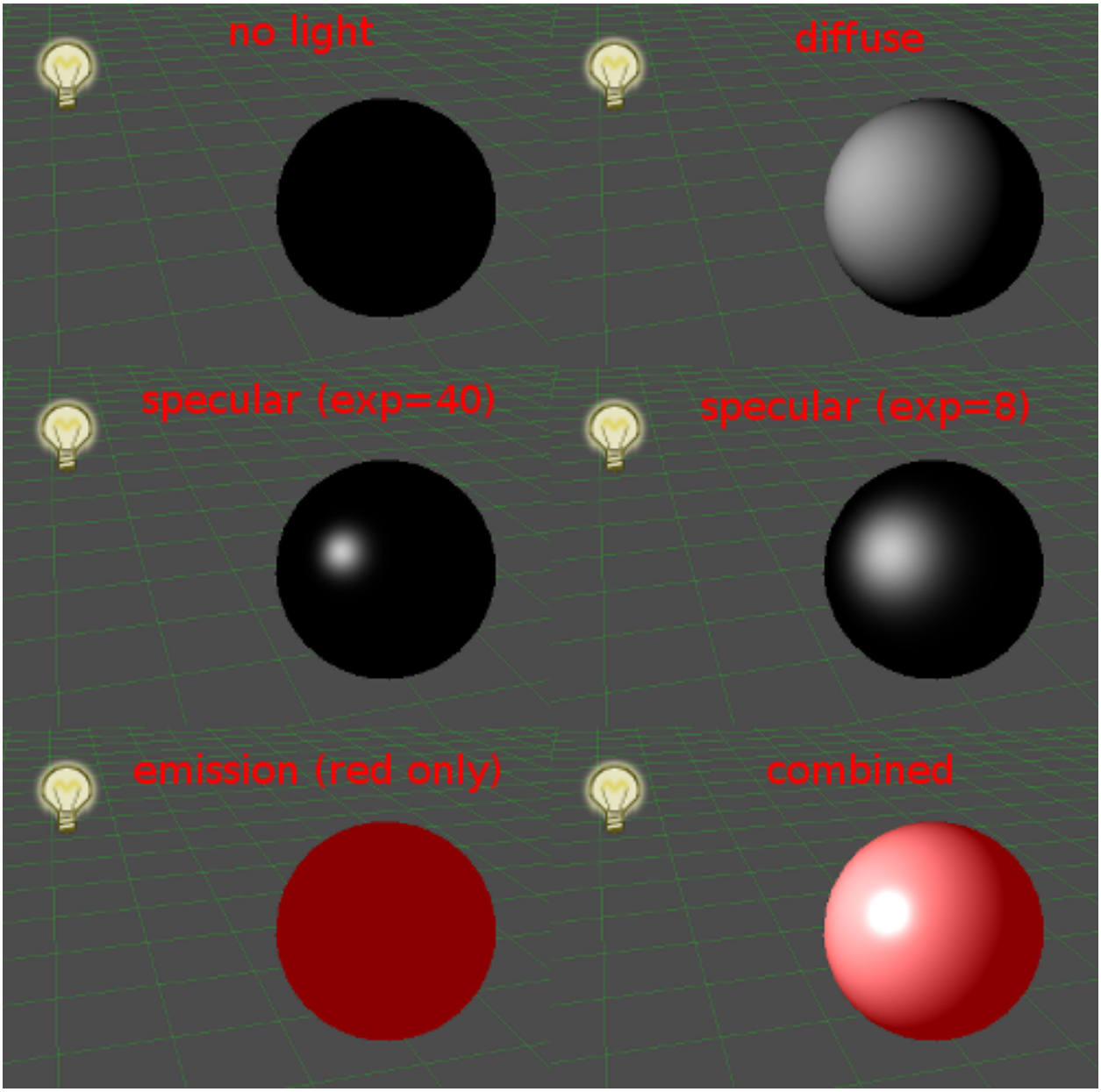
Parameteros

Diffuse, specular, emission y specular exponent

Esto son los colores base para un material.

- Diffuse color es responsable por la iluminación que llega al material, y luego se difumina al rededor. Este color varia por el ángulo a la luz y la distancia (en el caso de luces spot y omni). Es el color que mejor representa al material. También puede tener alpha (transparencia)
- Specular color es el color de la luz reflejada y es responsable de los brillos. Es afectada por el valor de specular exponent.
- Emission es el color de la luz generada dentro del material (aunque no iluminara nada al rededor a no ser que se haga baking). Este color es constante.
- Specular Exponent (o "Shininess"/"Intensity" en algunos DCCs 3D) es la forma en que la luz es reflejada. Si el valor es alto, la luz es completamente reflejada, de otra forma es difuminada mas y mas.

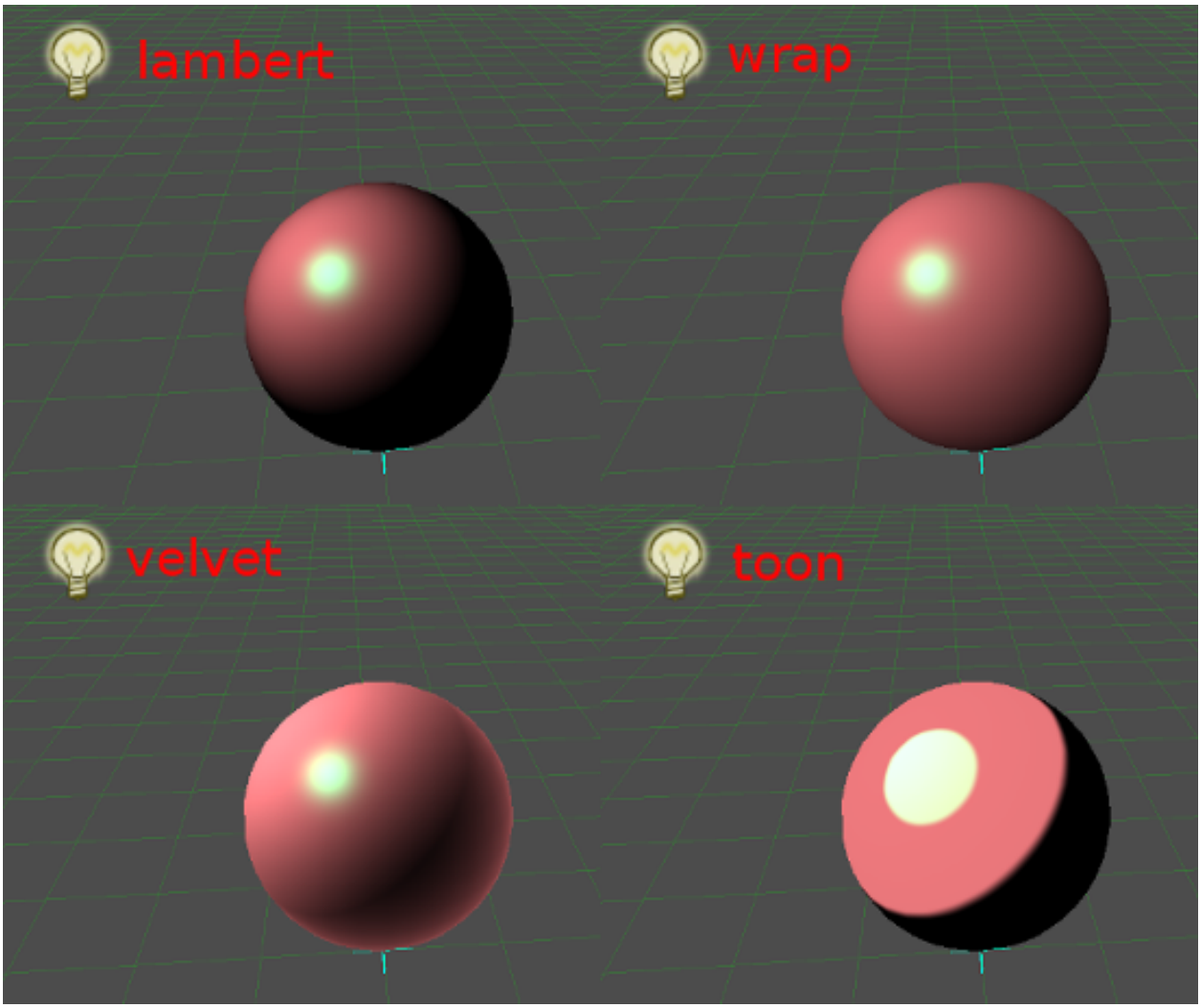
Abajo un ejemplo de como interactúan:



Shader & shader param

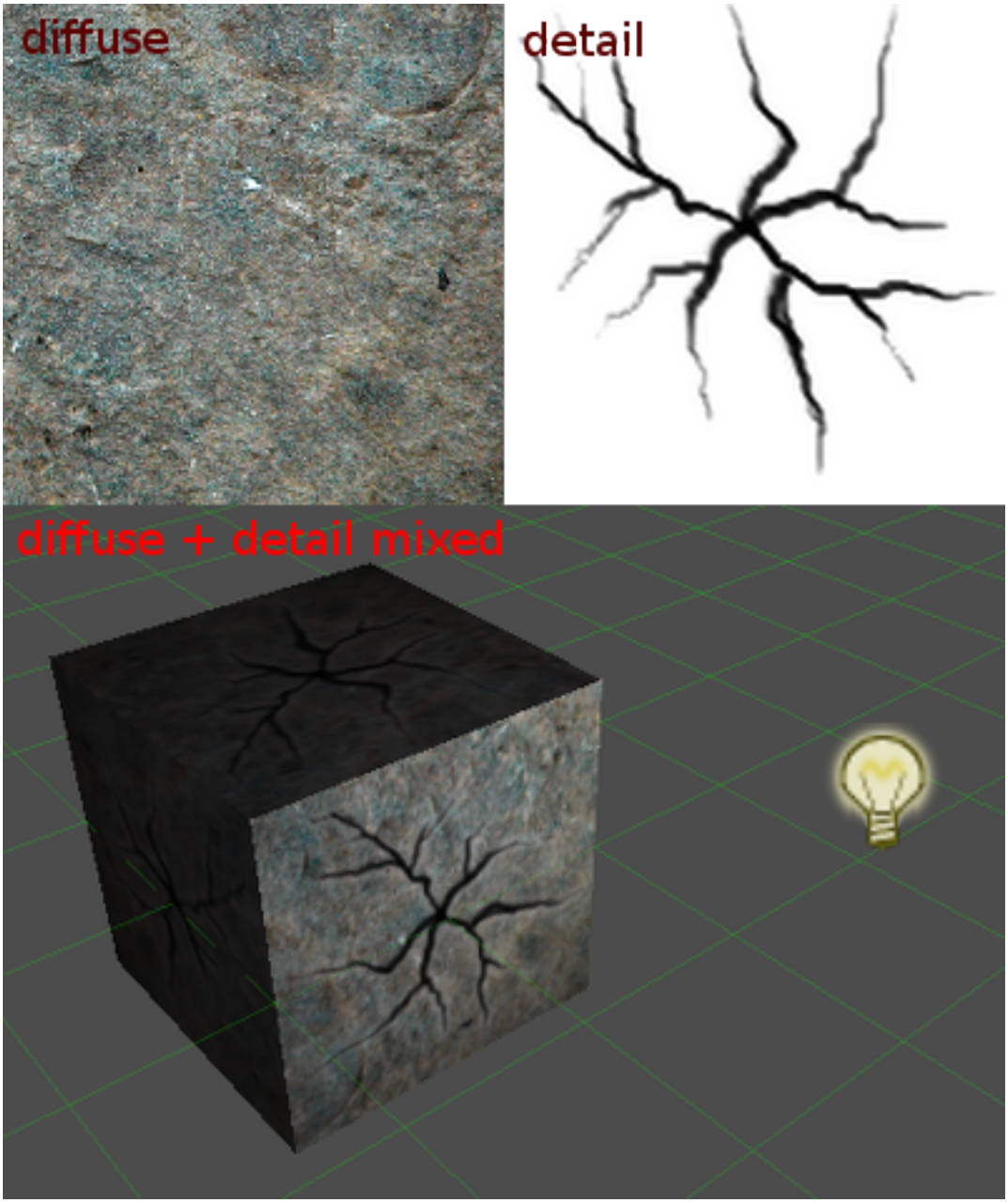
Los materiales shader regulares permiten código personalizado de iluminación. Los materiales fijos vienen con cuatro tipos predefinidos de shader:

- **Lambert**: La luz difusa estandard, donde la cantidad de luz es proporcional al ángulo del emisor de luz.
- **Wrap**: Una variante de Lambert, donde el "coverage" de la luz puede ser cambiado. Esto es útil para muchos tipos de materiales como son madera, arcilla, pelo, etc.
- **Velvet**: Este es similar a Lambert, pero agrega light scattering en los bordes. Es útil para cueros y algunos tipos de metales.
- **Toon**: Sombreado estilo toon standard con un parámetro de cobertura. El componente especular también se vuelve toon-isado.



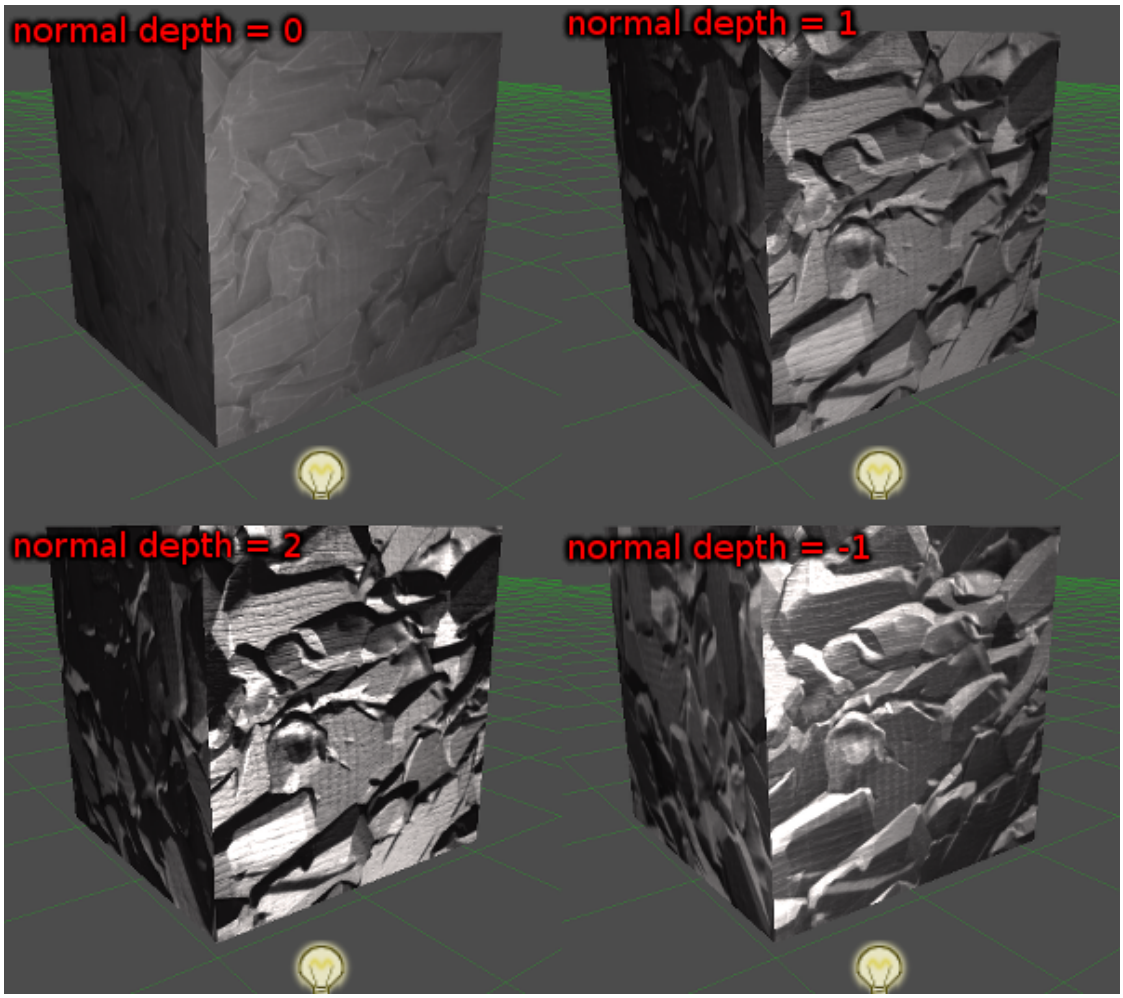
Detail & detail mix

Detail es una segunda textura difusa la cual puede ser mezclada con la primera (mas sobre texturas luego!). Detail blend y mix controlan como estas son sumadas, aquí hay un ejemplo de lo que hacen las texturas detail:



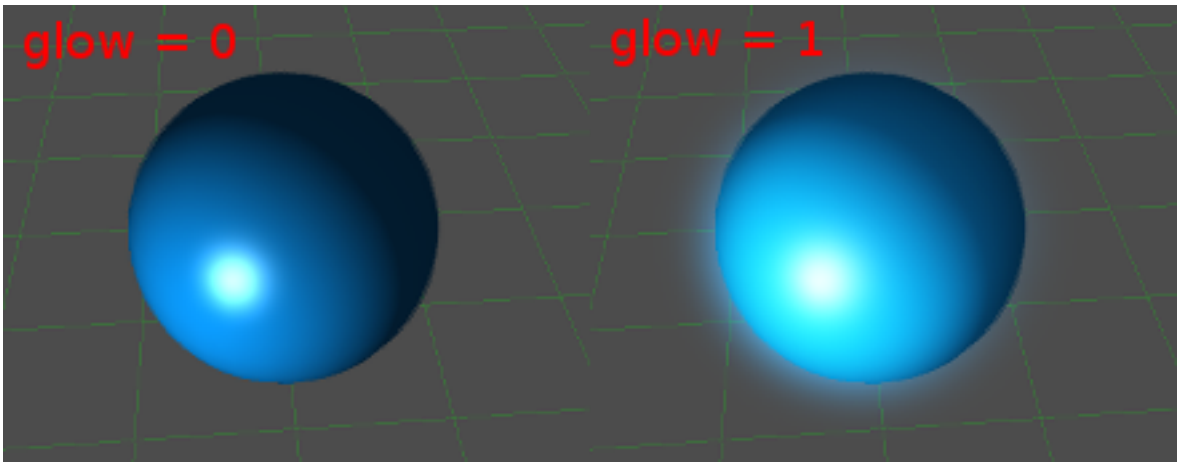
Normal depth

Normal depth controla la intensidad del normal-mapping así como la dirección. En 1 (por defecto) normalmapping aplica normalmente, en -1 el map es invertido y en 0 deshabilitado. Valores intermedios o mayores son aceptados. Aquí esta como debería verse:



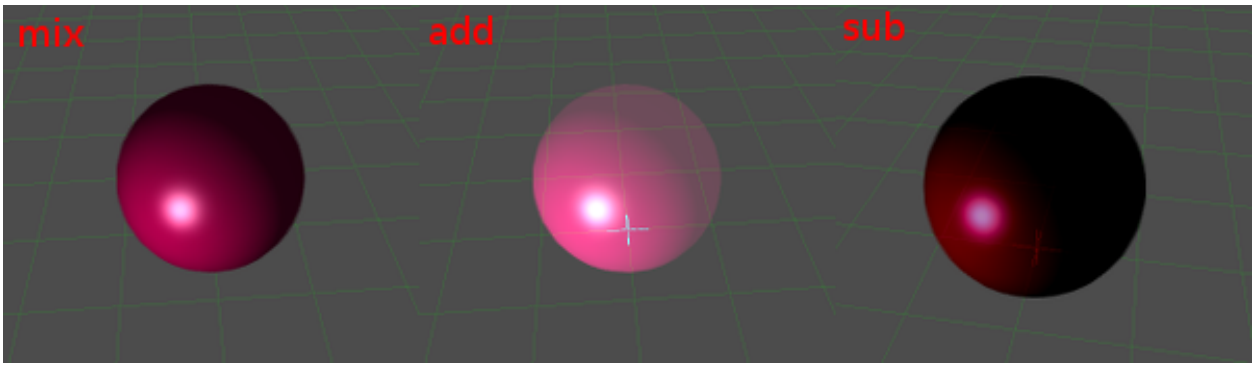
Glow

Este valor controla que cantidad de color es enviado al buffer glow. Puede ser mayor a 1 para un efecto mas fuerte. Para que glow funcione, debe existir un WorldEnvironment con Glow activado.



Blend mode

Los objetos son usualmente mezclados en modo Mix. Otros modos de mezcla (Add y Sub) existen para casos especiales (usualmente efectos de partículas, rayos de luz, etc.) pero los materiales pueden ser ajustados a ellos:



Point size, line width

Cuando dibujas puntos o líneas, el tamaño de ellos puede ser ajustado aquí por material.

Texturas

Casi todos los parámetros de arriba pueden tener una textura asignada a ellos. Hay cuatro opciones de donde pueden obtener sus coordenadas UV:

- **UV Coordinates (UV Array)**: Este es el arreglo de coordenada regular UV que fue importado con el modelo.
- **UV x UV XForm**: Coordenadas UV multiplicadas por la matriz UV Xform.
- **UV2 Coordinates**: Algunos modelos importados pueden venir con un segundo grupo de coordenadas UV. Son comunes en texturas detail

  o para texturas baked light.
- **Sphere**: Coordenadas esféricas (diferencia de la normal en el pixel por la normal de la cámara).

El valor de cada pixel de la textura es multiplicado por el parámetro original. Esto implica que si una textura es cargada para diffuse, será multiplicada por el color del parametro diffuse color. Lo mismo aplica a todos los demás excepto por specular exponent, que es remplazada.

4.1.4 Shader materials (Materiales Shader)

Introducción

Para la mayoría de los casos, *Fixed Materials (Materiales Fijos)* es suficiente para crear las texturas deseadas o el "look and feel". Los materiales shader están un paso más adelante, agregando una cantidad enorme de flexibilidad. Con ellos es posible:

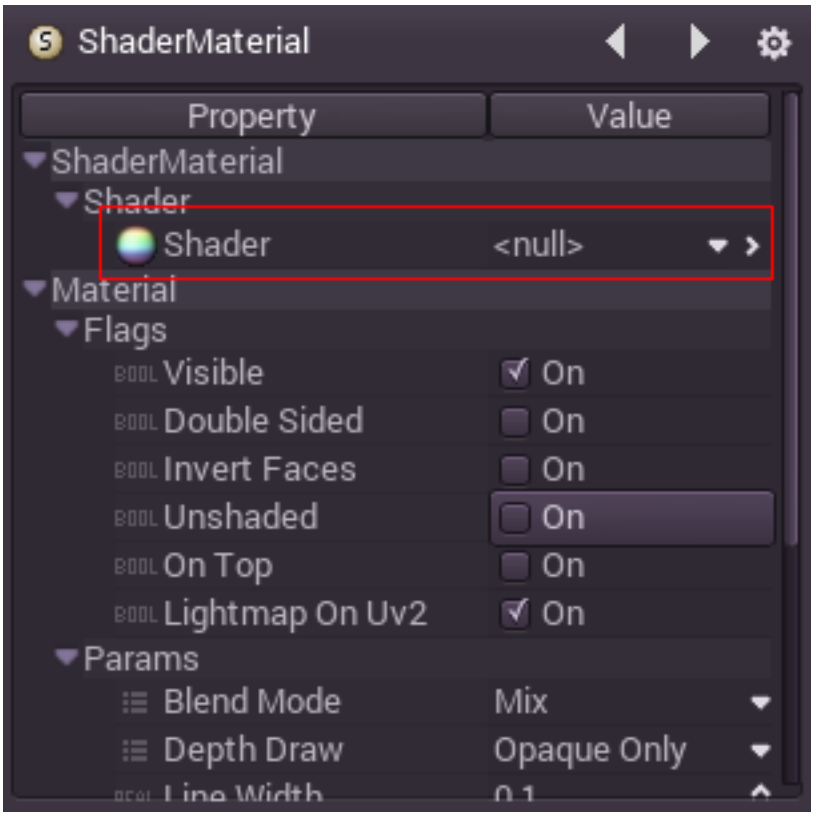
- Crear texturas procedurales.
- Crear texturas con mezclas complejas.
- Crear materiales animados, o materiales que cambian con el tiempo.
- Crear efectos refractarios u otros efectos avanzados.

- Crear shaders de iluminación especial para materiales mas exóticos.
- Animar vértices, como hojas de árboles o pasto.
- Y mucho mas!

Tradicionalmente, la mayoría de los motores te pedirán que aprendas GLSL, HLSL o CG, los cuales son bastante complejos para las habilidades de la mayoría de los artistas. Godot usa una versión simplificada de un lenguaje de shaders que detecta errores en la medida que escribes, así puedes ver la edición de tus shaders en tiempo real. Adicionalmente, es posible editar shaders usando un editor gráfico visual basado en nodos.

Creando un ShaderMaterial

Crea un nuevo ShaderMaterial en algún objeto de tu elección. Ve a la propiedad "Shader", luego crea un nuevo "MaterialShader" (usa "MatrialShaderGraph" para acceder al editor gráfico visual):



Edita el shader recientemente creado, y el editor de shader aparecerá:

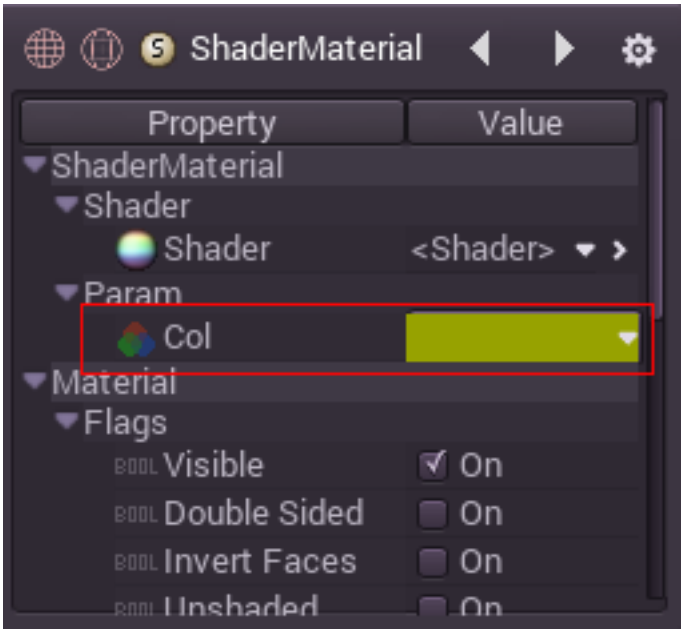
Hay tres pestañas de código abiertas, la primera es para el vertex shader (shader de vértice), la segunda para el fragment (fragmento) y la tercera para lighting (iluminación). El lenguaje de shader esta documentado en *Shading language* asique un pequeño ejemplo será presentado a continuación.

Crear un shader fragment muy simple que escribe un color:

```
uniform color col;
DIFFUSE = col.rgb;
```

Los cambios de código tienen lugar en tiempo real. Si el código es modificado, se recompila instantáneamente y el objeto es actualizado. Si se comete un error de escritura, el editor notificara la falla de compilación:

Finalmente, ve atrás y edita el material, y la exportación uniforme será visible instantáneamente:



Esto permite crear de forma muy rápida y personalizada, materiales complejos para cada tipo de objeto.

4.1.5 Iluminación

Introducción

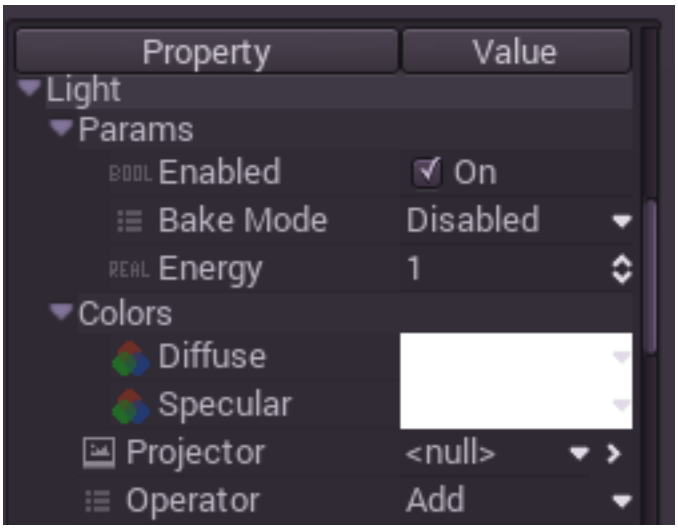
Las luces emiten luz que se mezcla con los materiales y produce un resultado visible. La luz puede venir de diferentes tipos de orígenes en una escena:

- Desde el propio Material, en la forma de emission color (aunque no afecta objetos cercanos, a no ser que sea baked).
- Nodos de luces: Directional, Omni y Spot.
- Luz de ambiente en la clase *Environment*.
- Baked Light (lee doc_light_baking).

La emisión de color es una propiedad del material, como se vio en los tutoriales previos sobre materiales (ve a leeros si aún no lo has hecho!).

Nodos de Luz

Como mencionamos antes, hay tres tipos de nodos de luz: Direccional, Ambiente y Spot. Cada una tiene diferentes usos y será descrita en detalle más abajo, pero primero tomemos un vistazo en los parámetros comunes para luces:

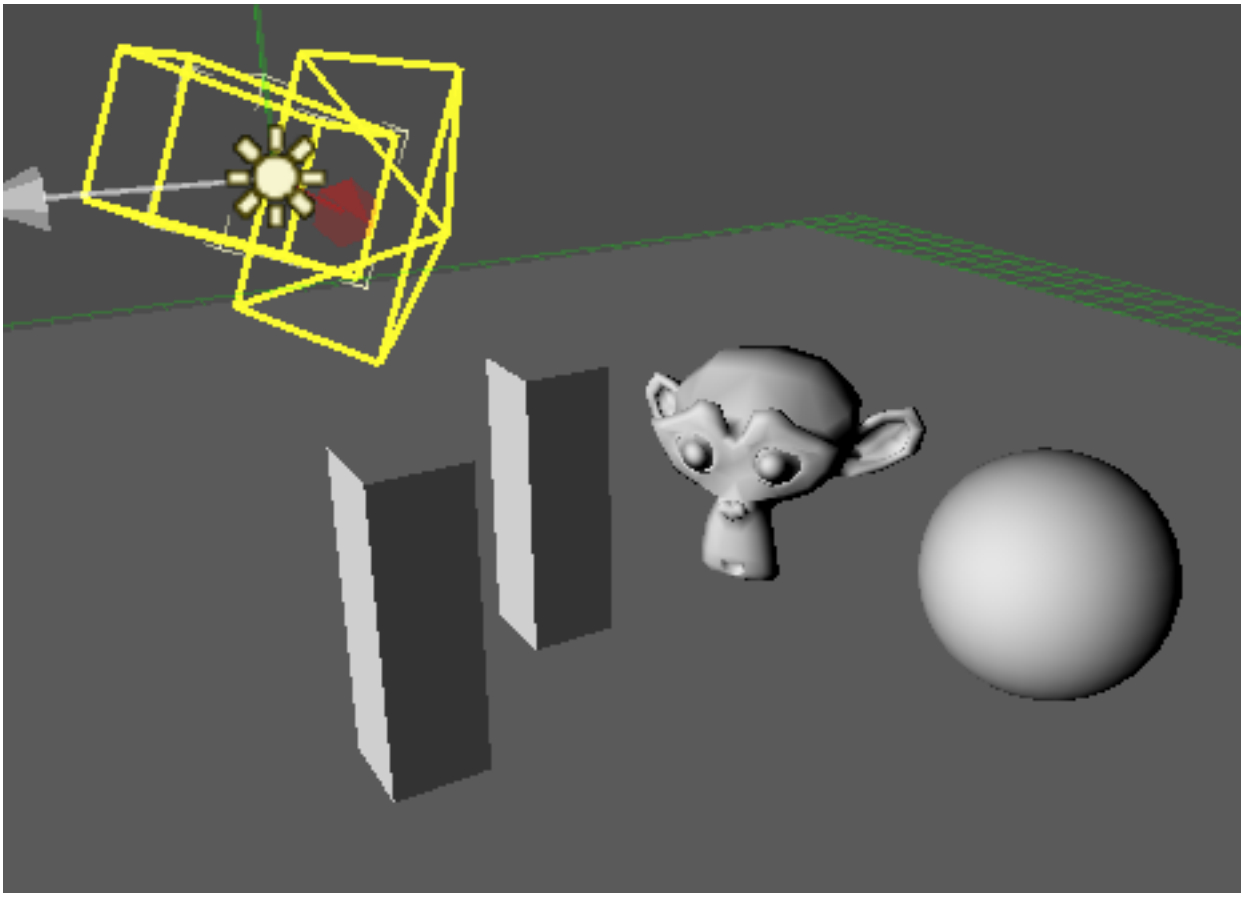


Cada una tiene una funcion especifica:

- **Enabled**: Las luces pueden deshabilitarse en cualquier momento.
- **Bake Mode**: Cuando se usa el light baker, el rol de esta luz puede ser definido en este enumerador. El rol sera respetado aun si la luz esta deshabilitada, lo que permite configurar una luz y luego deshabilitarla para baking.
- **Energy**: Este valor es un múltiplo para la luz, es especialmente útil para *High dynamic range* y para luces Spot y Omni, porque puede crear spots muy brillantes cerca del emisor.
- **Diffuse and Specular**: Estos valores de luz son multiplicados por la luz del material y los colores difusos, por lo que un valor blanco no significa que la luz será toda blanca, pero que el color original será mantenido.
- **Operator**: Es posible hacer algunas luces negativas para un efecto de oscurecimiento.
- **Projector**: Las luces pueden proyectar una textura para la luz difusa (actualmente solo soportado para luces Spot).

Directional light

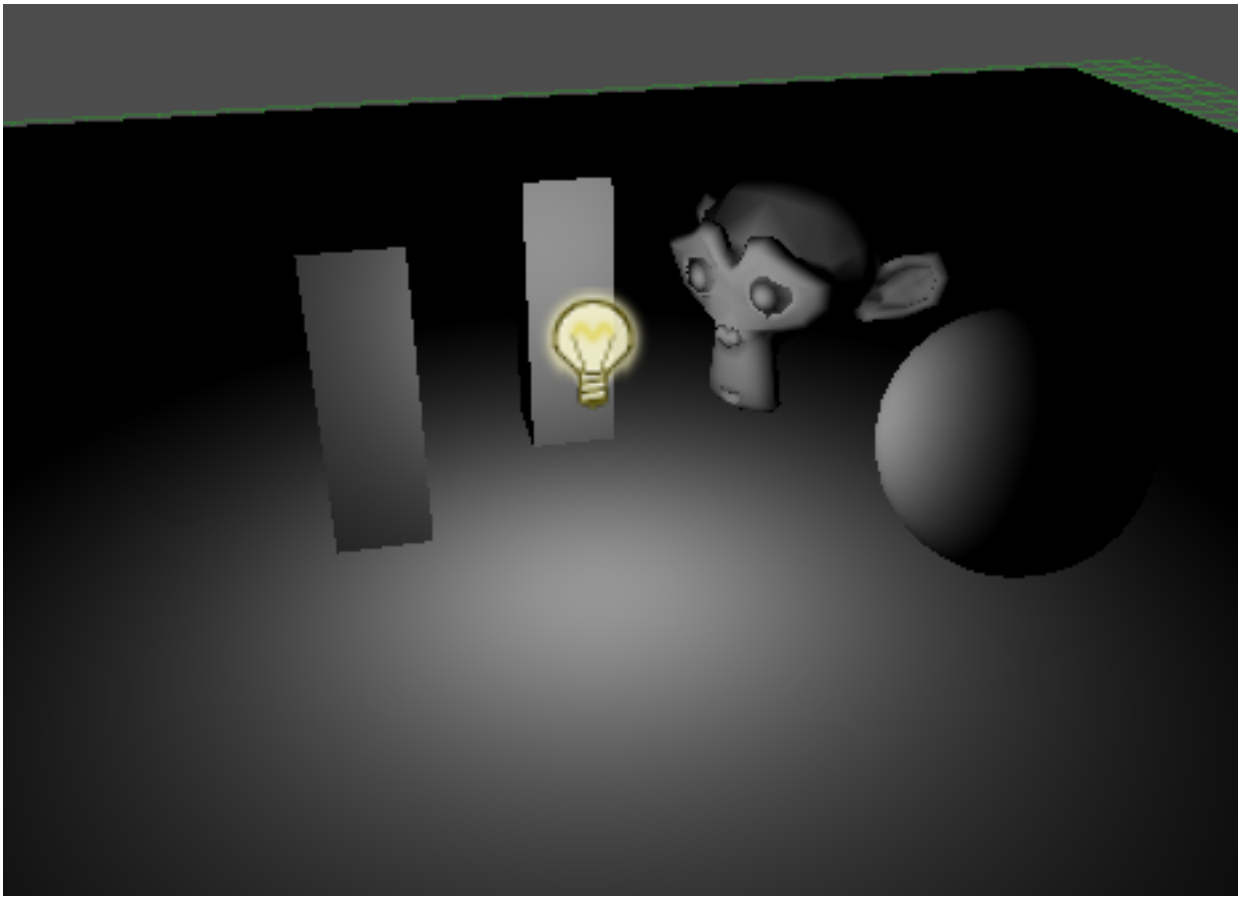
Este es el tipo más común de luz y representa al sol. También es la luz más barata de computar y debe ser usada siempre que sea posible (aunque no es el shadow-map mas barato de computar, más sobre ello luego). Los nodos de luces direccionales están representados por una flecha grande, que representa la dirección de la luz, sin embargo la posición del nodo no afecta la luz para nada, y puede estar en cualquier lugar.



Básicamente lo que mira hacia la luz es iluminado, lo que no es oscuro. La mayoría de las luces tienen parámetros específicos pero las luces direccionales son bastante simples por naturaleza por lo que no tienen.

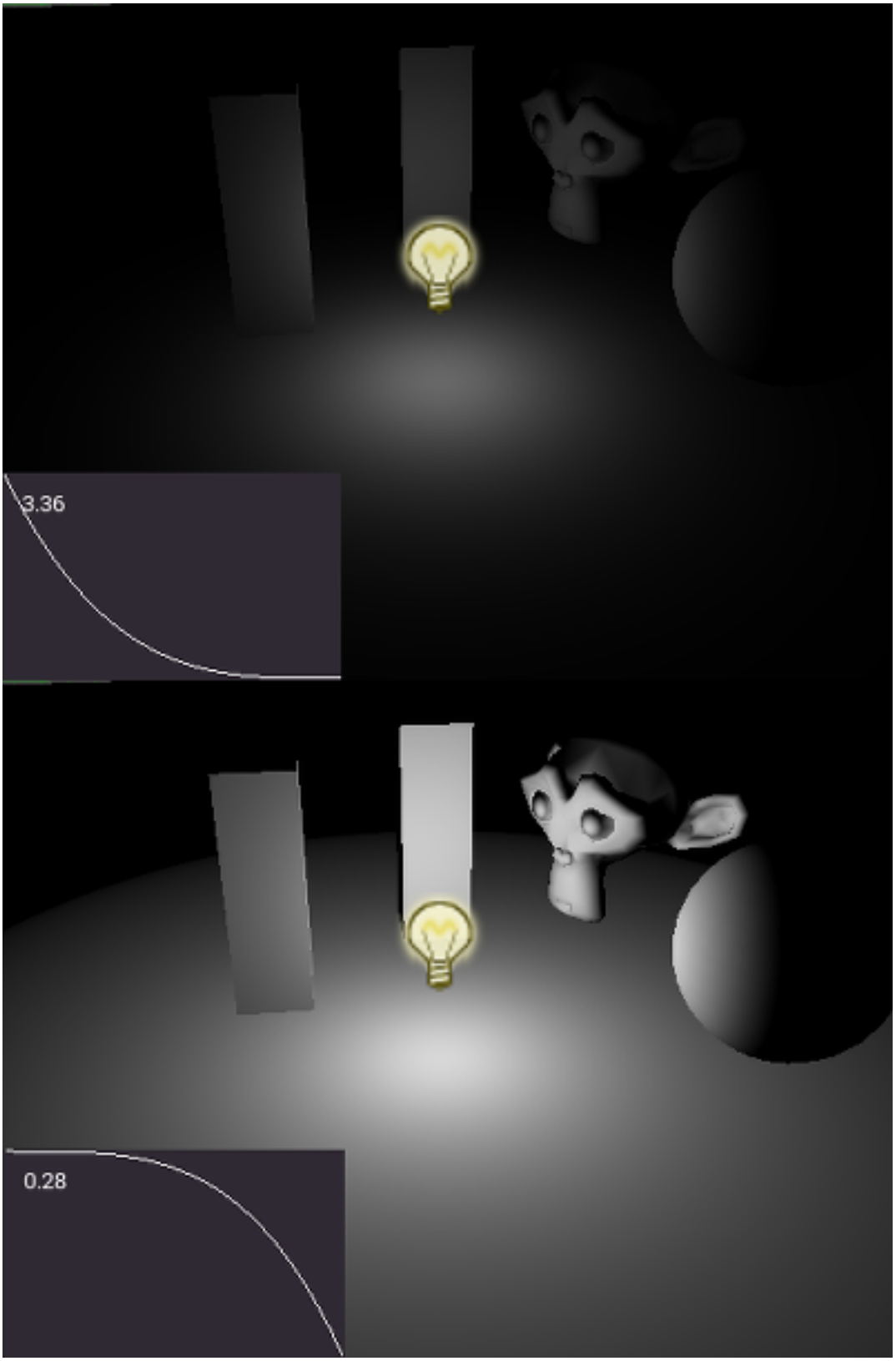
Omni light

Las luces Omni son un punto que tira luz a su alrededor hasta un cierto radio (distancia) que puede ser controlado por el usuario. La luz se atenúa con la distancia y llega a 0 en el borde. Representa lámparas o cualquier otra fuente de luz que viene de un punto.



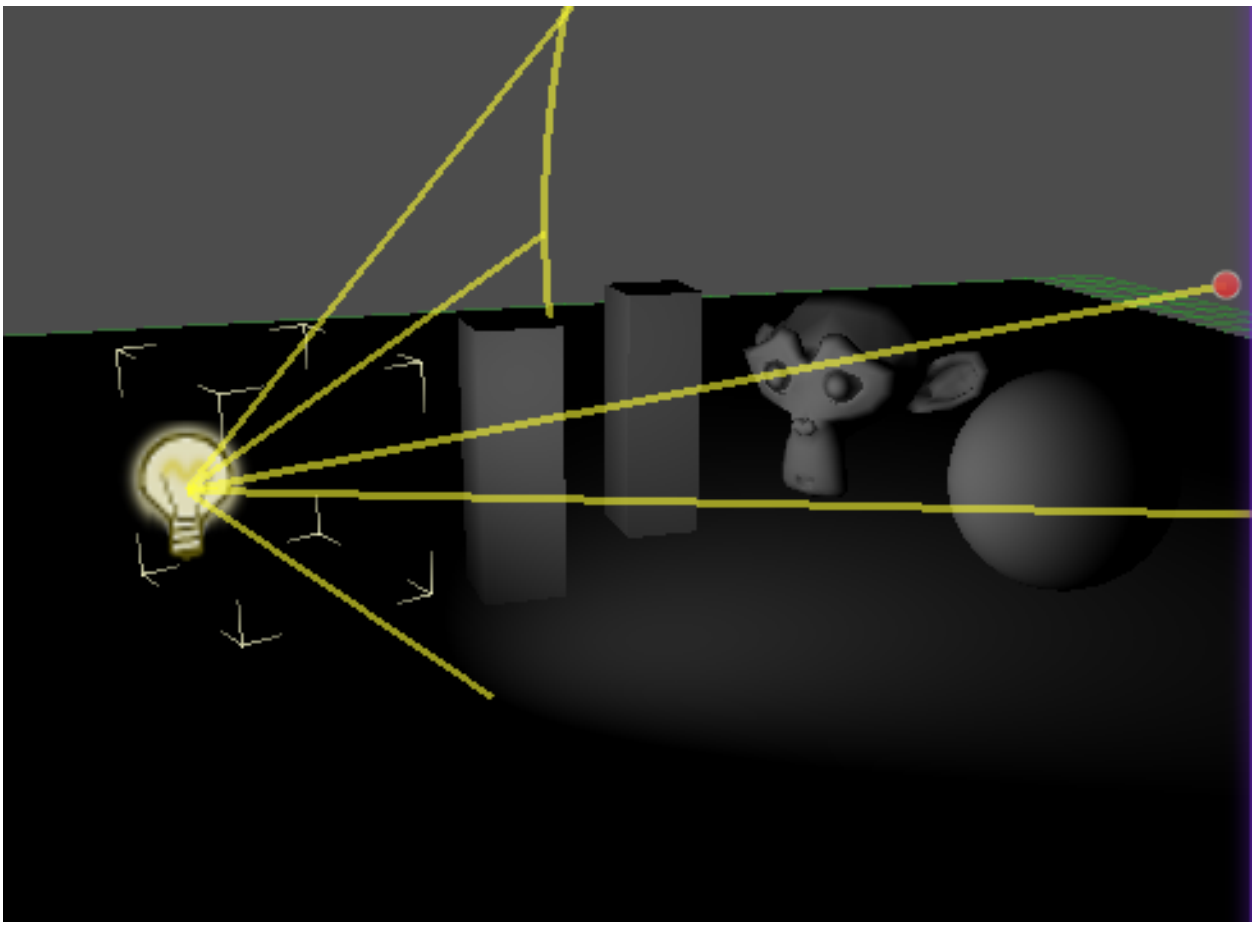
La curva de atenuación para este tipo de luces por naturaleza es computada con una función cuadrática inversa que nunca llega a cero y tiene valores infinitamente grandes cerca del emisor.

Esto las vuelve considerablemente inconveniente para que los artistas las ajusten, así que Godot las simula con una curva exponencial controlada por el artista.


3.36
0.28

Spot Light

Las luces Spot son similares a las luces Omni, excepto que solo operan entre un ángulo dado (o "cutoff"). Son útiles para simular linternas, luces de autos, etc. Este tipo de luz también es atenuada hacia la dirección opuesta de donde apunta.



Ambient light

La luz de ambiente puede ser encontrada en las propiedades de un WorldEnvironment (recuerda solo una de ellas puede ser instanciada por escena). Ambient light consiste de luz y energía uniformes. Esta luz es aplicada por igual a cada pixel de la escena renderizada, excepto los objetos que usan baked light.

Baked light

Baked light significa luz de ambiente pre-computada. Puede servir muchos propósitos, como hacer baking de emisores de luz que no serán usados en tiempo real, y backing de rebotes de luces de tiempo real para agregar más realismo a la escena (ve el tutorial doc_light_baking para mas información).

4.1.6 Shadow mapping (Mapeo de sombras)

Introducción

Simplemente tirar una luz no es suficiente para iluminar realísticamente una escena. Debería ser, en teoría, pero debido a la forma en la que el hardware de video funciona, partes de objetos que no deberían ser alcanzados por la luz son iluminados de todas formas.

La mayoría de la gente (incluyendo artistas), ven las sombras como algo proyectado por la luz, como si fueran creados por la misma luz al oscurecer lugares que están escondidos del origen de luz.

Esto en realidad no es correcto y es importante entender que las sombras son lugares donde la luz simplemente no llega. Como regla (y sin contar la luz indirecta) si una luz es apagada, los lugares donde hay sombras deberían permanecer igual. En otras palabras, las sombras no deberían ser vistas como algo "agregado" a la escena, pero como un área que "permanece oscura".

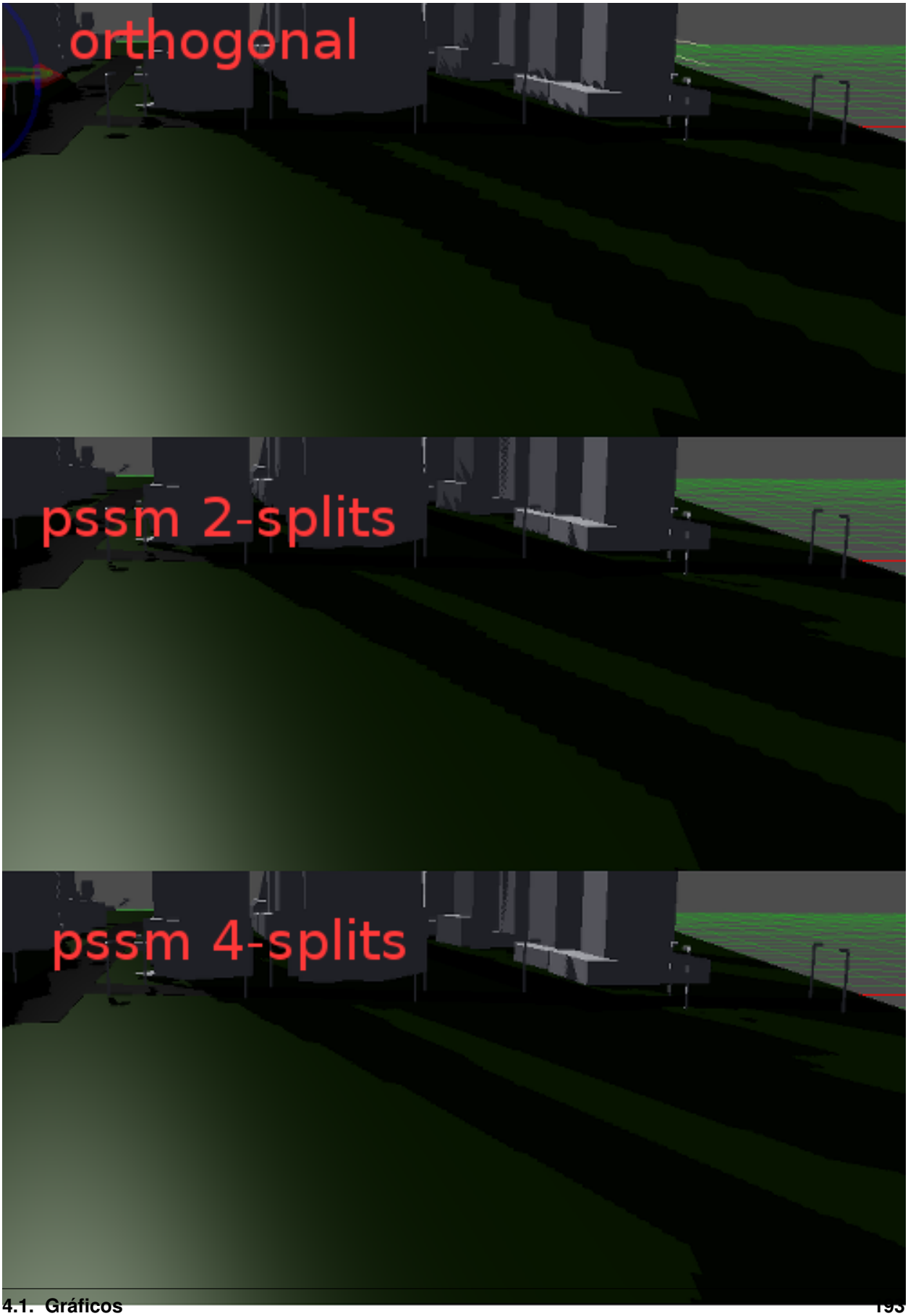
Todos los tipos de luces en Godot pueden usar shadow mapping, y todos soportan varias técnicas diferentes que intercambian calidad por rendimiento. Shadow mapping usa una textura que guarda la "vista de profundidad" de la luz y chequea contra ella en tiempo real para cada pixel que renderiza.

Cuanto mas grande la resolución de la textura shadow map, mayor el detalle que la sombra tiene, pero consumirá mas memoria de video y ancho de banda (lo cual significa que los cuadros por segundo bajan).

Sombras por tipo de luz

Sombras para Directional lights

Directional lights pueden afectar un área realmente grande. Cuanto mas grande la escena, mayor el área afectada. Dado que la resolución del shadow map se mantiene igual, la misma cantidad de pixeles de sombra cubren un área mayor, resultando en sombras en bloques. Muchas técnicas existen para tratar con problemas de resolución, pero la mas común es PSSM (Parallel Split Shadow Maps):

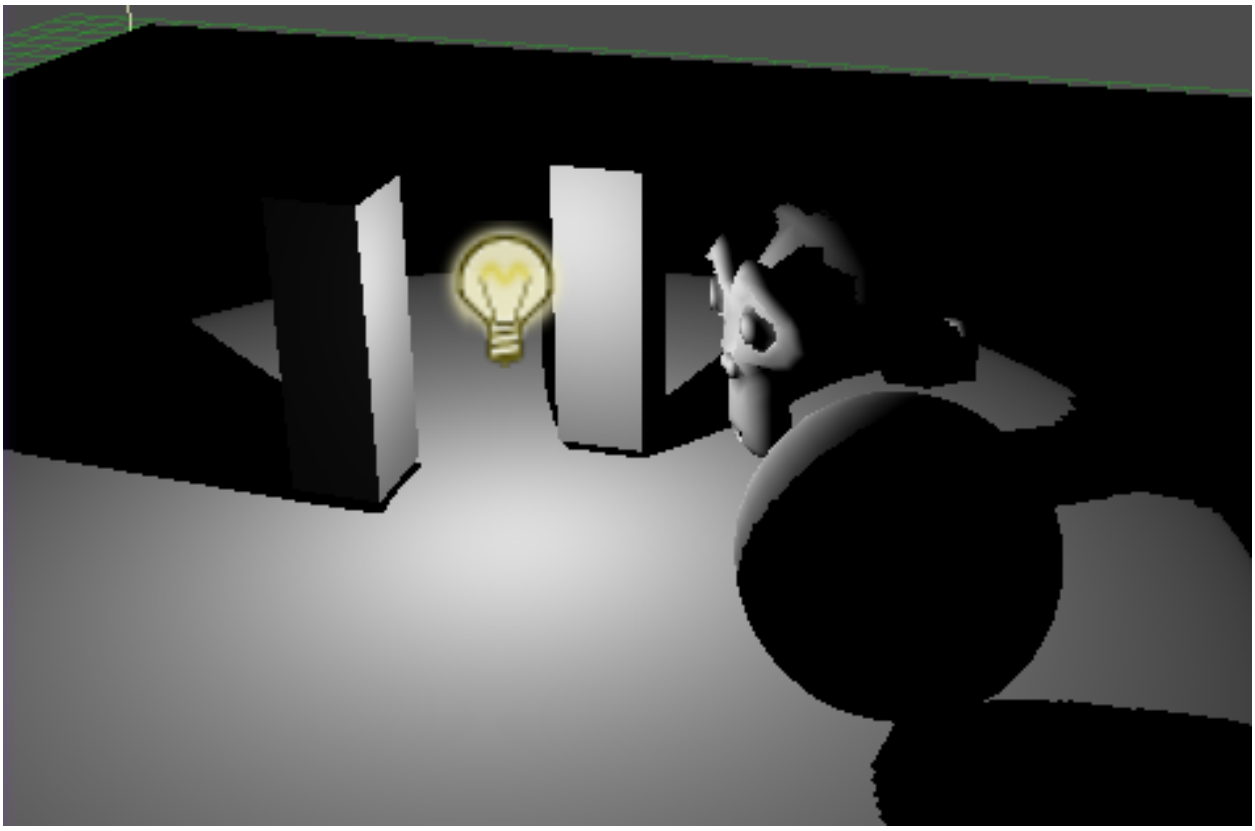

orthogonal
pssm 2-splits
pssm 4-splits

Esta técnica divide la vista en 2 o 4 secciones, y una sombra es renderizada para cada uno. De esta forma, los objetos cercanos pueden usar sombras mas grandes mientras que cuando mas lejano el objeto usara una con menos detalle, pero en proporción esto parece hacer que el tamaño del shadow map crezca cuando en realidad se mantiene igual. Por supuesto, esta técnica no es gratis, cuanto mas divisiones mas bajara el rendimiento. Para mobile, es generalmente inconveniente usar mas de 2 secciones.

Una técnica alternativa es PSM (Perspective Shadow Mapping). Esta técnica es mucho mas barata que PSSM (tan barata como ortogonal), pero solo funciona realmente para unos pocos ángulos de cámara respecto a la luz. En otras palabras, PSM solo es util para juegos donde la dirección de la cámara y luz son ambas fijas, y la luz no es paralela a la cámara (que es cuando PSM se rompe por completo).

Sombras para Omni light

Las luces omnidireccionales también son problemáticas. Como representar 360 grados de luz con una sola textura? Hay dos alternativas, la primera es usar DPSM (Dual Paraboloid Shadow Mapping). Esta técnica es rápida, pero requiere la utilización de DISCARD (lo que la vuelve no muy útil para mobile). DPSM también puede lucir mal si la geometría no tiene suficiente tessellated, por lo que mas vértices pueden ser necesarios si no luce correcto. La segunda opción es simplemente no usar un shadow map, y usar un shadow cubemap. Esto es mas rápido, pero requiere seis pasadas para renderizar en todas las direcciones y no es soportado en el renderer actual (GLES2).



Algunas consideraciones cuando se usan shadow mas DPSM:

- Mantén Slope-Scale en 0.
- Usa un valor pequeño para Z-Offset, si luce mal, redúcelo mas.
- ESM filtering puede mejorar como luce.

- Las juntas entre las dos mitades de la sombra son generalmente perceptibles, así que rota la luz para hacer que se vean menos.

Sombras Spot light

Las sombras spot light son generalmente las mas simples, solo necesitan una única textura y sin técnicas especiales.

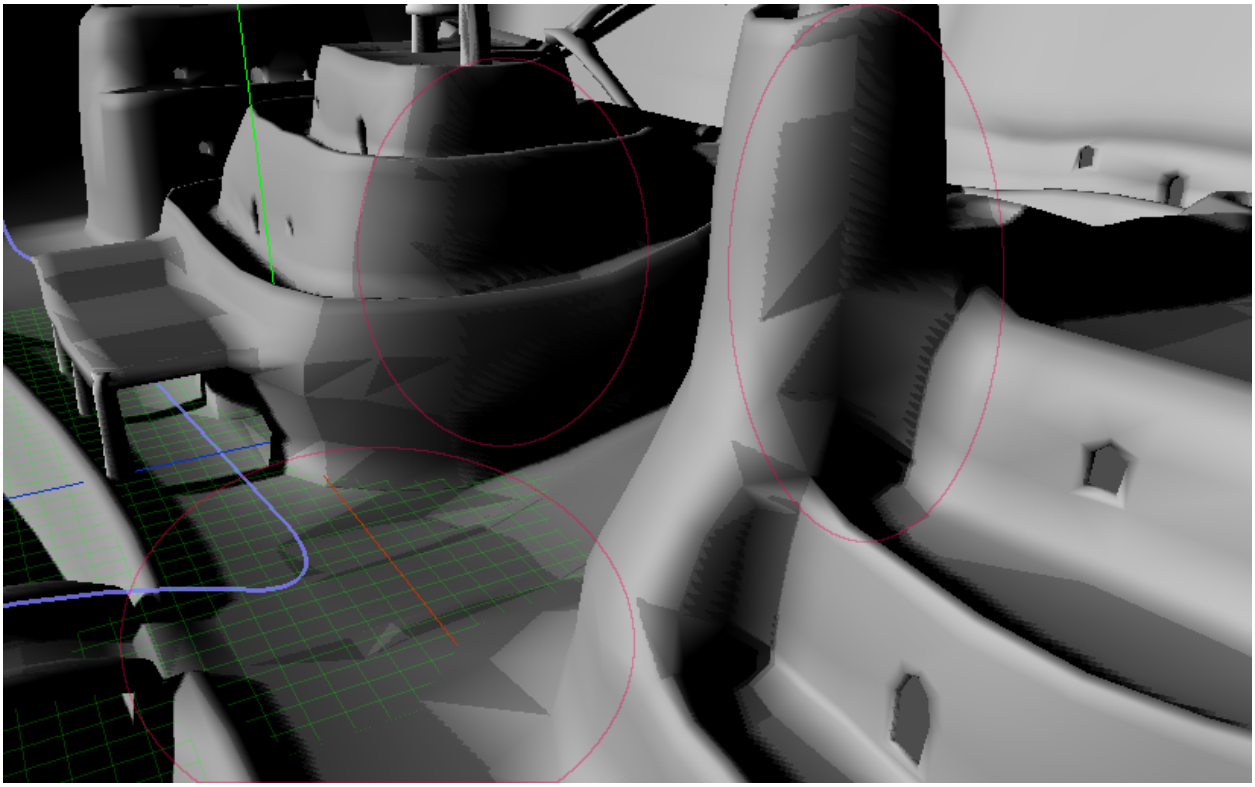
Parámetros de sombras

El hecho de que las sombras son en realidad una textura puede generar varios problemas. El mas común es Z fighting (líneas en los bordes de los objetos que proyectan las sombras). Hay dos formas de solucionar esto, la primera es ajustar los parámetros offset, y la segunda es usar un algoritmo de filtración de sombras, que generalmente luce mejor y no tiene tantos problemas técnicos, pero consume mas GPU.

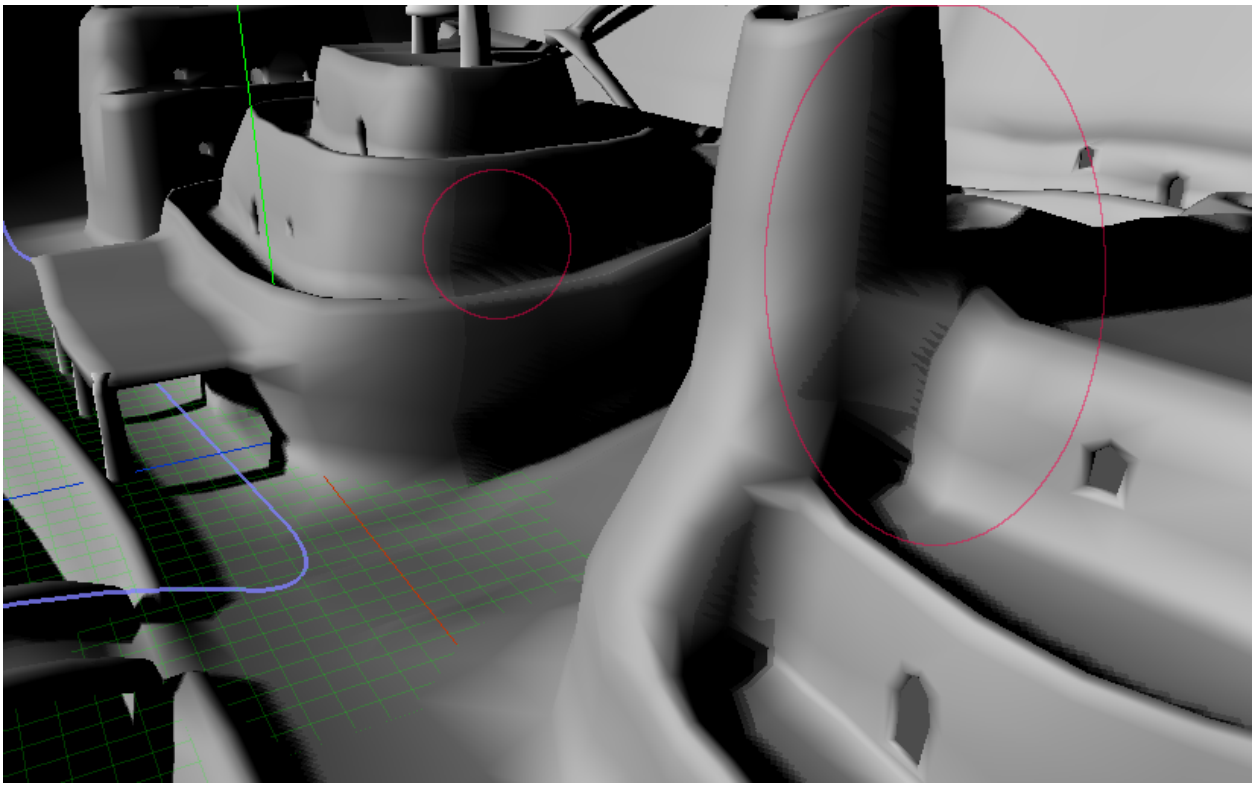
Ajustando z-offset

Entonces, te decidiste a ir con sombras no filtradas porque son mas rápidas, necesitas un poco mas de detalle o tal vez es que te gustan las sexy líneas exteriores de sombras con forma de sierra porque te recuerdan a tus juegos favoritos de generación anterior. La verdad es que, esto puede ser un dolor, pero la mayoría de las veces puede ser ajustado para tener lindos resultados. No hay número mágico y cualquiera sea el resultado que obtengas será diferente de escena a escena, necesita un tiempo de ajuste. Vayamos paso por paso.

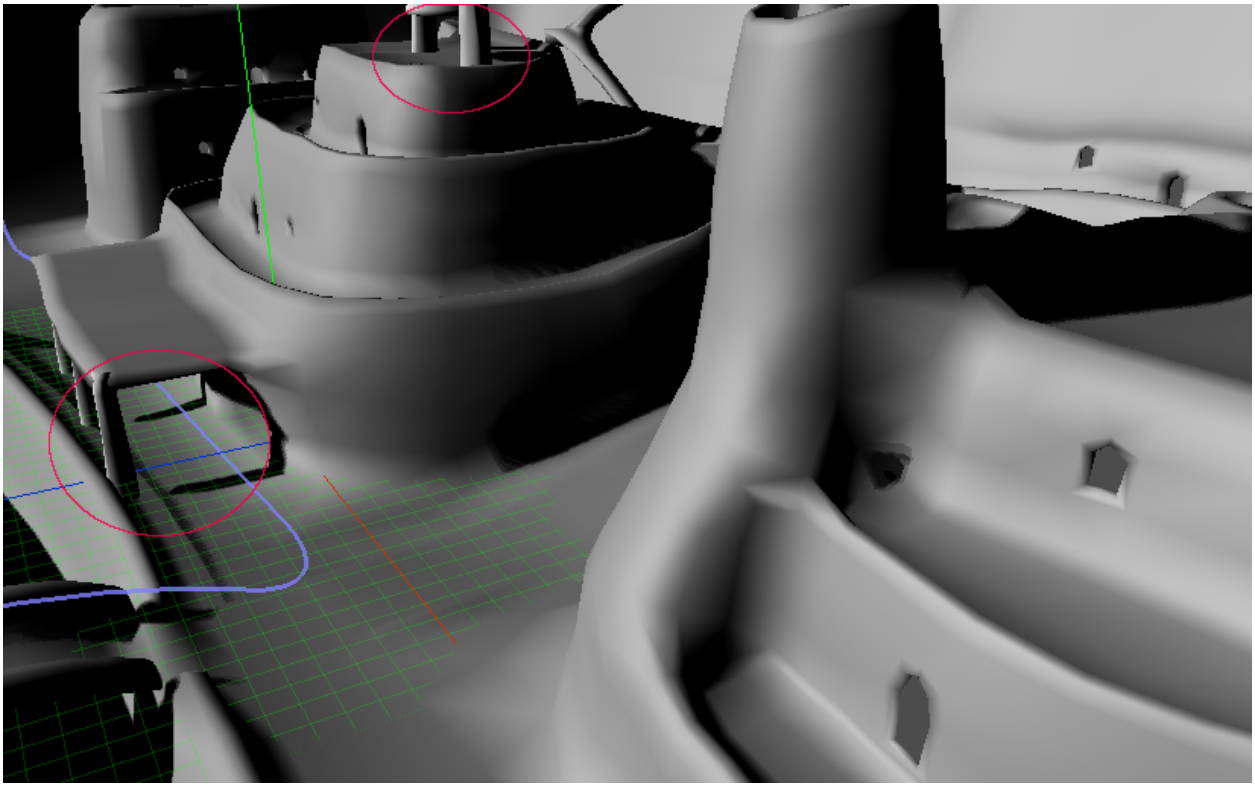
El primer paso es encender las sombras, asumamos que ambos Z-Offset y Z-Slope-Scale estan en 0. Esto se presentara:



Rayos, la sombra esta por todos lados y con problemas técnicos extremos! Esto sucede porque la sombra esta peleando con la misma geometría que la esta proyectando. Esto es llamado "self-shadowing". Para evitar esto no tiene sentido luchar, te darás cuenta que necesitas la paz entre la sombra y la geometría, asi que empujas la sombre un poco al incrementar el Z-Offset de la sombra. Esto mejora las cosas un montón:

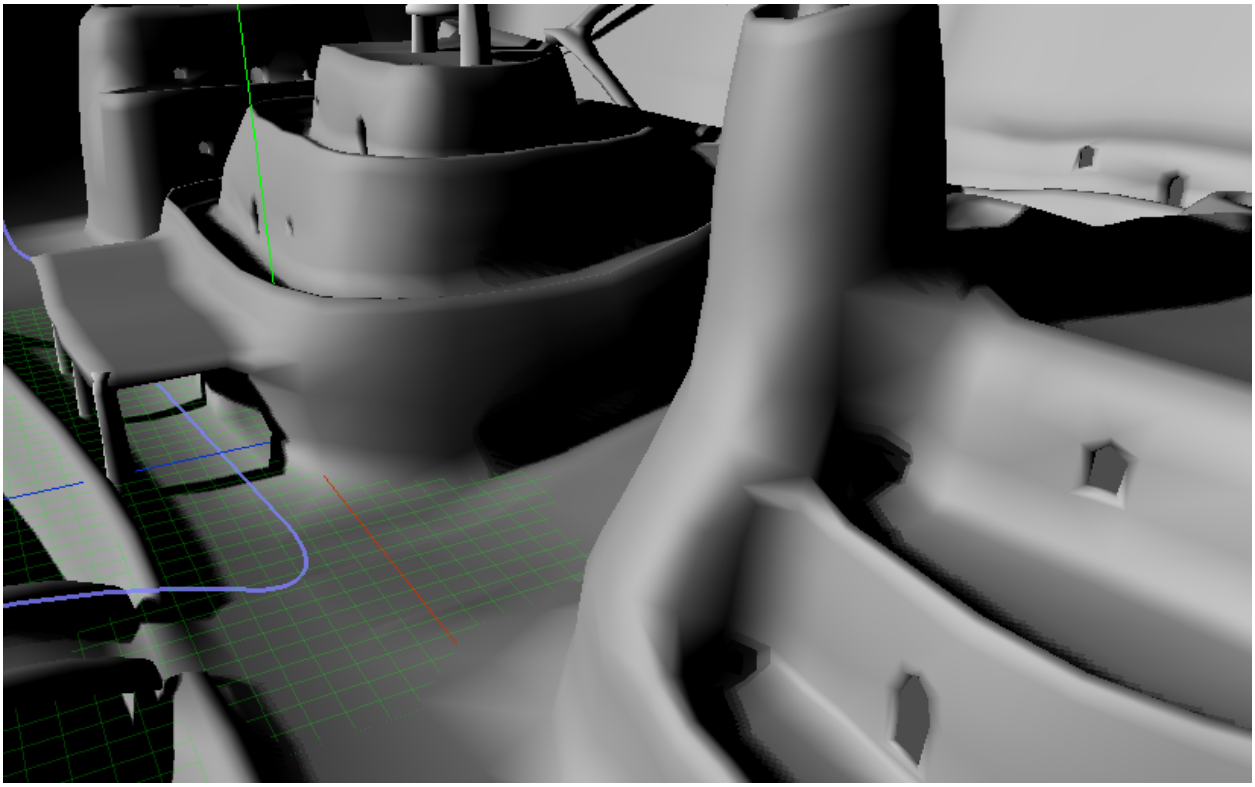


Pero aun no es perfecto, self-shadowing no desapareció por completo. Estamos cerca de la perfección pero aun no la logramos.. así que en un impulso de codicia aumentas el Z-Offest aún más!

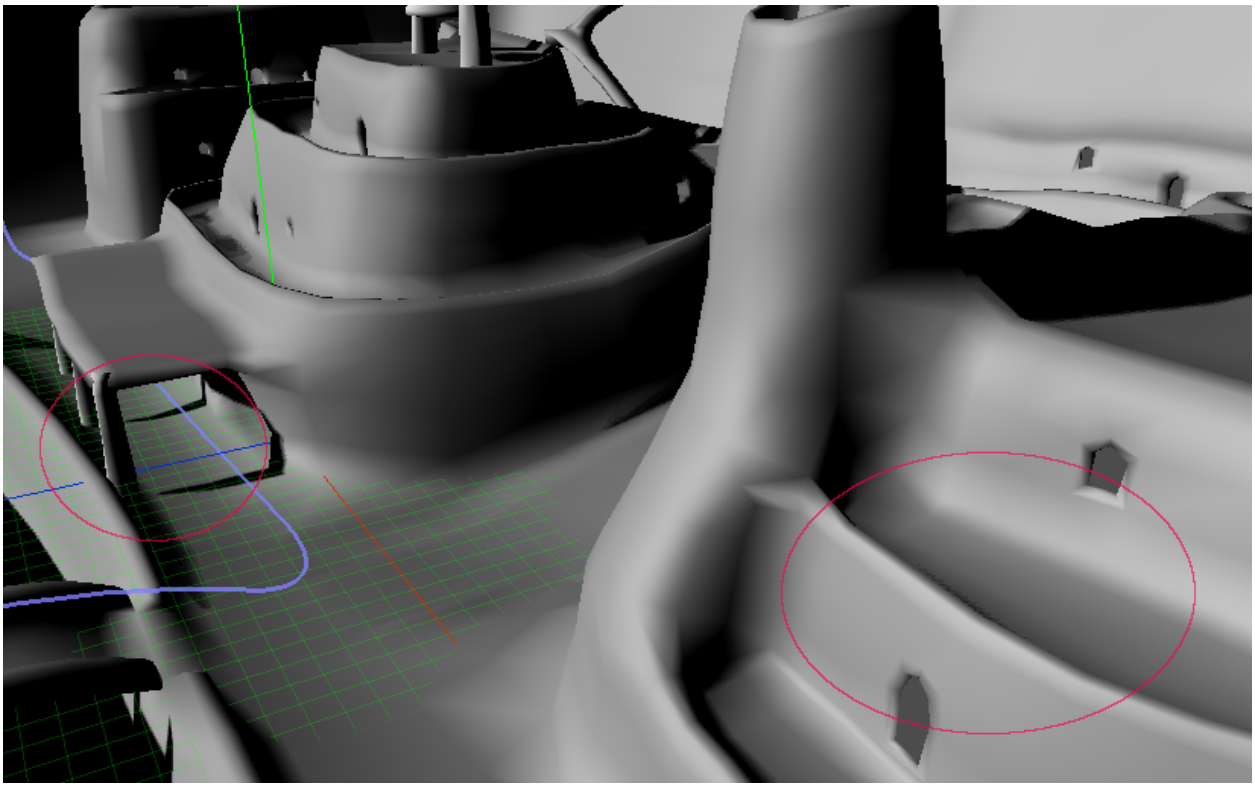


Y logra eliminar esos self-shadowings! Hooray! Excepto que algo esta mal.. oh, bueno. Al ser empujadas demasiado, las sombras empiezan a desconectarse de sus emisores, lo cual luce bastante feo. Ok, vas de nuevo al Z-Offset previo.

Aquí es cuando Z-Slope-Scale viene a salvar el día. Este ajusta hace mas delgados a los objetos emisores de sombras, entonces los bordes no sufren self-shadow:



Aha! Finalmente algo que luce aceptable. Es perfectamente aceptable y tu puedes perfectamente sacar un juego que luce como esto (imagina que estas mirando calidad de arte de Final Fantasy, no este horrible intento de modelado 3D). Puede haber muy pequeñas partes aun con self-shadowing que a nadie le importa, así que tu codicia interminable vuelve a aparecer y aumentas nuevamente Z-Slope-Scale:



Bueno, eso fue mucho, las sombras emitidas son demasiado finas y ya no lucen bien. Mala suerte, el ajuste previo era bueno de todas formas, aceptemos que la perfección no existe y vayamos por algo mas.

Importante!

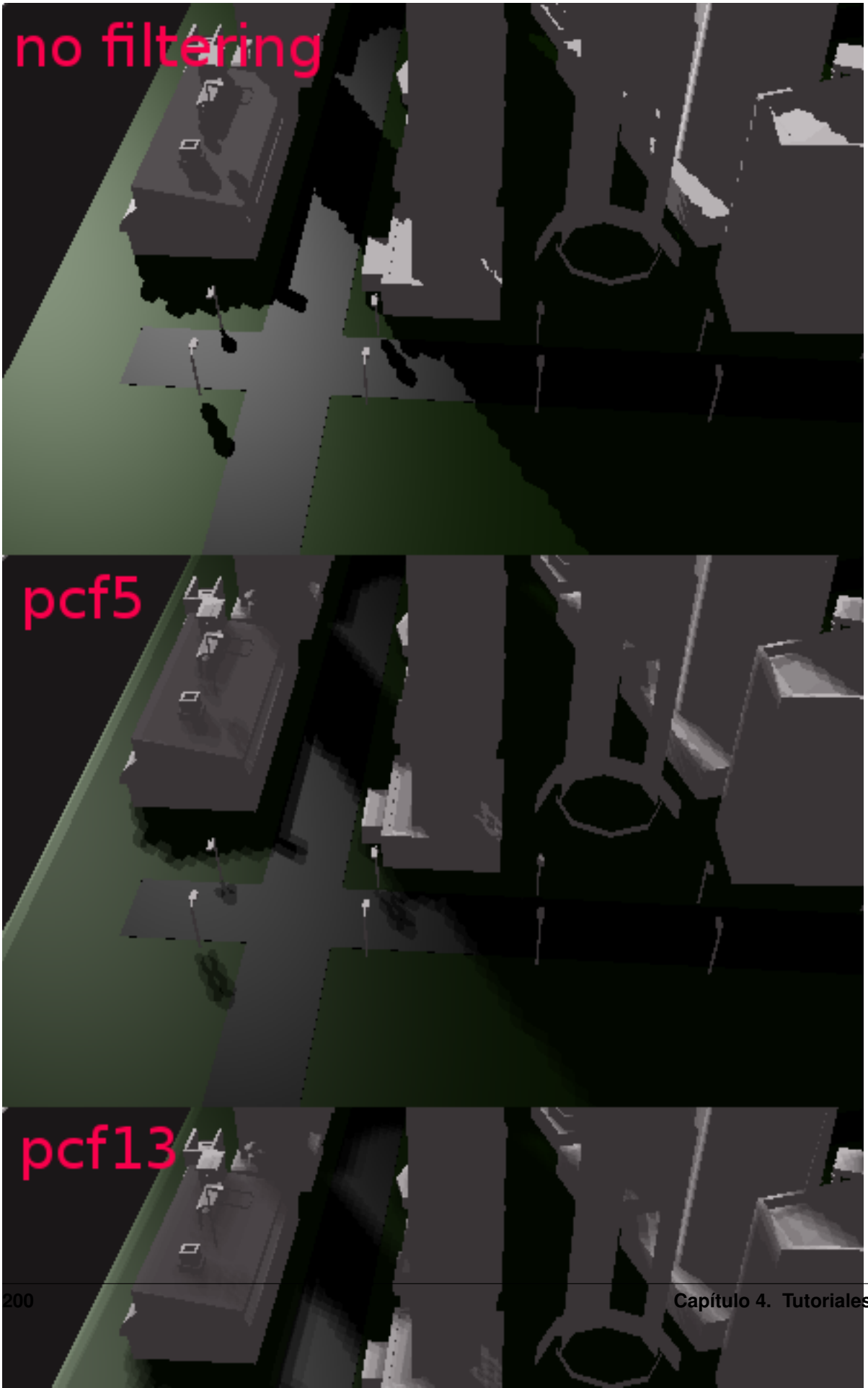
Si estas usando shadow maps con directional lights, asegúrate que la *view distance* de la *camara* esta ajustada en *optimal range*. Esto implica que si la distancia entre tu cámara y el final visible de la escena es 100, entonces ajusta la view distance a ese valor. Si un valor mas grande que el necesario es usado, los shadow maps van a perder detalle ya que intentaran cubrir un área mayor.

Así que, siempre asegúrate de usar el rango optimo!

Filtrado de sombras

Las sombras crudas tienen bloques. Aumentar su resolución solo vuelve los bloques más pequeños, pero aún son bloques.

Godot ofrece algunas formas de filtrarlas (la sombre en este ejemplo es de baja resolución a propósito!):


no filtering
pcf5
pcf13

PCF5 y PCF13 son filtros textura-espacio simples. Harán la textura un poco mas aceptable pero aun necesitara una resolución considerable para lucir bien.

ESM es un filtro más complejo y tiene algunos parámetros de ajuste. ESM usa shadow blurring (la cantidad de blur passes y multiplier pueden ser ajustados).

4.1.7 High dynamic range

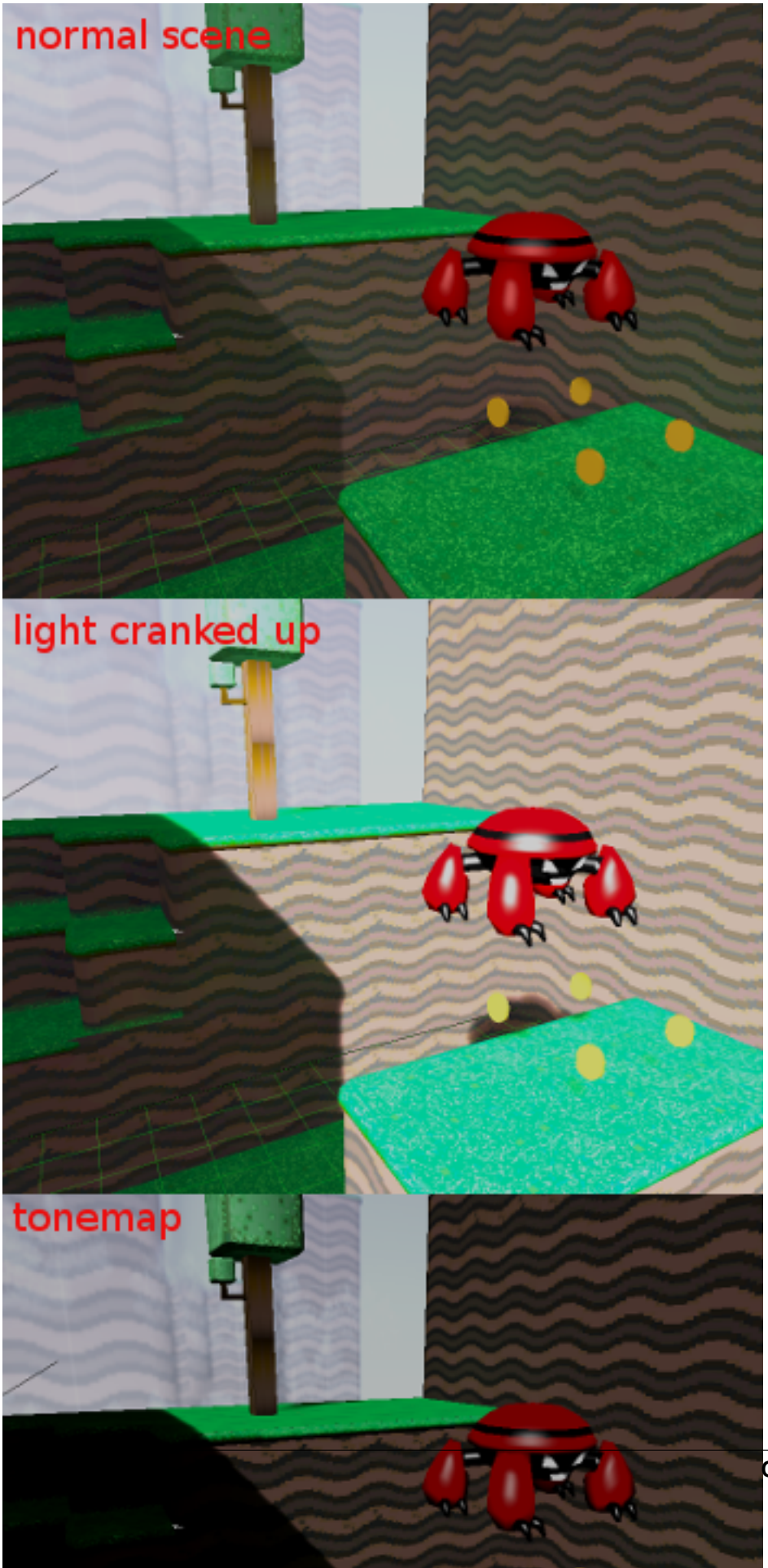
Introducción

Normalmente, un artista hace todo el modelado 3d, luego las texturas, mira su modelo que luce increíblemente bien en el DCC 3D (Maya, Blender, etc) y dice "luce fantástico, listo para la integración!" luego va al juego, se ajustan las luces y el juego corre.

Así que de donde viene todo este tema de HDR? La idea es que en lugar de tratar con colores que van del negro al blanco (0 a 1), usamos colores más blancos que el blanco (por ejemplo, 0 a 8 veces blanco).

Para ser más practico, imagina que en una escena regular, la intensidad de una luz (generalmente 1.0) se ajusta a 5.0. La escena completa se volverá muy brillosa (hacia el blanco) y lucirá horrible.

Luego de esto el luminance de la escena es computado al promediar el luminance de cada pixel que la conforma, y este valor es usado para traer la escena de vuelta a rangos normales. Esta última operación es llamada tone-mapping. Al final, estamos en un lugar similar a donde empezamos:


normal scene
light cranked up
tonemap

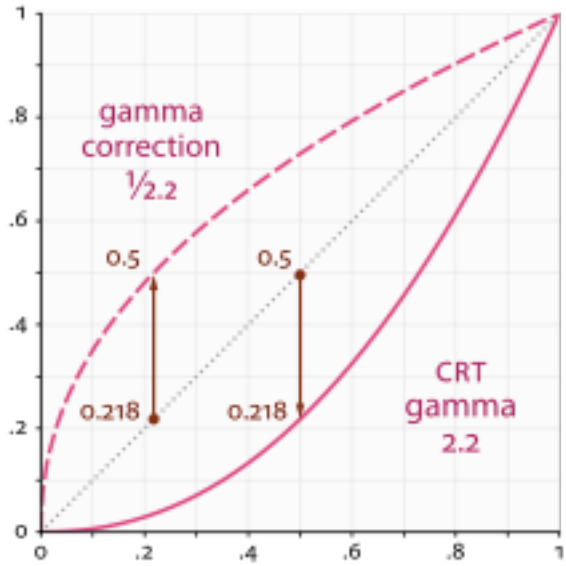
Excepto que la escena tiene más contraste, porque hay un mayor rango de iluminación en juego. Para que sirve esto? La idea es que el luminance de la escena va a cambiar mientras te mueves a través del mundo, permitiendo que sucedan situaciones como esta:

Además, es posible ajustar un valor threshold para enviar al glow buffer dependiendo en el luminance del pixel. Esto permite efectos de light bleeding más realistas en la escena.

Espacio de color lineal

El problema con esta técnica es que los monitores de computadora aplican una curva gamma para adaptarse mejor a la forma que ve el ojo humano. Los artistas crean su arte en la pantalla también, así que su arte tiene una curva gamma implícita aplicada.

El espacio de color donde las imágenes creadas en computadora existen es llamada "sRGB". Todo el contenido visual que la gente tiene en sus computadoras o descarga de Internet (como imagenes, peliculas, etc.) esta en este colorspace.



La matemática de HDR requiere que multipliquemos la escena por diferentes valores para ajustar el luminance y exposure a diferentes rangos de luz, y esta curva molesta ya que necesitamos colores en espacio linear para esto.

Espacio de color lineal & asset pipeling

Trabajar en HDR no es solo presionar un interruptor. Primero, los assets de imágenes importados deben convertirse a espacio linear al importarse. Hay dos formas de hacer esto:

SRGB -> linear conversion al importar imagen

Esta es la forma más compatible de usar assets de espacio-linear y funcionará en todos lados incluyendo todos los dispositivos moviles. El principal tema con esto es la pérdida de calidad, ya que sRGB existe para evitar este mismo problema. Usando 8 bits por canal para representar colores lineales es ineficiente desde el punto de vista del ojo humano. Estas texturas pueden comprimirse más tarde también, lo que vuelve al problema peor.

Pero en cualquier caso, esta es la solución fácil que funciona en todos lados.

sRGB por Hardware -> linear conversion

Esta es la forma mas correcta de usar assets en espacio-linear, ya que el sampler de texturas en la GPU hará la conversión luego de leer el texel usando punto flotante. Esto funciona bien en PC y consolas, pero la mayoría de los

dispositivos móviles no lo soporta, o no lo soportan en formato de textura comprimida (iOS por ejemplo).

Linear -> sRGB al final

Luego de que todo el renderizado está hecho, la imagen de espacio-linear renderizada debe ser convertida de nuevo a sRGB. Para hacer esto, solo habilita sRGB conversión en el *Environment* actual (mas sobre esto abajo).

Maten en mente que las conversiones sRGB [STRIKEOUT:> Linear and Linear]> sRGB deben siempre estar **ambas** habilitadas. Fallar al habilitar una de ellas resultara en visuales horribles adecuadas solo para juegos indie avant garde experimentales.

Parámetros de HDR

HDR se encuentra en el recurso *Environment*. Estos se encuentran la mayoría de las veces dentro de un nodo *WorldEnvironment*, o ajustado en una cámara. Hay muchos parámetros para HDR:

ToneMapper

El ToneMapper es el corazón del algoritmo. Muchas opciones para tonemappers son proveídas:

- Linear: El tonemapper más simple. Hace su trabajo para ajustar el brillo de la escena, pero si la diferencia en luz es muy grande, causara que los colores estén muy saturados.
- Log: Similar a linear, pero no tan extremo.
- Reinhardt: Tonemapper clásico (modificado para que no desature demasiado)
- ReinhardtAutoWhite: Igual que el anterior, peor usa el luminance de escena máximo para ajustar el valor blanco.

Exposure

El mismo parámetro de exposure que en cámaras reales. Controla cuanta luz entra a la cámara. Valores altos resultara en una escena mas brillante mientras valores bajos resultara en una escena más oscura.

White

Valor máximo de blanco.

Glow threshold

Determina luego de que valor (desde 0 a 1 luego que a la escena se le hace tonemapped), la luz empezara a hacer bleeding.

Glow scale

Determina cuanta luz hará bleeding.

Min luminance

Valor más bajo de luz para la escena en la cual el tonemapper dejara de trabajar. Esto permite que escenas oscuras permanezcan oscuras.

Max luminance

Valor más alto de luz para la escena en la cual el tonemapper dejara de trabajar. Esto permite que las escenas brillantes se mantengan saturadas.

Exposure adjustment speed

Auto-exposure cambiara lentamente y llevara un rato ajustarlo (como en las cámaras reales). Valores más altos significan ajustes más rápidos.

4.1.8 Rendimiento 3D y limitaciones

Introducción

Godot sigue una filosofía de rendimiento balanceado. En el mundo del rendimiento, siempre hay concesiones, que consisten en intercambiar velocidad por usabilidad y flexibilidad. Algunos ejemplos prácticos son:

- Renderizar objetos eficientemente en grandes cantidades es fácil, pero cuando una escena grande debe ser renderizada se puede volver ineficiente. Para resolver esto, la computación de la visibilidad debe ser agregado al renderizado, lo cual vuelve el renderizado menos eficiente, pero al mismo tiempo se renderizan menos objetos, entonces la eficiencia en su conjunto mejora.
- Configurar las propiedades de cada material para cada objeto que necesita ser renderizado también es lento. Para resolver esto, los objetos son ordenados por material para reducir los costos, pero al mismo tiempo ordenarlos tiene un costo.
- En física 3D sucede una situación similar. Los mejores algoritmos para manejar grandes cantidades de objetos físicos (como SAP) son muy lentos para insertar/remover objetos y hacer ray-casting. Los algoritmos que permiten inserción y remoción mas rápida, asi como ray-casting no serán capaces de manejar tantos objetos activos.

Y hay muchos ejemplos más de esto! Los motores de juegos se esfuerzan en ser de propósito general por naturaleza, así que los algoritmos balanceados siempre son favorecidos sobre algoritmos que pueden ser mas rápidos en algunas situaciones y lentos en otras.. o algoritmos que son rápidos pero vuelven la usabilidad más difícil.

Godot no es una excepción y, aunque está diseñado para tener backends intercambiables para diferentes algoritmos, los que están por defecto (o mejor dicho, los únicos que están allí por ahora) priorizan balance y flexibilidad sobre rendimiento.

Con esto aclarado, el objetivo de este tutorial es explicar cómo obtener la máxima performance de Godot.

Renderizado

El renderizado 3D es una de las áreas mas difíciles desde la cual obtener rendimiento, así que esta sección tendrá una lista de consejos.

Reusar shaders y materiales

El renderizador de Godot es un poco diferente a lo que hay por ahí. Esta diseñado para minimizar los cambios de estado de la GPU lo más posible. *FixedMaterial* hace un buen trabajo de reusar materiales que necesitan shaders similares pero, si un shader personalizado es usado, asegúrate de reusarlo lo mas posible. Las prioridades de Godot serán las siguientes:

- **Reusar Materiales**: Cuanto menor es la cantidad de materiales diferentes en una escena, mas rápido será el renderizado. Si una escena tiene una cantidad enorme de objetos (en los cientos o miles) intenta reusar los materiales o en el peor caso usa atlases.
- **Reusar Shaders**: Si los materiales no pueden ser reusados, al menos intenta reusar los shaders (o FixedMaterials con diferentes parámetros pero misma configuración).

Si una escena tiene, por ejemplo, 20.000 objetos con 20.000 diferentes materiales cada uno, el renderizado será realmente lento. Si la misma escena tiene 20.000 objetos, pero apenas usa 100 materiales, el renderizado será muy veloz.

Costo de pixel vs costo de vertex

Es un pensamiento común que cuanto menor cantidad de polígonos en un modelo, mas rápido será renderizado. Esto es *realmente* relativo y depende de muchos factores.

En PC y consolas modernas, el costo de vertex es bajo. Muy bajo. Las GPUs originalmente solo renderizaban triángulos, así que todos los vértices:

1. Tenían que ser transformados por la CPU (incluyendo clipping).
2. Tenían que ser enviadas a la memoria del GPU desde la memoria RAM.

Hoy en día, todo esto es manejado dentro de la GPU, así que la performance es extremadamente alta. Los artistas 3D usualmente tienen el pensamiento incorrecto sobre la performance por cantidad de polígonos debido a los DCCs 3D (como blender, Max, etc.) necesitan mantener la geometría en la memoria CPU para poder ser editada, reduciendo la performance real. La verdad es, un modelo renderizado por un motor 3D es mucho más optimo que como se muestran por los DCCs 3D.

En dispositivos móviles, la historia es diferente. Las GPU de PCs y consolas son monstruos de fuerza-bruta que pueden obtener tanta electricidad como necesiten de la red eléctrica. Las GPUs móviles están limitadas a una pequeña batería, por lo que necesitan ser mucho mas eficiente con la energía.

Para ser más eficientes, las GPUs móviles intentan evitar el *overdraw*. Esto significa, el mismo pixel en la pantalla siendo renderizado (con cálculos de luces, etc.) más de una vez. Imagina una ciudad con varios edificios, las GPUs realmente no saben que esta visible y que esta oculto hasta que lo dibujan. Una casa puede estar dibujada y luego otra

casa frente de la primera (el renderizado sucedió dos veces para el mismo pixel!). Las GPUs de PC normalmente no se preocupan mucho por esto y solo agregan mas procesadores de pixels al hardware para aumentar el rendimiento (pero esto también aumenta el consumo de poder).

En móvil, obtener más energía no es una opción, así que una técnica llamada "Tile Based Rendering" es usada (prácticamente todo hardware móvil usa una variante de dicha técnica), la cual divide la pantalla en una cuadricula (grid). Cada celda mantiene la lista de triángulos que se dibujan en ella y los ordena por profundidad para minimizar el *overdraw*. Esta técnica mejora el rendimiento y reduce el consumo de poder, pero impone una pena para el rendimiento vertex. Como resultado, menos vértices y triángulos pueden ser procesados para dibujar.

Generalmente, esto no es tan malo, pero hay un caso para móvil que debe ser evitado, que es tener objetos pequeños con mucha geometría dentro de una pequeña porción de la pantalla. Esto fuerza al GPU móvil a poner un gran esfuerzo en una sola celda de la pantalla, disminuyendo

Resumiendo, no te preocupes tanto por la cantidad de vértices en móvil, pero evita la concentración de vértices en partes pequeñas de la pantalla. Si, por ejemplo, un personaje, NPC, vehículo, etc. esta lejos (y luce minúsculo), usa un modelo con menor nivel de detalle (LOD).

Una situación extra donde el costo de vértices debe ser considerado son objetos que tienen procesamiento extra por vértice, como es:

- Skinning (skeletal animation)
- Morphs (shape keys)
- Vertex Lit Objects (común en móvil)

Compresión de texturas

Godot ofrece comprimir las texturas de los modelos 3D cuando se importan (compresión VRAM). La compresión Video RAM no es tan eficiente en tamaño como PNG o JPG al guardarse, pero aumenta enormemente el rendimiento cuando se dibuja.

Esto es debido a que el objetivo principal de la compresión de textura es la reducción del ancho de banda entre la memoria y la GPU.

En 3D, la forma de los objetos depende más de la geometría que la textura, por lo que la compresión en general no es aparente. En 2D, la compresión depende más de las formas dentro de las texturas, por lo que los artefactos resultantes de la compresión es más aparente.

Como una advertencia, la mayoría de los dipositivos Android no soportan compresión de texturas para texturas con transparencia (solo opacas), así que ten esto en cuenta.

Objetos transparentes

Como se mencionó antes, Godot ordena los objetos por material y shader para mejorar el rendimiento. Esto, sin embargo, no puede ser hecho en objetos transparentes. Los objetos transparentes son renderizados desde el fondo al frente para hacer que la mezcla con lo que esta atrás funcione. Como resultado, por favor trata de mantener los objetos transparentes al mínimo! Si un objeto tiene una sección pequeña con transparencia, trata de hacer esa sección un material separado.

Level of detail (LOD)

Como se mencionó antes, usar objetos con menos vértices puede mejorar el rendimiento en algunos casos. Godot tiene un sistema muy simple para usar nivel de detalle, los objetos basados en *GeometryInstance* tienen un rango de visibilidad que puede ser definido. Tener varios objetos GeometryInstance en diferentes rangos funciona como LOD.

Usar instanciamiento (MultiMesh)

Si varios objetos idénticos tienen que ser dibujados en el mismo lugar o cerca, intenta usar *MultiMesh* en su lugar. MultiMesh permite dibujar docenas de miles de objetos con muy poco costo de rendimiento, haciéndolo ideal para bandadas, pasto, partículas, etc.

Bake lighting

Las luces pequeñas usualmente no son un problema de rendimiento. Las sombras un poco más. En general, si varias luces necesitan afectar la escena, es ideal hacerles bake (doc_light_baking). Baking también puede mejorar la calidad de la escena al agregar rebotes de luz indirectos.

Si trabajas en móvil, hacer baking a las texturas es recomendado, dado que este método es mas rápido.

4.1.9 Trabajando con esqueletos 3D

El soporte de esqueletos 3D actualmente es bastante rudimentario. El nodo y clase *Skeleton* fueron diseñados principalmente para soportar la importación de animaciones por esqueleto como un conjunto de matrices de transformación.

Nodo Skeleton

Un nodo Skeleton puede ser directamente agregado en el lugar que quieras en la escena. Usualmente Mesh es hijo de Skeleton, ya que es más fácil de manipular de esta forma, porque las transformaciones en esqueletos son relativas a donde se encuentra. Pero puedes especificar nodo Skeleton en toda MeshInshatnce.

Siendo obvio, Skeleton está pensado para deformar mallas (meshes), y consiste de estructuras llamadas "bones"(huesos). Cada "bone" se representa como un Transform(transformación), la cual es aplicada a un grupo de vértices dentro de la malla. Puedes controlar directamente un grupo de vértices desde Godot. Para ello por favor ve la referencia en *MeshDataTool* class, method *set_vertex_bones*. Esta clase es muy poderosa.

Los "bones" son organizados en jerarquía, cada hueso, excepto por el/los hueso/huesos root(raíz) tienen padres. Cada hueso tiene un nombre asociado que podes usar para referirlo (ej: "raiz" o "mano.I", etc.). Además los huesos están numerados, esos números son IDs de huesos. La referencia de los huesos padre es por sus números ID.

Para el resto del articulo consideramos la siguiente escena:

```
main (Spatial) - el script siempre está aquí
== skel (Skeleton)
==== mesh (MeshInstance)
```

Esta escena esta importada de Blender. Contiene la malla arm(brazo) con 2 huesos - upperarm y lowerarm, y upperarm es el padre de lowerarm.

Clase Skeleton

Puedes ver la ayuda interna de Godot para descripciones de todas las funciones. Básicamente todas las operaciones en huesos son hechos usando su ID numérica. Puedes convertir de nombre a numero ID y vice-versa.

Para encontrar el número de huesos en un esqueleto usamos la función get_bone_count()

```
extends Spatial
var skel

func _ready():
```

```
    skel = get_node("skel")
    var id = skel.find_bone("upperarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
```

para encontrar el ID para el hueso, usa la función find_bone()

```
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("upperarm")
    print("bone id:", id)
```

Ahora, queremos hacer algo interesante con ID excepto imprimirlo. Además, podemos necesitar información adicional - para encontrar huesos padres para completar la cadena, etc. Esto se hace completamente con las funciones get/set_bone_*.

Para encontrar padres de hueso usamos la función get_bone_parent(id)

```
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("upperarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
```

Las transformaciones de huesos es el motivo por el cual estamos acá. Hay 3 tipos de transformaciones - local, global, custom (personalizada).

Para encontrar el Transform local usamos la función get_bone_pose(id)

```
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("upperarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
    var t = skel.get_bone_pose(id)
    print("bone transform: ", t)
```

Así que vemos una matriz 3x4 allí, la primer columna de 1s. Que podemos hacer sobre eso? Es un Transform, así que podemos hacer todo lo que se hace con Transform, básicamente traslación, rotación y escala. También podemos multiplicar transformaciones para obtener transformaciones complejas. Recuerda, los "bones" en Godot son solo Transforms sobre un grupo de vétrices. También podemos copiar Transforms de otros objetos aquí. Así que vamos a rotar nuestro hueso "upperarm":

```
extends Spatial
var skel
var id
```

```
func _ready():
    skel = get_node("skel")
    id = skel.find_bone("upperarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
    var t = skel.get_bone_pose(id)
    print("bone transform: ", t)
    set_process(true)

func _process(dt):
    var t = skel.get_bone_pose(id)
    t = t.rotated(Vector3(0.0, 1.0, 0.0), 0.1 * dt)
    skel.set_bone_pose(id, t)
```

Ahora podemos rotar huesos individuales. Lo mismo sucede para escala y traslación - inténtalos por tu cuenta y ve los resultados.

Lo que usamos ahora fue local pose. Por defecto todos los huesos no están modificados. Pero este Transform no nos dice nada sobre la relación entre huesos. Esta información es necesaria para un buen número de cosas. Como lo podemos obtener? Aquí viene global transform:

Para encontrar el Transform global del hueso usamos la función get_bone_global_pose(id)

Vamos a encontrar el Transform global para el hueso lowerarm:

```
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("lowerarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
    var t = skel.get_bone_global_pose(id)
    print("bone transform: ", t)
```

Como puedes ver, este transform no está en ceros. Aunque se llama global, en realidad es relativo al origén del Skeleton. Para el hueso raíz, el origén siempre esta en 0 si no fue modificado. Vamos a imprimir el origén de nuestro hueso lowerarm:

```
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("lowerarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
    var t = skel.get_bone_global_pose(id)
    print("bone origin: ", t.origin)
```

Vas a ver un número. Que significa este número? Es un punto de rotación de Transform. Así que es la parte base del hueso. En Blender, puedes ir a modo Pose e intentar allí rotar los huesos - van a rotar al rededor de su origen. Pero que hay sobre el extremo del hueso? No podemos saber cosas como el largo del hueso, el cual es necesario para

muchas cosas, sin saber la ubicación del extremo. Para todos los huesos en cadena excepto el ultimo podemos calcular la ubicación del extremo - es simplemente el origen de un hueso hijo. Sí, hay situaciones donde esto no es cierto, para huesos no conectados. Pero eso está OK por ahora, ya que no es importante respecto a los Transforms. Pero el extremo de leaf bone no se puede encontrar. Leaf bone es un hueso sin hijo. Por lo que no tienes ninguna información sobre su extremo. Pero esto no es tan grave. Puedes superarlo ya sea agregando un hueso extra a la cadena o simplemente calculando el largo del leaf bone en Blender y guardando su valor en tu script.

Usando "bones" 3D para control de malla

Ahora que ya sabes lo básico podemos aplicar esto para hacer FK-control completo de nuestro brazo (FK es forward-kinematics)

Para controlar completamente nuestro brazo necesitamos los siguientes parámetros:

- Upperarm angle x, y, z
- Lowerarm angle x, y, z

Todos estos parámetros pueden ser ajustados, incrementados y reducidos.

Crea el siguiente árbol de nodos:

```
main (Spatial) <- el script esta aquí
+-arm (arm scene)
+ DirectionLight (DirectionLight)
+ Camera
```

Ajusta una Camera de tal forma que el brazo este adecuadamente visible. Rota DirectionLight así ese brazo es apropiadamente iluminado cuando esta en modo de reproducción de escena.

Ahora necesitamos crear un nuevo script debajo de main:

Primero ajustamos los parámetros:

```
var lowerarm_angle = Vector3()
var upperarm_angle = Vector3()
```

Ahora necesitamos configurar una forma de cambiarlos. Usemos las teclas para eso.

Por favor crea 7 acciones en la configuración de proyecto:

- **selext_x** - bind to X key
- **selext_y** - bind to Y key
- **selext_z** - bind to Z key
- **select_upperarm** - bind to key 1
- **select_lowerarm** - bind to key 2
- **increment** - bind to key numpad +
- **decrement** - bind to key numpad -

Así que ahora queremos ajustar los parámetros de arriba. Entonces vamos a crear código para que lo haga:

```
func _ready():
    set_process(true)
var bone = "upperarm"
var coordinate = 0
func _process(dt):
    if Input.is_action_pressed("select_x"):
```

```
        coordinate = 0
    elif Input.is_action_pressed("select_y"):
        coordinate = 1
    elif Input.is_action_pressed("select_z"):
        coordinate = 2
    elif Input.is_action_pressed("select_upperarm"):
        bone = "upperarm"
    elif Input.is_action_pressed("select_lowerarm"):
        bone = "lowerarm"
    elif Input.is_action_pressed("increment"):
        if bone == "lowerarm":
            lowerarm_angle[coordinate] += 1
        elif bone == "upperarm":
            upperarm_angle[coordinate] += 1
```

El código completo para control de brazo es este:

```
extends Spatial

# member variables here, example:
# var a=2
# var b="textvar"
var upperarm_angle = Vector3()
var lowerarm_angle = Vector3()
var skel

func _ready():
    skel = get_node("arm/Armature/Skeleton")
    set_process(true)
var bone = "upperarm"
var coordinate = 0
func set_bone_rot(bone, ang):
    var b = skel.find_bone(bone)
    var rest = skel.get_bone_rest(b)
    var newpose = rest.rotated(Vector3(1.0, 0.0, 0.0), ang.x)
    var newpose = newpose.rotated(Vector3(0.0, 1.0, 0.0), ang.y)
    var newpose = newpose.rotated(Vector3(0.0, 0.0, 1.0), ang.z)
    skel.set_bone_pose(b, newpose)

func _process(dt):
    if Input.is_action_pressed("select_x"):
        coordinate = 0
    elif Input.is_action_pressed("select_y"):
        coordinate = 1
    elif Input.is_action_pressed("select_z"):
        coordinate = 2
    elif Input.is_action_pressed("select_upperarm"):
        bone = "upperarm"
    elif Input.is_action_pressed("select_lowerarm"):
        bone = "lowerarm"
    elif Input.is_action_pressed("increment"):
        if bone == "lowerarm":
            lowerarm_angle[coordinate] += 1
        elif bone == "upperarm":
            upperarm_angle[coordinate] += 1
    elif Input.is_action_pressed("decrement"):
        if bone == "lowerarm":
            lowerarm_angle[coordinate] -= 1
```

```
        elif bone == "upperarm":
            upperarm_angle[coordinate] -= 1
    set_bone_rot("lowerarm", lowerarm_angle)
    set_bone_rot("upperarm", upperarm_angle)
```

Presionando las teclas 1/2 seleccionas upperarm/lowerarm, selecciona el eje al presionar x, y, z, rota usando "+"/"-" en el teclado numérico.

De esta forma tu puedes controlar por completo el brazo en modo FK usando 2 huesos. Puedes agregar huesos adicionales y/o mejorar el "feel" de la interface usando coeficientes para el cambio. Recomiendo que juegues con este ejemplo un montón antes de ir a la siguiente parte.

Puedes clonar el código del demo para este capítulo usando

```
git clone git@github.com:slapin/godot-skel3d.git
cd demo1
```

O puedes navegarlo usando la interfaz web:

https://github.com/slapin/godot-skel3d

Usando "bones" 3D para implementar Inverse Kinematics

Ve *Inverse kinematics*.

Usando "bones" 3D para implementar física ragdoll

Para hacer.

4.1.10 Inverse kinematics

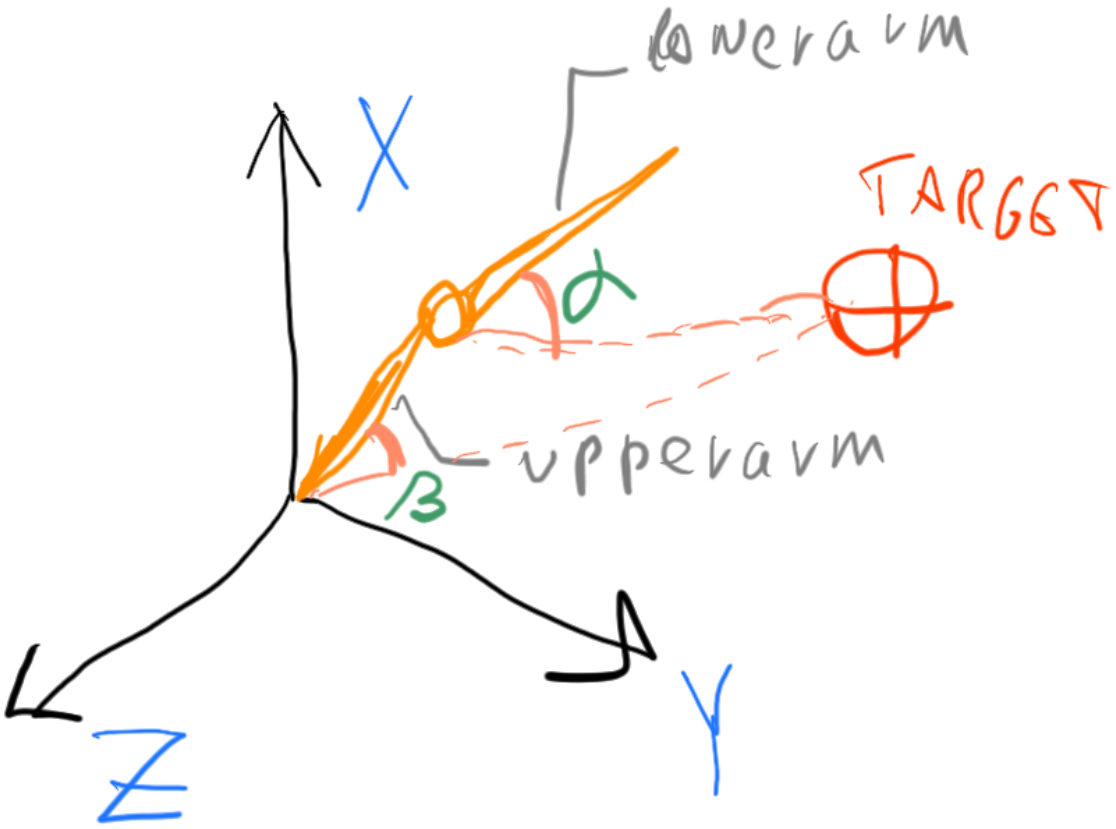
Este tutorial es una continuación de *Trabajando con esqueletos 3D*.

Antes de continuar, recomendaría leer algo de teoría, el artículo mas simple que encontré es este:

http://freespace.virgin.net/hugo.elias/models/m_ik2.htm

Problema inicial

Hablando en terminología Godot, la tarea que queremos resolver aquí es posicionar los 2 ángulos que hablamos antes, entonces, el extremo del hueso lowerarm esta tan cerca del target point, el cual está ajustado por el target Vector3() como posible usando solo rotaciones. Esta tarea es muy intensiva en calculo y nunca resuelta por una ecuación analítica. Entonces, es un problema sin restricciones, lo cual significa que hay un número ilimitado de soluciones a la ecuación.



Para simplificar los cálculos, en este capítulo consideramos que target también es hijo de Skeleton. Si este no es el caso para tu configuración, puedes siempre re-apadrinarlo en tu script, ya que ahorraras en cálculos si lo haces.

El eje de rotación es fácilmente calculado usando el producto cruzado del vector de hueso y el vector target (objetivo). La rotación en este caso va a ser siempre en dirección positiva. Si t es la Transform que obtenemos desde la función get_bone_global_pose(), el vector hueso es

```
t.basis[2]
```

Así que tenemos toda la información aquí para ejecutar nuestro algoritmo.

En el desarrollo de juegos es común resolver este problema acercándose de forma iterativa a la ubicación deseada, sumando/restando pequeños ángulos hasta que el cambio de distancia logrado es menos que algún pequeño valor de error. Suena bastante fácil, pero hay problemas en Godot que debemos resolver para lograr nuestro objetivo.

- **Como encontrar las coordenadas del extremo del hueso?**
- **Como encontrar el vector desde la base del hueso hasta el objetivo?**

Para nuestro propósito (extremo del hueso movido dentro del área del objetivo), necesitamos saber dónde está el extremo de nuestro hueso IK. Ya que no usamos un leaf bone como IK bone, sabemos que las coordenadas de la base del hueso es el extremo del hueso padre. Todos estos cálculos son bastante dependientes en la estructura del esqueleto. Pueden usar constantes pre-calculadas también. Puedes agregar un hueso extra para el extremo de IK y calcularlo usando eso.

Implementación

Vamos a usar la variable exportada para el largo del hueso para simplificarlo.

```
export var IK_bone="lowerarm"
export var IK_bone_length=1.0
export var IK_error = 0.1
```

Ahora, necesitamos aplicar nuestras transformaciones desde el hueso IK a la base de la cadena. Así que aplicamos rotación al hueso IK y luego mover desde nuestro hueso IK hasta su padre, y aplicar rotación nuevamente, luego mover hasta el padre del hueso actual nuevamente, etc. Así que necesitamos limitar nuestra cadena un poco.

```
export var IK_limit = 2
```

For `_ready()` function:

```
var skel
func _ready():
    skel = get_node("arm/Armature/Skeleton")
    set_process(true)
```

Ahora podemos escribir nuestra función de pasaje de cadena:

```
func pass_chain():
    var b = skel.find_bone(IK_bone)
    var l = IK_limit
    while b >= 0 and l > 0:
        print( "name":", skel.get_bone_name(b))
        print( "local transform":", skel.get_bone_pose(b))
        print( "global transform":", skel.get_bone_global_pose(b))
        b = skel.get_bone_parent(b)
        l = l - 1
```

Y para la función `_process()`:

```
func _process(dt):
    pass_chain(dt)
```

Ejecutando este script solo pasara a través de la cadena de huesos imprimiendo las transformaciones de huesos.

```
extends Spatial

export var IK_bone="lowerarm"
export var IK_bone_length=1.0
export var IK_error = 0.1
export var IK_limit = 2
var skel
func _ready():
    skel = get_node("arm/Armature/Skeleton")
    set_process(true)
func pass_chain(dt):
    var b = skel.find_bone(IK_bone)
    var l = IK_limit
    while b >= 0 and l > 0:
        print("name: ", skel.get_bone_name(b))
        print("local transform: ", skel.get_bone_pose(b))
        print( "global transform:", skel.get_bone_global_pose(b))
        b = skel.get_bone_parent(b)
```

```
        l = l - 1
func _process(dt):
    pass_chain(dt)
```

Ahora necesitamos trabajar con el objetivo. El objetivo debe estar ubicado en algún lugar accesible. Ya que "arm" es una escena importada, lo mejor es ubicar el nodo target dentro de nuestro nivel superior de escena. Pero para que podamos trabajar fácilmente con el objetivo su Transform debe estar en el mismo nivel que el Skeleton.

Para hacer frente a este problema creamos un nodo "target" bajo nuestra raíz de nodos de escena y cuando corra el script vamos a re apadrinarlo copiando la transformación global, lo cual logra el efecto deseado.

Crea un nuevo nodo Spatial bajo la raíz y renómbralo a "target". Luego modifica la función `_ready()` para que luzca así:

```
var skel
var target
func _ready():
    skel = get_node("arm/Armature/Skeleton")
    target = get_node("target")
    var ttrans = target.get_global_transform()
    remove_child(target)
    skel.add_child(target)
    target.set_global_transform(ttrans)
    set_process(true)
```

4.2 Physics

4.2.1 Usando gridmaps

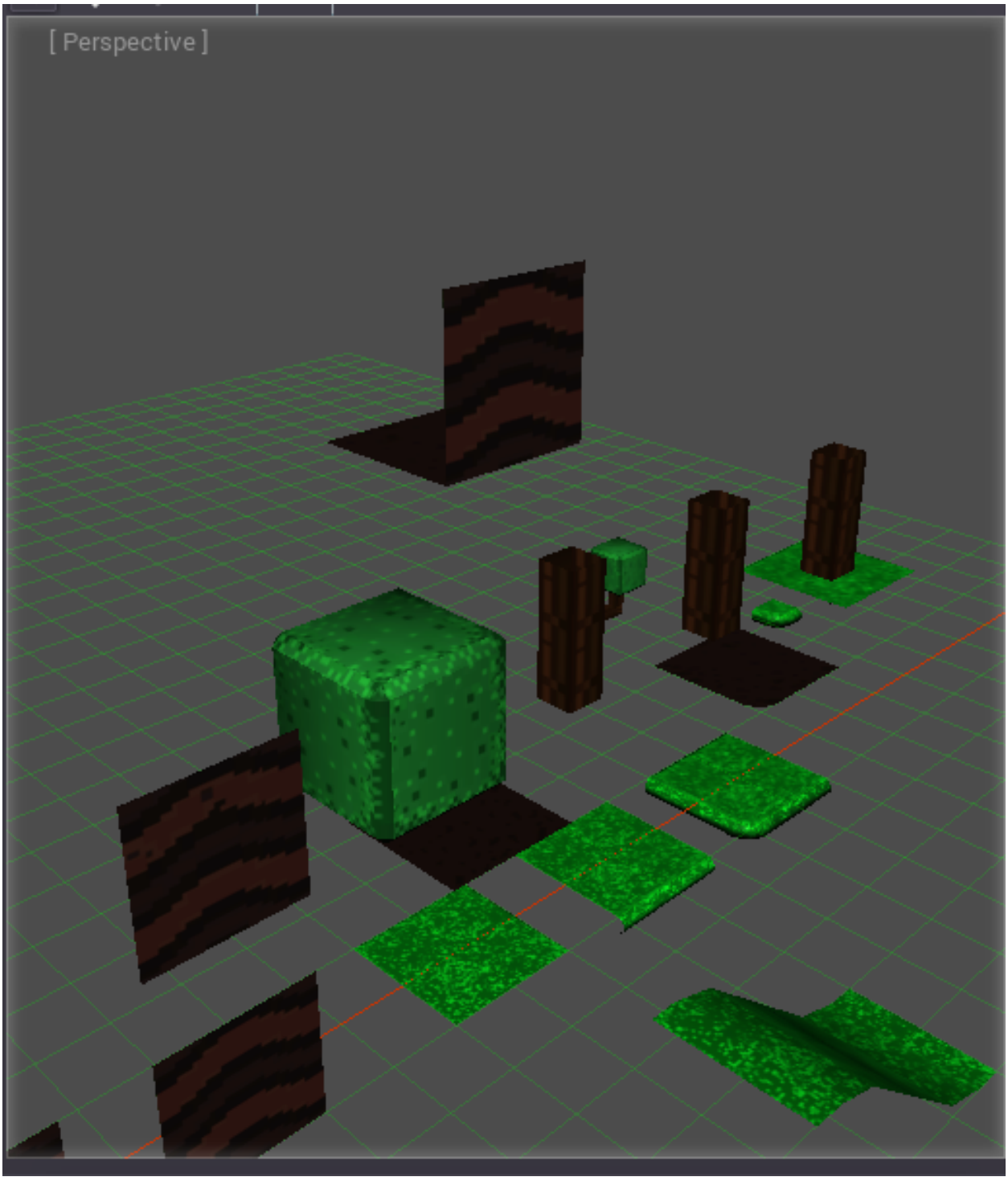
Introducción

Los *Gridmaps* son una forma simple y rápida de crear niveles para juegos 3D. Piensa de ellos como una versión 3D de los nodos *TileMap*. De forma similar, empiezas con una librería predefinida de mallas 3D que pueden ser puestas en un grid, justo como si estuvieras haciendo un nivel con una cantidad ilimitada de bloques Lego.

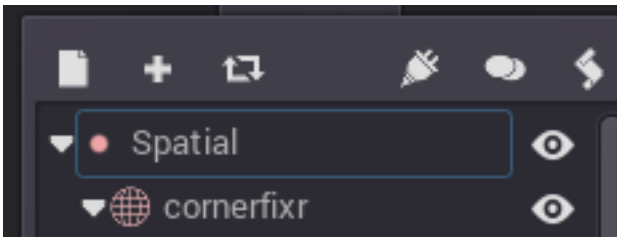
Las colisiones también pueden ser agregadas a las mallas, como harías con los mosaicos de un tilemap.

Creando un MeshLibrary

Para empezar, necesitas un *MeshLibrary*, el cual es una colección de mallas que pueden ser usadas en un gridmap. Aquí hay algunas mallas que puedes usar para configurarlo.



Abre una nueva escena y crea el nodo raíz (esto es importante, ya que sin nodo raíz, no será capaz de generar el MeshLibrary!). Luego, crea un nodo *MeshInstance*:

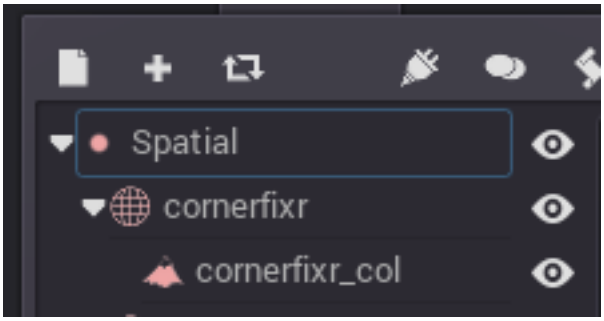


Si no necesitas aplicar física a los bloques de construcción, eso es todo lo que tienes que hacer. En la mayoría de los casos sin embargo, vas a necesitar que tu bloque genere colisiones, así que veamos como agregarlas.

Colisiones

Para asignar una *CollisionShape* y *PhysicsBody* a las mallas, la forma más simple es hacerlo mientras creas el MeshLibrary. De forma alternativa, también puedes editar una MeshLibrary existente desde dentro del inspector de GridMap, pero solo CollisionShapes pueden ser definidos allí y no PhysicsBody.

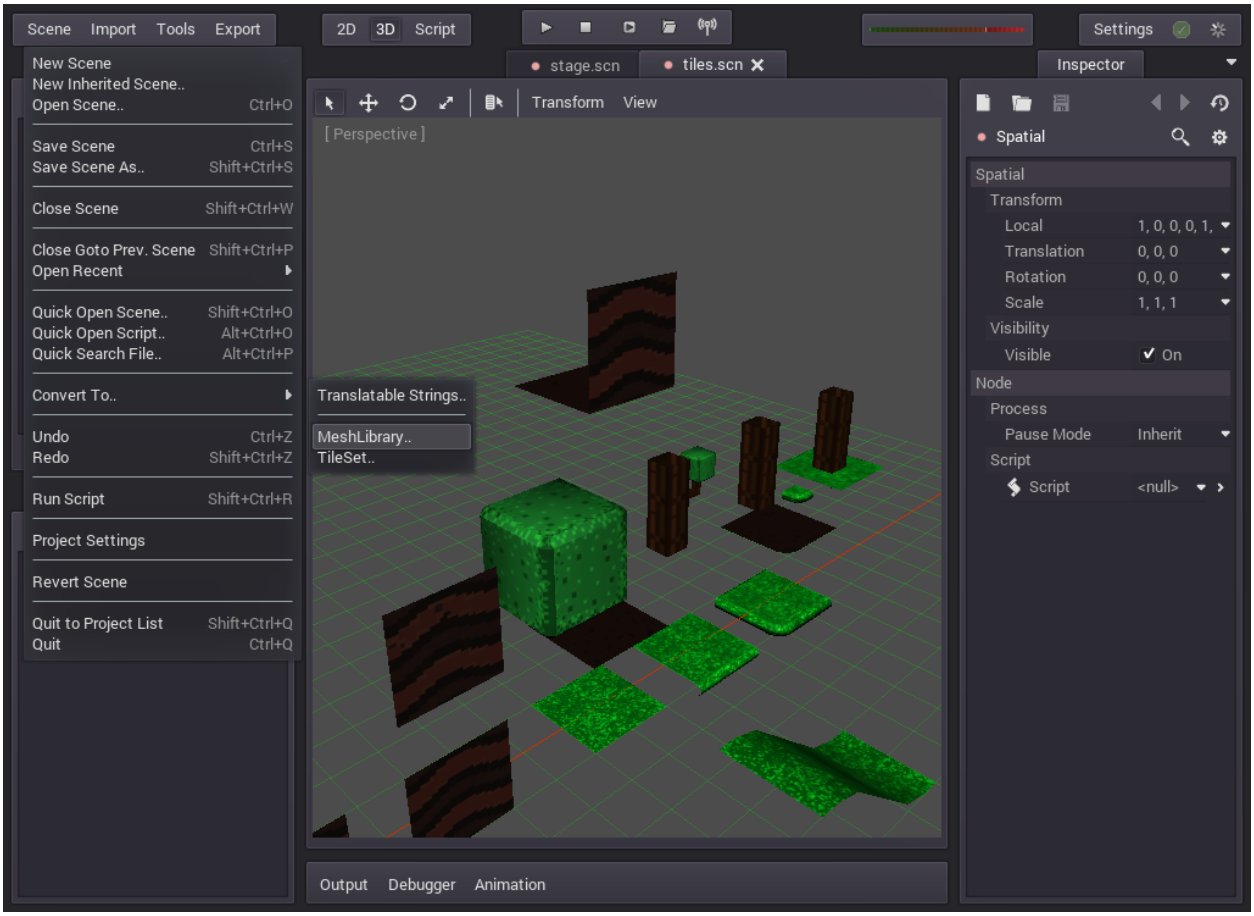
Para darle a las mallas un CollisionShape, simplemente agrega nodos hijos al nodo MeshInstance. Típicamente se agregan los PyhisicsBody y CollisionShape necesarios en este orden:



Puedes ajustar el orden de acuerdo a tus necesidades.

Exportando la MeshLibrary

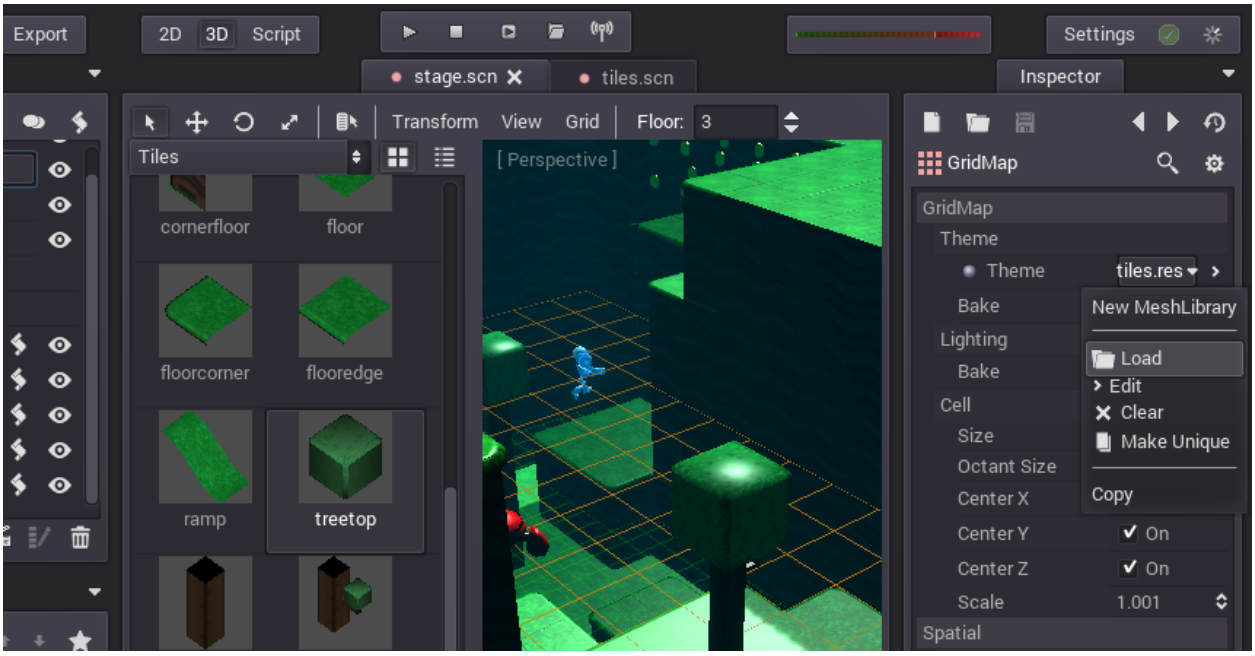
Para exportar, ve a `Escena > Convertir a.. > MeshLibrary...` y la guardas como un recurso.



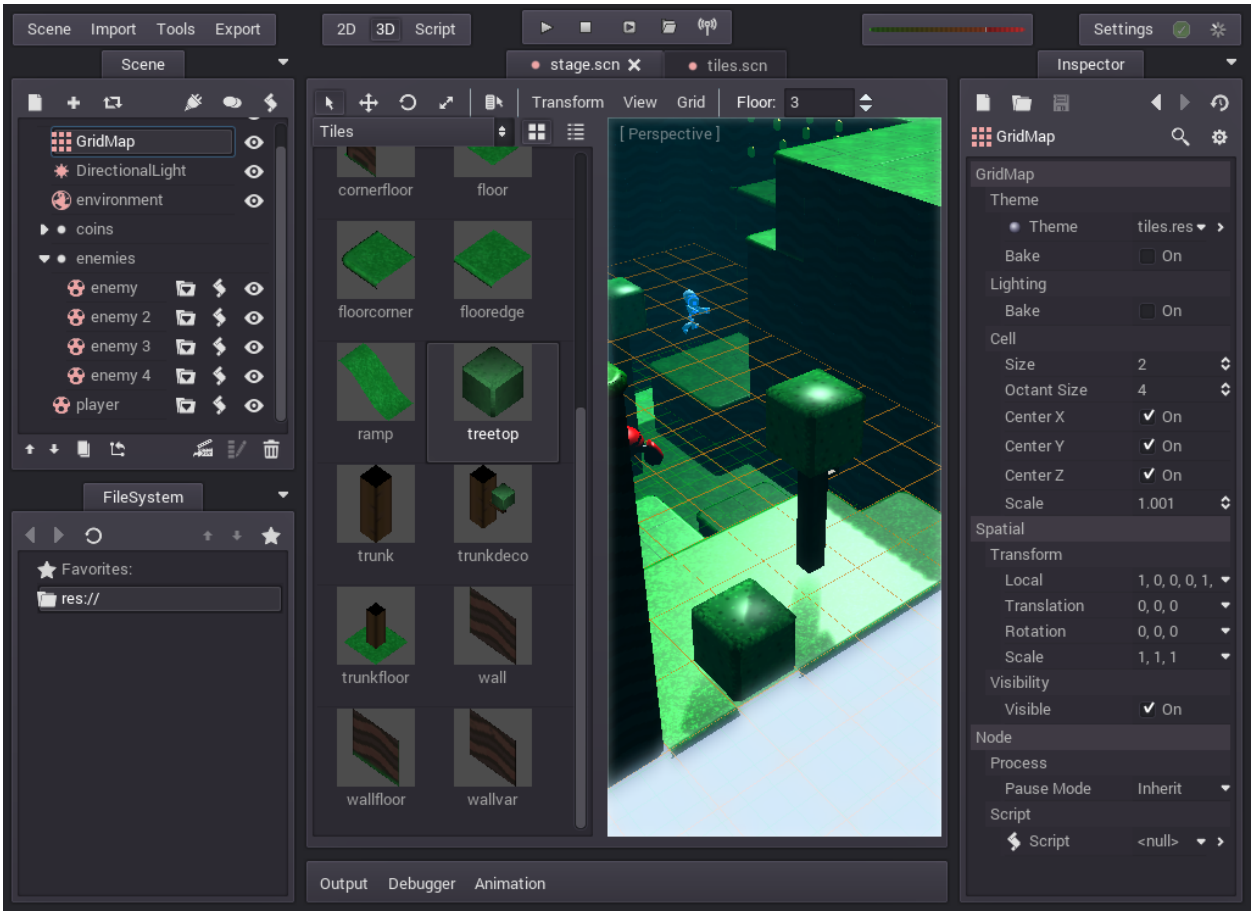
Ahora estás listo para usar el nodo GridMap.

Usando MeshLibrary en un GridMap

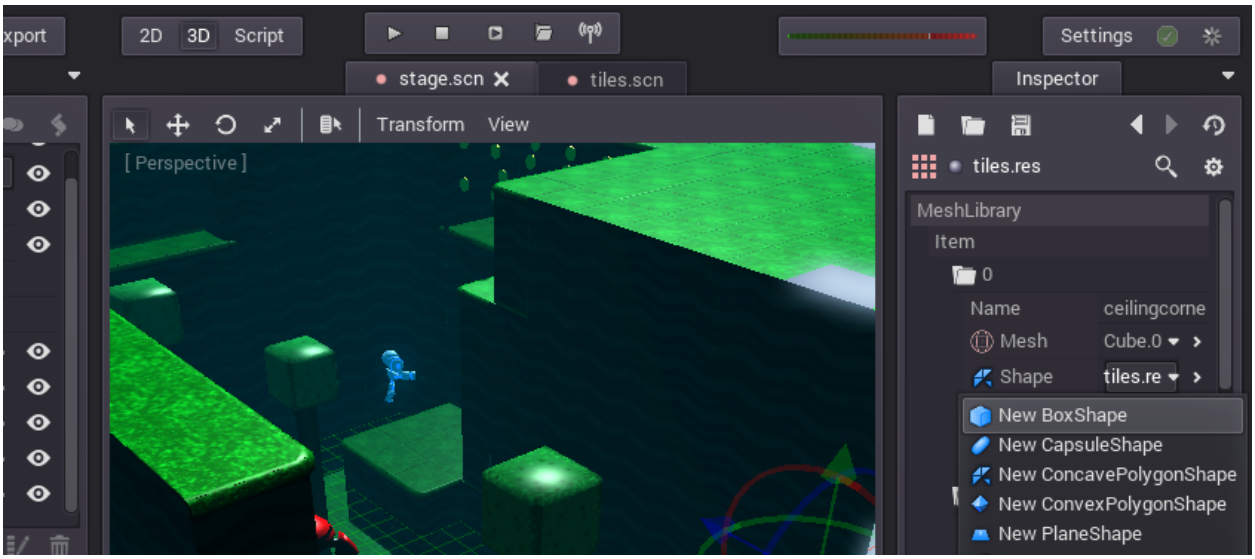
Crea una nueva escena usando cualquier nodo como raíz, luego agrega un nodo GridMap. Despues, carga el MeshLibrary que recién exportaste.



Ahora, puedes construir tu propio nivel como te parezca mejor. Usa clic izquierdo para agregar mosaicos y botón derecho para quitarlos. Puedes ajustar el nivel del piso cuando necesitas poner mallas en alturas específicas.



Como mencionamos arriba, también puedes definir nuevas CollisionShapes en esta etapa siguiendo estos pasos:



Lo lograste!

Recordatorio

- Se cuidadoso antes de escalar mallas si no estás usando mallas uniformes.
- Hay muchas formas de usar gridmaps, se creativo!

4.3 Importar

4.3.1 Importando mallas 3D

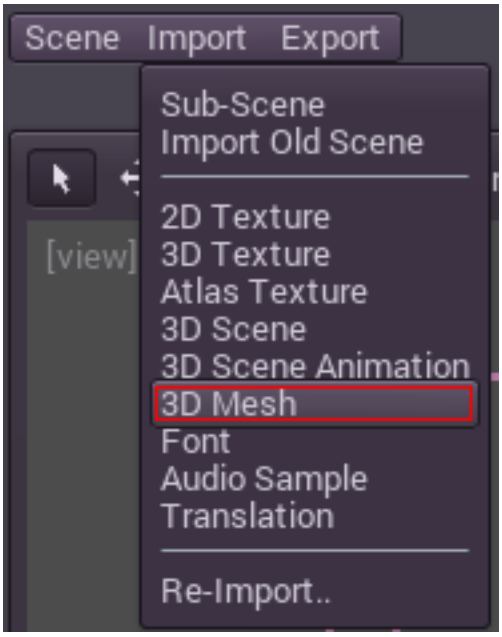
Introducción

Godot soporta un importador de escena flexible y poderoso *3D scene importer* que permite importar una escena completa. Para un montón de artistas y desarrolladores esto es mas que suficiente. Sin embargo, a muchos no les gusta este workflow tanto y prefieren importar 3D Meshes individualmente y construir la escena dentro del editor 3D de Godot. (Nota que para características más avanzadas como animación por esqueletos, no hay opción en el 3D Scene Importer).

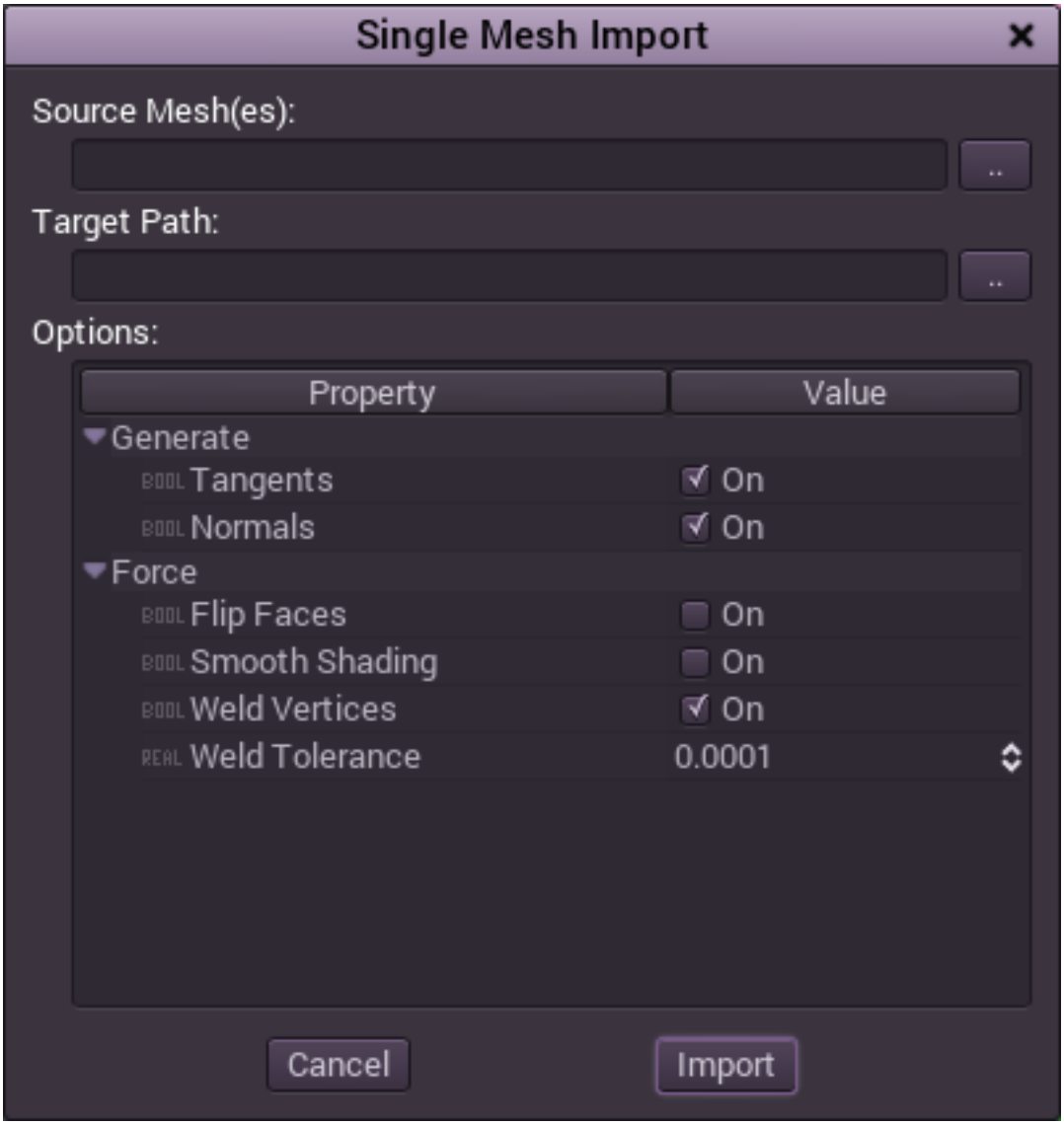
El workflow de importación de malla 3D es simple y funciona usando el formato de archivo OBJ. Las mallas importadas resultan en un archivo binario .msh que el usuario puede poner en un *MeshInstance*, el cual a su vez es puede ser ubicado en algún lugar de la escena editada.

Importando

La importación es hecha a través del menú Importar > Mesh:



El cual abre la ventana de importación de Mesh:



Este dialogo permite la importación de uno o más archivos OBJ en un target path. Los archivos OBJ son convertidos a archivos .msh. Son importados sin material en ellos, el material debe ser agregado por el usuario (ve el tutorial *Fixed Materials (Materiales Fijos)*). Si el archivo externo OBJ cambia será re-importado, manteniendo el material recientemente asignado.

Opciones

Algunas opciones están presentes. Las Normals son necesarias para sombreado regular, mientras que Tangents son necesarias si planificas usar normal-mapping en el material. En general, los archivos OBJ describen como se debe sombrear muy bien, pero una opción para forzar smooth shading está disponible.

Finalmente, hay una opción para soldar vértices. Dado que los archivos OBJ son basados en texto, es común encontrar algunos de esos vértices que no cierran, lo que resulta en sombreado extraño. La opción weld vertices mezcla vértices que están demasiado cercanos para mantener un sombreado suave adecuado.

Uso

Los recursos de Mesh (lo que este importador importa hacia) son usados dentro de nodos MeshInstance. Simplemente ajusta la propiedad Mesh de ellos.

Y eso es todo.

4.3.2 Importando escenas 3D

Introducción

La mayoría de los motores de juegos solo importan objetos 3D, los cuales pueden contener esqueletos o animaciones, y luego todo el demás trabajo es hecho in la UI del motor, como la ubicación de los objetos, animaciones de pantalla completa, etc. En Godot, dado que el sistema de nodos es muy similar a cómo funcionan las herramientas 3D DCC (como Maya, 3DS Max o Blender), escenas completas en 3D pueden ser importadas en toda su gloria. Adicionalmente, al usar un lenguaje simple de etiquetas, es posible especificar que objetos son importados para diferentes cosas, como colisionables, cuartos y portales, vehículos y ruedas, distancias LOD, billboards, etc.

Esto permite algunas características interesantes:

- Importar escenas simples, objetos con rig, animaciones, etc.
- Importar escenas completas. Escenarios enteros pueden ser creados y actualizados en el 3D DCC e importados a Godot cada vez que cambian, luego apenas un poco de edición es necesaria del lado del motor.
- Cutscenes completas pueden ser importadas, incluyendo animaciones de múltiples personajes, iluminación, movimiento de cámara, etc.
- Las escenas pueden continuar siendo editadas y scripteadas en el motor, donde los shaders y efectos de ambiente pueden ser agregados, enemigos ser instanciados, etc. El importador va a actualizar los cambios de geometría si la escena de origen cambia pero mantendrá los cambios locales también (en tiempo real mientras usas el editor Godot!)
- Las texturas pueden ser todas importadas en lotes y actualizadas cuando la escena de origen cambia.

Esto se logra usando un lenguaje de etiquetas muy simple que será explicado en detalle a continuación.

Exportando archivos DAE

Porque no FBX?

La mayoría de los motores de juegos usan el formato FBX para importar escenas 3D, el cual definitivamente es uno de los más estandarizados en la industria. Sin embargo, este formato requiere el uso de una librería cerrada de Autodesk la cual es distribuida con términos de licenciamiento más restrictivos que Godot. El plan es, en algún momento del futuro, implementar una conversión binaria externa, pero mientras tanto FBX no está soportado.

Exportando archivos DAE desde Maya y 3DS Max

Autodesk agrego soporte incorporado collada a Maya y 3DS Max, pero esta realmente roto y no debería ser usado. La mejor forma de exportar este formato es usando los plugins OpenCollada Estos funcionan realmente bien, aunque no siempre están al día con la ultima versión del software.

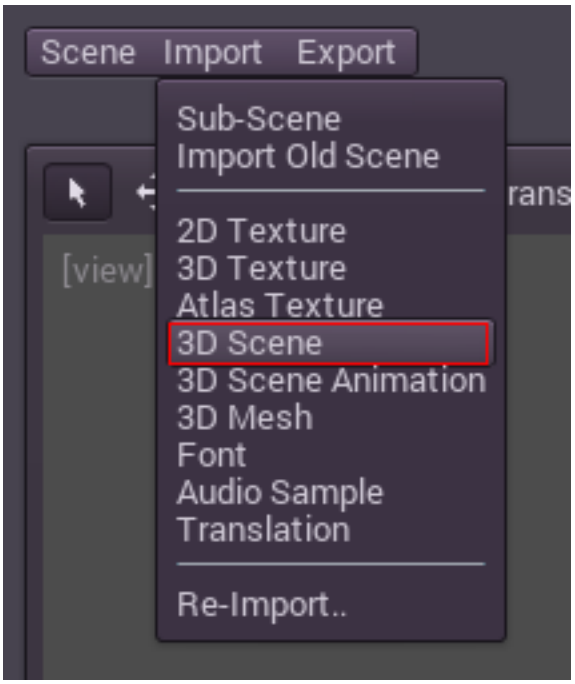
Exportando archivos DAE desde Blender

Blender también tiene soporte collada incorporado, pero está realmente roto y no debería ser usado tampoco.

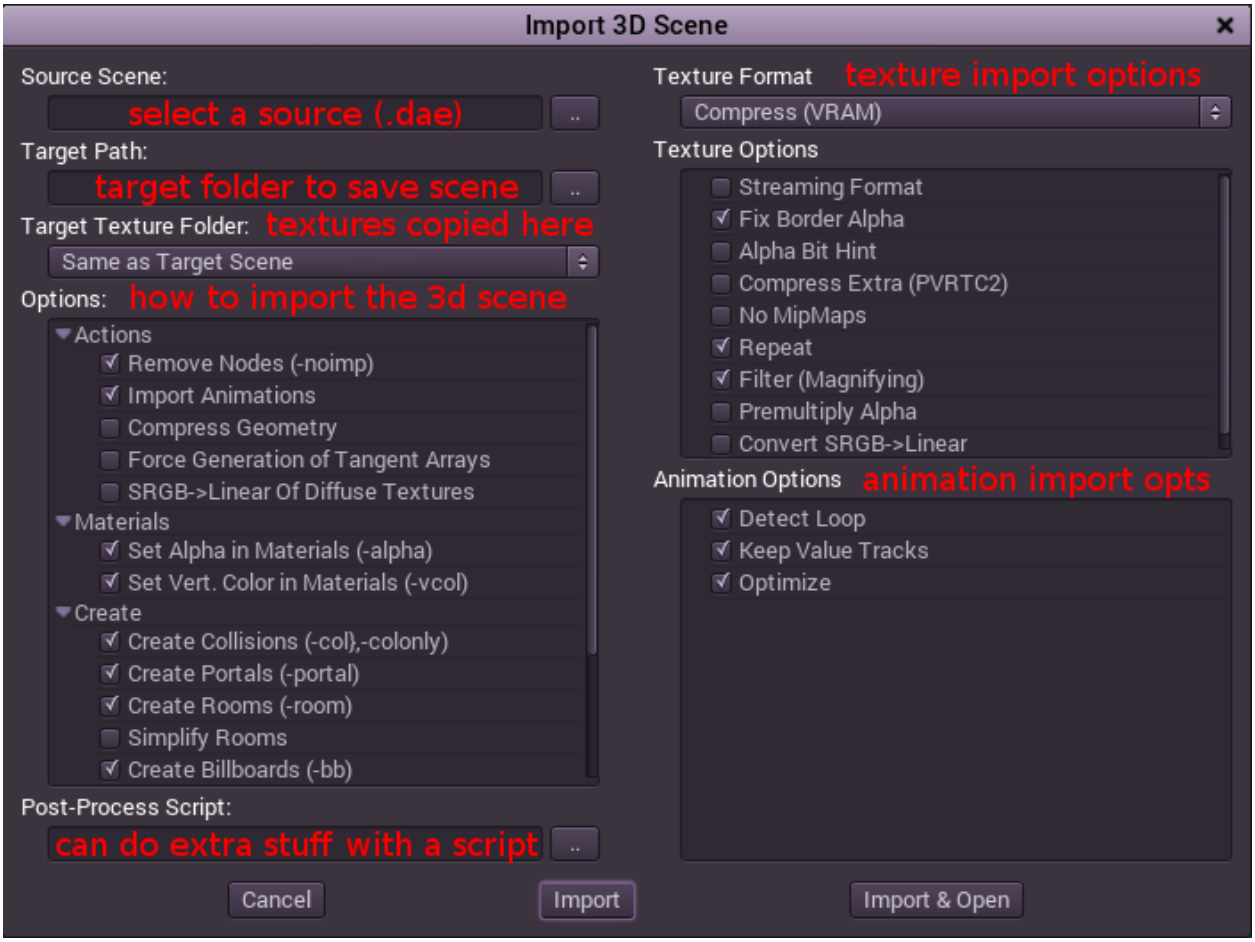
Godot provee un Python Plugin que hará un trabajo mucho mejor de exportar las escenas.

El proceso de importación

El proceso de importación comienza con el menú de importar escena 3D:



Esto abre lo que es probablemente el más grande dialogo de importación de todos:



Muchas opciones existen allí, así que cada sección será explicada a continuación:

Source & target paths

Para importar, dos opciones se necesitan. La primera es el archivo .dae fuente (.dae es por Collada. Mas formatos de importación se agregaran eventualmente, pero Collada es el formato abierto más completo cuando esto escribió).

Un target folder necesita proveerse, entonces el importador puede importar la escena allí. La escena importada tendrá el mismo nombre de archivo que la fuente, excepto por la extensión .scn, así que asegúrate de elegir buenos nombres cuando exportas!

Las texturas usadas serán copiadas y convertidas. Las texturas en aplicaciones 3D son usualmente solo archivos PNG o JPG. Godot los convertirá a formato de compresión de textura de memoria de video(s3tc, pvrtc, ericsson, etc.) por defecto para mejorar el rendimiento y ahorrar recursos.

Debido a que las texturas originales, archivo 3D y texturas no son usualmente necesarias, es recomendado mantenerlas fuera del proyecto. Para algunos consejos sobre cómo hacer esto de la mejor forma, puedes chequear el tutorial *Organización de proyectos*.

Dos opciones para texturas se proveen. Pueden ser copiadas al mismo lugar que la escena, o pueden copiarse a un path común (ajustable en la configuración de proyecto). Si eliges esto, asegúrate que no haya dos texturas con el mismo nombre.

Consejos de Rigging 3D

Antes de ir a las opciones, aquí hay algunos consejos para asegurarte que tus rigs se importen adecuadamente

- Solo se importan hasta 4 weights por vertex, si un vertex depende de más de 4 huesos, solo los 4 más importantes (los que tienen más peso) van a ser importados. Para la mayoría de los modelos esto funciona usualmente bien, pero tenlo en mente.
- No uses animaciones de hueso con escala no uniforme, ya que esto es probable que no se importe adecuadamente. Intenta lograr el mismo efecto agregando huesos.
- Cuando exportas desde Blender, asegúrate que los objetos modificados por un esqueleto sean hijos de el. Muchos objetos pueden ser modificados por un solo esqueleto, pero todos deben ser hijos directos.
- De la misma forma, cuando uses Blender, asegúrate que la transformación relativa de los nodos hijos al esqueleto sea cero (sin rotación, sin translación, sin escala. Todos ceros y con una escala de 1.0). La posición de ambos objetos (el pequeño punto naranja) debe estar en el mismo lugar.

Opciones de importación 3D

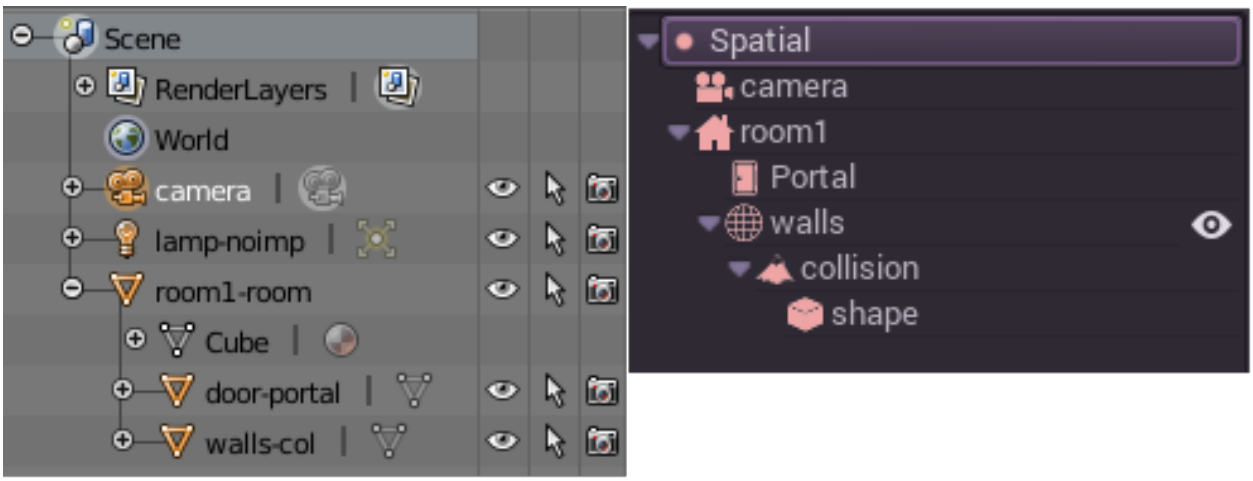
Esta sección contiene muchas opciones para cambiar la forma en que el workflow de importación funciona. Algunos (como HDR) serán mejor explicadas en otras secciones, pero en general un patrón puede ser visible en las opciones y esto es, muchas de las opciones terminan con "-something". Por ejemplo:

- Remover Nodos (-noimp)
- Ajustar Alpha en Materiales (-alpha)
- Crear colisiones (-col)

Esto significa que los nombres de objeto en el DCC 3D necesitan tener esas opciones agregadas al final para que el importador sepa que son. Cuando se importan, Godot las convertirá a lo que están destinadas a ser.

Nota: Usuarios de Maya deben usar "_" (underscore) en lugar de "-" (minus).

Aquí hay un ejemplo de como una escena en el DCC 3D luce (usando Blender), y como se importa en Godot:



Fíjate que:

- La cámara se importó normalmente.
- Un cuarto fue creado (-room).
- Un portal fue creado (-portal).

- El Mesh tuvo agregado de colisión estática (-col).
- La luz no se importo (-noimp).

Opciones en detalle

A continuación una lista de las opciones más importantes y lo que hacen con mas detalle.

Remove nodes (-noimp)

Los nombres de nodos que tengan esto al final serán removidos en tiempo de importación, no importa su tipo. Borrarlos luego no tiene sentido la mayoría de las veces porque serán restauradas si la escena fuente cambia.

Import animations

Algunos formatos de escena (.dae) soportan una o más animaciones. Si esto es chequeado, un nodo AnimationPlayer será creado, conteniendo las animaciones.

Compress geometry

Esta opción (deshabilitada [STRIKEOUT:o mas bien, siempre habilitada] al momento de escribir esto) va a comprimir la geometría de forma que tome menos espacio y renderize mas rápido (al costo de menos precisión).

Force generation of tangent arrays

El importador detecta cuando has usado una textura normalmap, o cuando el archivo fuente contiene información de tangentes/binormales. Estos arreglos son necesarios para que funcione normalmapping, y la mayoría de los exportadores saben lo que hacen cuando exportan esto. Sin embargo, es posible encontrarse con escenas que no tienen esta información lo cual, como resultado, hace que normal-mapping no funcione. Si notas que los normal-maps no funcionan cuando importas la escena, prende esto!

SRGB -> linear of diffuse textures

Cuando renderizas usando HDR (High Dynamic Range) puede ser deseable usar texturas linear-space para lograr iluminación más real. De otra forma, los colores pueden saturar y contrastar demasiado cuando cambia la exposición. Esta opción debe ser usada junto con SRGB en WorldEnvironment. Las opciones de importación de textura también tienen la opción para hacer esta conversión, pero si esta propiedad esta encendida, las conversión siempre será hecha a las texturas difusas (usualmente lo que se desea). Para mas información, lee el tutorial *High dynamic range*.

Set alpha in materials (-alpha)

Cuando trabajas con la mayoría de los DCCs 3D, es bastante obvio cuando una textura es transparente y tiene opacidad lo cual raramente afecta el workflow del renderizado final. Sin embargo, cuando tratas con renderizado en tiempo real, los materiales con alpha blending son usualmente menos óptimos para dibujar, por lo que deben ser marcados específicamente como tales.

Originalmente Godot detectaba esto basado en si la textura fuente tenia un canal alpha, pero la mayoría de las aplicaciones de manipulación de imágen como Photoshop o Gimp van a exportar este canal de todas formas aun si no es

usado. Se agregó código más tarde para chequear manualmente si realmente había alguna transparencia en la textura, pero los artistas de todas formas muy a menudo dejan uvmaps en partes opacas de la textura y otras áreas sin usar (donde no existe UV) transparentes, volviendo esta detección sin sentido.

Finalmente, se decidió que es mejor importar todo como opaco y dejar a los artistas solucionar los materiales que necesitan transparencia cuando es obvio que no lucen bien (ve el tutorial *Materiales*).

Como un ayudante, dado que todo DCC 3D permite nombrar los materiales y mantener su nombre al exportar, el modificador (-alpha) en su nombre apuntara al importador de escena 3D de Godot que ese material va a usar el canal alpha para transparencia.

Set vert. color in materials (-vcol)

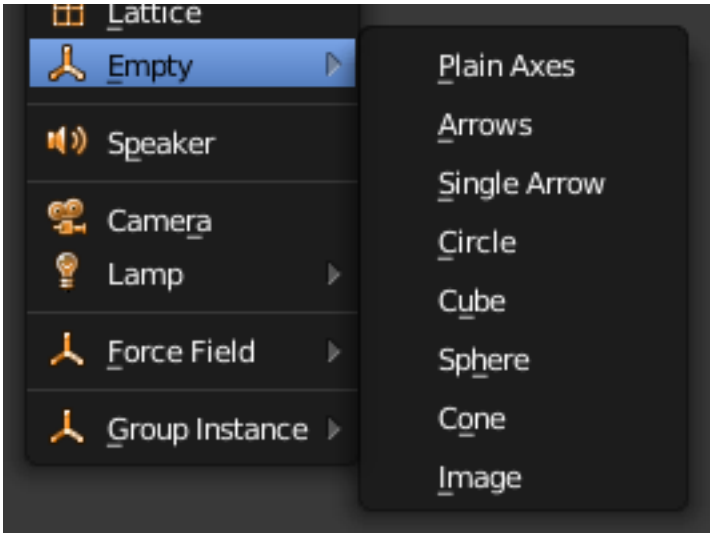
La mayoría de los DCC 3D soportan pintado por vertex. Esto es generalmente aplicado como una multiplicación o mezcla de pantalla. Sin embargo, a menudo se presenta el caso de que tu exportador va a exportar esta información como 1s, o exportarla como alguna otra cosa y no te darás cuenta. Debido a que en la mayoría de los casos esta opción no es deseada, solo agrega esto a cualquier material para confirmar que se desea usar vertex colors.

Create collisions (-col, -colonly)

La opción "-col" solo funcionara para nodos Mesh. Si es detectada, un nodo hijo de colisión estática será agregado, usando la misma geometría que la malla.

Sin embargo, a menudo sucede que la geometría visual es demasiado compleja o muy poco suave para colisiones, lo que termina no funcionando bien. Para resolver esto, existe el modificador -"colonly", el cual remueve la malla cuando se importa y en su lugar crea una colisión StaticBody. Esto ayuda a separar la malla visual y colisión.

La opción "-colonly" puede también ser usada con objetos vacíos de Blender. Al importar creara un StaticBody con nodos de colisión como hijos. Los nodos de colisión serán de las formas predefinidas, dependiendo en el tipo de empty draw de Blender:



- Una flecha crea RayShape
- El cubo crea BoxShape
- Una imagen crea PlaneShape
- Una esfera (y otros no listados) crea SphereShape

Para mejor visibilidad en el editor de Blender el usuario puede ajustar la opción "on colision empties" y ajustar algún color distintivo para ellas en User Preferences / Themes / 3D View / Empty.

Create rooms (-room)

Esto es usado para crear un cuarto. Como regla general, cualquier nodo que es un hijo de este nodo será considerado dentro del cuarto (incluyendo portales).

Para más información sobre rooms/portals, mira el tutorial [[Portals and Rooms]].

Hay dos formas posibles para usar este modificador. La primera es usando un nodo Dummy/Empty en la aplicación 3D con la etiqueta "-room". Para que esto funcione, el "interior" del cuarto debe estar cerrado (la geometría de los hijos debe contener paredes, techo, piso, etc. y los únicos orificios al exterior deben estar cubiertos por portales). El importador entonces creara una versión simplificada de la geometría para el cuarto.

La segunda forma es usando el modificador "-room" en un nodo Mesh. Esto usará la malla como base para el árbol BPS que contiene los límites de los cuartos. Asegúrate que la forma de la malla sea **cerrada**, todas las normales **apunten hacia afuera** y que la geometría no se **intersecte a si misma**, de otra forma los limites pueden ser mal computados (los árboles BSP son demasiado detallistas y difíciles de trabajar, motivo por el cual se usan muy poco en la actualidad..).

De cualquier forma, el cuarto necesitara portales, que se describen a continuación.

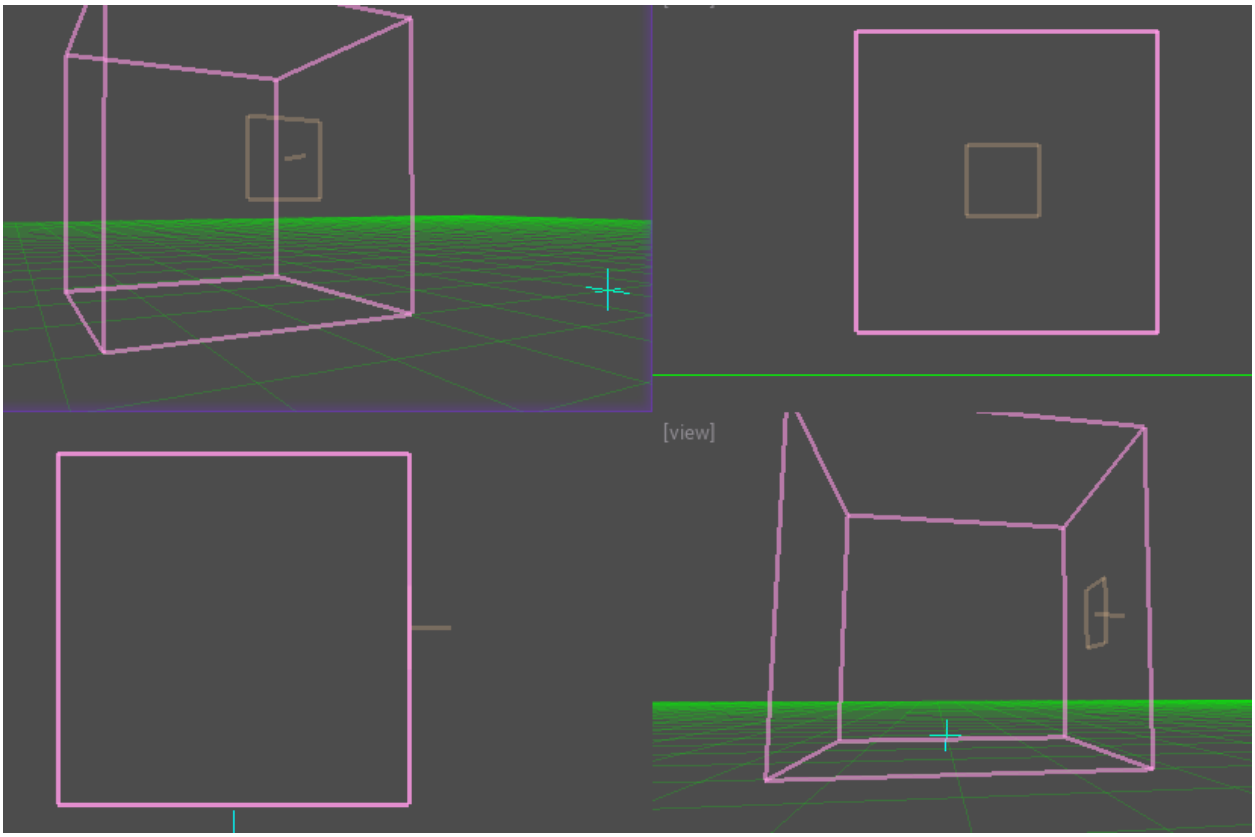
Create portals (-portal)

Los portales son la vista para mirar fuera del cuarto. Siempre son algún tipo de forma plana en la superficie del cuarto. Si el portal es dejado solo, es usado para activar la oclusión cuando miras dentro<->fuera del cuarto.

Básicamente, las condiciones para hacer e importar un portal desde un DCC 3D son:

- Debe ser hijo de un cuarto.
- Debe reposar en la superficie del cuarto (esto no necesita ser súper exacto, solo hazlo lo más cercano posible a ojo y Godot lo ajustara)
- Debe ser plano, con forma convexa, cualquier forma plana y convexa está bien, no importa el eje o tamaño.
- Las normales de la forma plana deben **todas apuntar hacia AFUERA** del cuarto.

Así es como luce usualmente:



Para conectar los cuartos, simplemente haz dos portales idénticos para ambos cuartos y ubícalos superpuestos. Esto no tiene que ser perfectamente exacto, nuevamente, Godot intentara arreglarlo.

[..]

El resto de las etiquetes en este sección deberían ser bastante obvios, o serán documentados/cambiados en el futuro.

Double-sidedness

Collada y cualquier otro formato soporta especificar el doble-lado de la geometría (en otras palabras, cuando no tiene doble lado, las caras hacia atrás no se dibujaran). Godot soporta esta opción por Material, no por Geometría.

Cuando exportas desde el DCC 3D que funciona con doble-lado por objeto (como Blender o Maya), asegúrate que los objetos con doble lado no comparten material con los que tienen un solo lado, o el importador no sabrá discernir.

Animation options

Algunas cosas a tener en cuenta cuando se importan animaciones. Los DCCs 3D permiten animación con curvas para cada componente x,y,z haciendo IK constraints y otras cosas. Cuando se importa para tiempo real, las animaciones son muestreadas (en pequeños intervalos) por lo que toda esta información se pierde. Las animaciones muestreadas son rápidas para procesar, pero pueden requerir cantidades considerables de memoria.

Por este motivo, la opción “Optimize” existe pero, en algunos casos, esta opción puede romper una animación, así que asegúrate de deshabilitarla si notas algún problema.

Algunas animaciones están destinadas a ser cíclicas (como animaciones de caminar) si este es el caso, las animaciones con nombres que terminan en "-cycle" o "-loop" son automáticamente ajustadas para repetirse.

Import script

Crear un script para procesar la escena importada es en los hechos realmente simple. Esto es genial para post processing, cambiar materiales, hacer cosas con la geometría, etc.

Crea un script que básicamente luce así:

```
tool # necesario para que corra en el editor
extends EditorScenePostImport

func post_import(scene):
  # haz tus cosas aquí
  pass # la escena contiene la escena importada comenzando del nodo raíz
```

La función por importación toma la escena importada como un parámetro (el parámetro es en realidad el nodo raíz de la escena).

Update logic

Otros tipos de recursos (como sonidos, mallas, fuentes, imágenes, etc) son re importados por completo cuando cambian y los cambios del usuario no se mantienen.

Debido a que las escenas 3D pueden ser realmente complejas, usan una estrategia de actualización diferente. El usuario puede haber hecho cambios locales para tomar ventaja de las características del motor y sería realmente frustrante si todo se pierde al re importar por que el asset fuente cambio.

Esto llevo a la implementación de una estrategia de actualización. La idea detrás de la misma es que el usuario no pierde nada de lo que hizo, y solo datos agregados o que no pueden ser editados en Godot se actualizarán.

Funciona como esto:

Estrategia

Cuando se realizan cambios en el asset fuente (ejemplo: .dae), y se re importa, el editor recordara como lucia la escena originalmente, y seguirá tus cambios locales como nodos renombrados, moviéndolos o re aparentándolos. Finalmente, lo siguiente será actualizado:

- Los datos de mallas serán reemplazados por los datos de la escena actualizada.
- Los materiales serán mantenidos si no fueron modificados por el usuario.
- Las formas de Portals y Rooms serán remplazadas por las de la escena actualizada.
- Si el usuario movio un nodo dentro de Godot, la transformación se mantendrá. Si el usuario movió un nodo en el asset fuente, la transformación será reemplazada. Finalmente, si el nodo fue movido en ambos lugares, la transformación será combinada.

En general, si el usuario borra cualquier cosa de la escena importada (nodo, malla, material, etc.), actualizar el asset fuente va a restaurar lo que fue borrado. Esta es una buena forma de revertir cambios locales para cualquier cosa. Si realmente no quieres más un nodo en la escena, bórralo de los dos lugares o agrega la etiqueta "-noimp" en el asset fuente.

Fresh re-import

También puede suceder que el asset fuente cambio hasta deja de ser reconocible y se desea una re importación completa. Si es así, solo re abre el dialogo de importación de escena 3D desde el menú Import -> Re-Import y realiza re-import.

CAPÍTULO 5

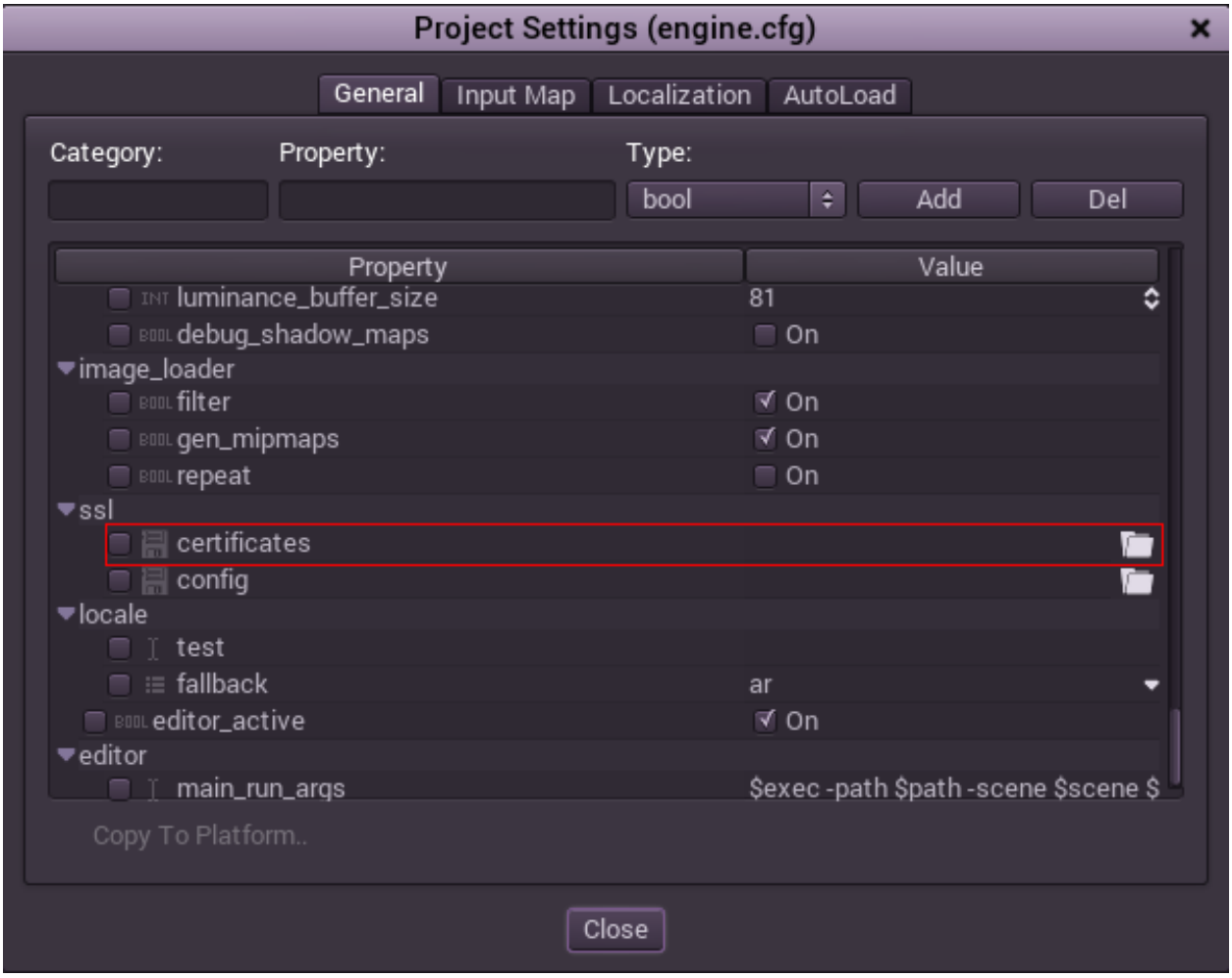
Redes

5.1 Certificados SSL

5.1.1 Introducción

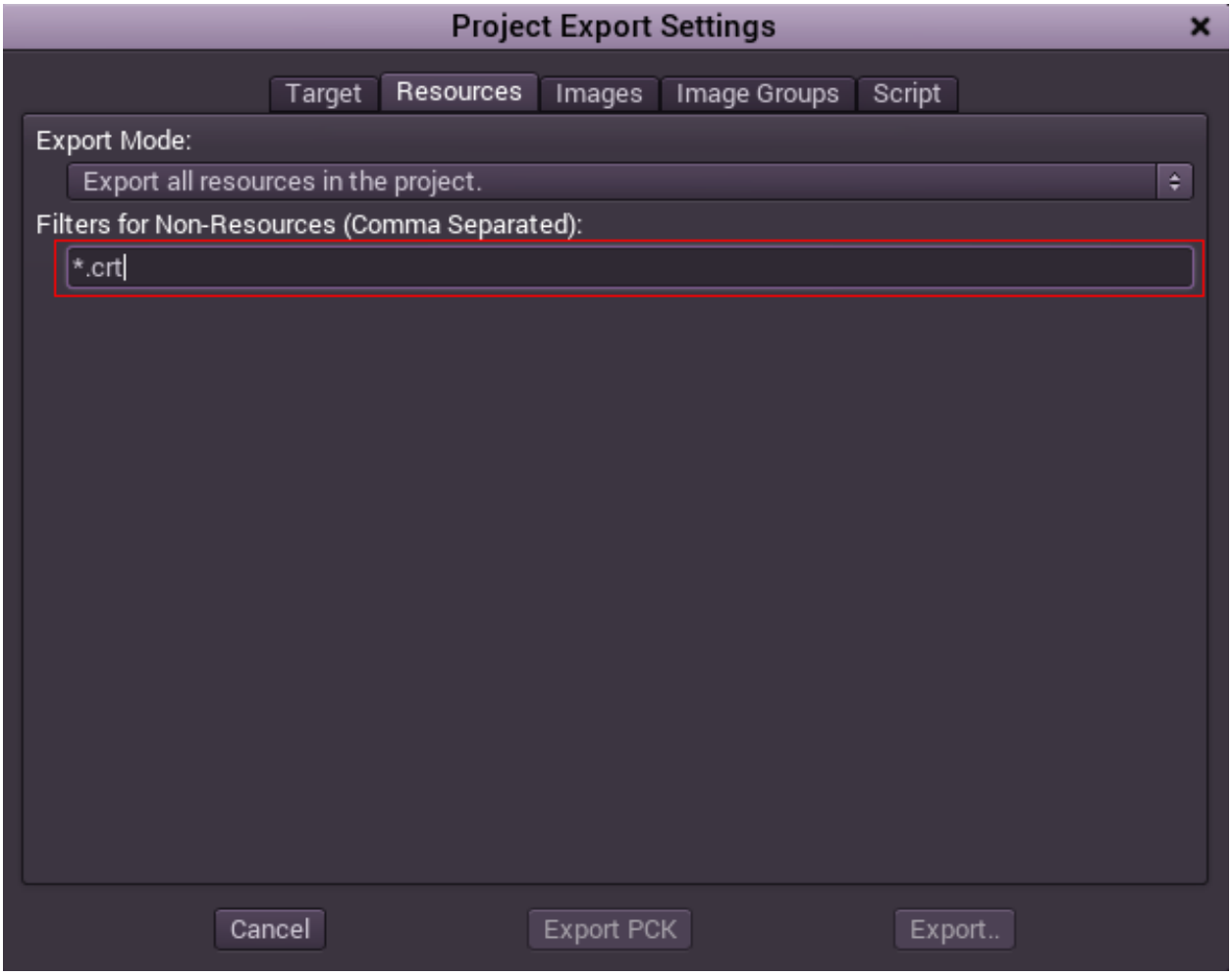
A menudo se quieren usar conexiones SSL para las comunicaciones de forma de evitar ataques "hombre en el medio". Godot tiene un wraper de conexión, *StreamPeerSSL*, el cual puede tomar una conexión regular y agregarle seguridad. La clase *HTTPClient* también soporta HTTPS al usar el mismo wrapper.

Para que funcione SSL, los certificados necesitan ser aprobados. Un archivo .crt debe ser especificado en la configuración de proyecto:



Este archivo debe contener cualquier número de certificados públicos en formato http://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail

Por supuesto, recuerda agregar .crt como filtro así el exportador lo reconoce cuando exporta tu proyecto.



Hay dos formas de obtener certificados:

5.1.2 Opción 1: self signed cert

La primer opción es la más simple, solo genera un par de claves privadas y públicas, y pon el par público en el archivo .crt (nuevamente, en formato PEM). La clave privada solo debería ir en tu servidor.

OpenSSL tiene algo de documentación sobre esto. Esta opción **no requiere validación de dominio** ni gastar una cantidad considerable de dinero comprando certificados de una autoridad de certificados (CA).

5.1.3 Opción 2: CA cert

La segunda opción consiste de usar una autoridad de certificados (CA) como Verisign, Geotrust, etc. Este es un proceso más pesado, pero es más "oficial" y asegura que tu identidad está claramente representada.

A no ser que estés trabajando con grandes compañias o corporaciones, o necesites conectarte al servidor de alguien más (ejemplo: conectarse a un proveedor REST API via HTTPS como Google) este método no es tan útil.

También, cuando usas un certificado emitido por una CA, **deber habilitar la validación de dominio**, para asegurar que el dominio al que te estas conectando es el adecuado, de otra forma cualquier sitio web puede emitir certificados en la misma CA y funcionaria.

Si usas Linux, puedes usar el archivo de certificados suministrado, generalmente ubicado en:

```
/etc/ssl/certs/ca-certificates.crt
```

Este archivo contiene conexiones HTTPS a virtualmente cualquier sitio web (ejemplo: Google, Microsoft, etc.).

O solo elije uno de los certificados mas específicos allí si te quieres conectar a uno en particular.

5.2 Clase HTTP cliente

Aquí un ejemplo usando la clase *HTTPClient*. Es solo un script, por lo que puede correrse ejecutando:

```
c:\godot> godot -s http_test.gd
```

Se conectara y descargara un sitio web.

```
extends SceneTree

# HTTPClient demo
# Esta simple clase puede hacer pedidos HTTP, no bloqueara pero necesita ser polled␣
↪(sondeada)

func _init():

    var err=0
    var http = HTTPClient.new() # Crea el Client

    var err = http.connect("www.php.net",80) # Conexión a host/port
    assert(err==OK) # Asegurarse que la conexión fue exitosa

    # Esperar hasta que resuelve y conecte
    while( http.get_status()==HTTPClient.STATUS_CONNECTING or http.get_
↪status()==HTTPClient.STATUS_RESOLVING):
        http.poll()
        print("Connecting..")
        OS.delay_msec(500)

    assert( http.get_status() == HTTPClient.STATUS_CONNECTED ) # No se pudo conectar

    # Algunos cabezales (headers)

    var headers=[
        "User-Agent: Pirulo/1.0 (Godot)",
        "Accept: */*"
    ]

    err = http.request(HTTPClient.METHOD_GET,"/ChangeLog-5.php",headers) # Pedir una␣
↪página del sitio

    assert( err == OK ) # Asegurarse que todo está OK

    while (http.get_status() == HTTPClient.STATUS_REQUESTING):
        # Mantenerse sondeando hasta que el pedido este en marcha
        http.poll()
        print("Requesting..")
        OS.delay_msec(500)
```

```
    assert( http.get_status() == HTTPClient.STATUS_BODY or http.get_status() ==␣
↪HTTPClient.STATUS_CONNECTED ) # Asegurarse que el pedido termino bien.

    print("response? ",http.has_response()) # Puede que el sitio no tenga una␣
↪respuesta.


    if (http.has_response()):
        # Si hay respuesta..

        var headers = http.get_response_headers_as_dictionary() # Obtener los␣
↪cabezales de respuesta
        print("code: ",http.get_response_code()) # Mostrar código de respuesta
        print("**headers:\\n",headers) # Mostrar cabezales

        # Obteniendo el Body HTTP

        if (http.is_response_chunked()):
            # Usa trozos (chunks)?
            print("La respuesta está en trozos!")
        else:
            # O es solo contenido plano
            var bl = http.get_response_body_length()
            print("Largo de respuesta: ",bl)

        # Este método funciona para ambos de todas formas

        var rb = RawArray() # Arreglo que contendrá los datos

        while(http.get_status()==HTTPClient.STATUS_BODY):
            # Mientras haya "Body" para leer
            http.poll()
            var chunk = http.read_response_body_chunk() # Obtén un trozo
            if (chunk.size()==0):
                # No se obtuvo nada, esperar a que los buffers se llenen un poco
                OS.delay_usec(1000)
            else:
                rb = rb + chunk # Agregar al buffer de lectura


        # Hecho!

        print("bytes got: ",rb.size())
        var text = rb.get_string_from_ascii()
        print("Text: ",text)


    quit()
```

CAPÍTULO 6

Plugins de Editor

Pronto™.

CAPÍTULO 7

Misceláneo

7.1 Matemática

7.1.1 Matemática vectorial

Introducción

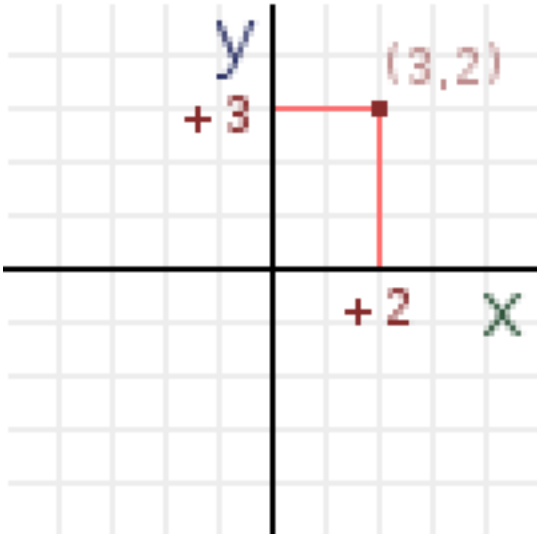
Este pequeño tutorial tiene como objetivo ser una introducción corta y práctica a la matemática de vectores, útil para juegos 3D pero también 2D. De nuevo, la matemática de vectores es útil para juegos 3D pero *también* 2D. Es una herramienta asombrosa cuando la entiendes y hace a la programación de comportamientos complejos mucho más simple.

A menudo sucede que programadores jóvenes confían demasiado en la matemática *incorrecta* para resolver un amplio abanico de problemas, por ejemplo usando solo trigonometría en lugar de matemática de vectores para juegos 2D.

Este tutorial se enfocara en el uso práctico, con aplicación inmediata al arte de la programación de juegos.

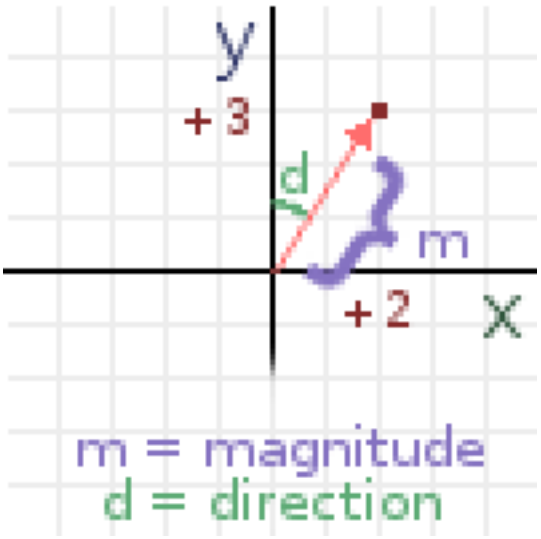
Sistema de coordenadas (2D)

Típicamente, definimos coordenadas como un par (x,y), x representa el valor horizontal e y el vertical. Esto tiene sentido dado que la pantalla es solo un rectángulo en dos dimensiones. Como ejemplo, aquí hay una posición en espacio 2D:



Una posición puede estar en cualquier lugar del espacio. La posición (0,0) tiene un nombre, es llamada el **origen**. Recuerda este término bien porque tiene más usos implícitos luego. El (0,0) de un sistema de n-coordenadas es el **origen**.

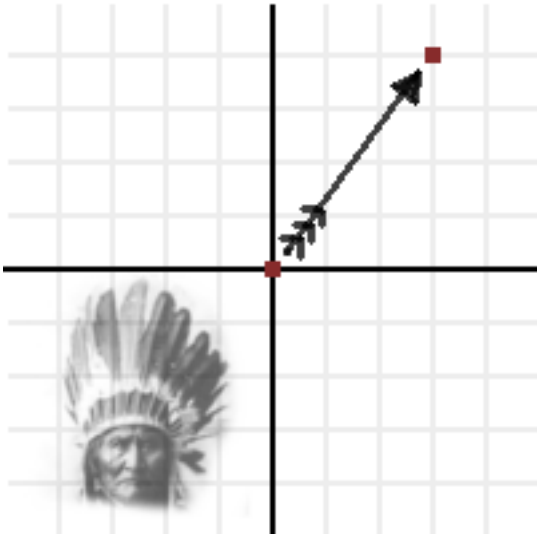
En matemática vectorial, las coordenadas tienen dos usos diferentes, ambos igualmente importantes. Son usadas para representar una *posición* pero también un *vector*. La misma posición que antes, cuando se imagina como un vector, tiene diferente significado.



Cuando se imagina como vector, dos propiedades pueden inferirse, la **dirección** y la **magnitud**. Cada posición en el espacio puede ser un vector, con la excepción del **origen**. Esto es porque las coordenadas (0,0) no pueden representar dirección (magnitud 0).

Dirección

Dirección es simplemente hacia donde el vector apunta. Imagina una flecha que comienza en el **origen** y va hacia la [STRIKEOUT:posición]. La punta de la flecha está en la posición, por lo que siempre apunta hacia afuera, alejándose del origen. Imaginarse vectores como flechas ayuda mucho.



Magnitud

Finalmente, el largo del vector es la distancia desde el origen hasta la posición. Obtener el largo del vector es fácil, solo usa el te teorema Pitagórico <http://en.wikipedia.org/wiki/Pythagorean_theorem>‘__.

```
var len = sqrt( x*x + y*y )
```

Pero... ángulos?

Pero porque no usar un *ángulo*? Después de todo, podríamos pensar de un vector como un ángulo y una magnitud, en lugar de una dirección y una magnitud. Los ángulos también son un concepto más familiar.

Para decir la verdad, los ángulos no son tan útiles en matemática de vectores, y la mayoría del tiempo no se los trata directamente. Tal vez funcionen en 2D, pero en 3D un montón de lo que usualmente puede ser hecho con ángulos no funciona.

Igual, usar ángulos no es una excusa, aun en 2D. La mayoría de lo que toma un montón de trabajo con ángulos en 2D, es de todas formas mas natural y fácil de lograr con matemática de vectores. En matemática vectorial, los ángulos son útiles solo como medida, pero tienen poca participación en la matemática. Entonces, deja de lado la trigonometría de una vez, y prepárate para adoptar vectores!

En cualquier caso, obtener el ángulo desde el vector es fácil y puede ser logrado con trig... er, que fue eso? Digo, la función *atan2()*.

Vectores es Godot

Para hacer los ejemplos más fáciles, vale la pena explicar cómo los vectores son implementados en GDScript. GDscript tiene ambos *Vector2* y *Vector3*, para matemática 2D y 3D respectivamente. Godot usa clases de vectores tanto para posición como dirección. También contienen variables miembro x e y (para 2D) y x, y, z (para 3D).

```
# crea un vector con coordenadas (2,5)
var a = Vector2(2,5)
# crea un vector y asigna x e y manualmente
var b = Vector2()
b.x = 7
b.y = 8
```

Cuando se opera con vectores, no es necesario operar directamente en los miembros (en realidad esto es mucho más lento). Los vectores soportan operaciones aritméticas regulares:

```
# add a and b
var c = a + b
# resultara en un vector c, con valores (9,13)
```

Es lo mismo que hacer:

```
var c = Vector2()
c.x = a.x + b.x
c.y = a.y + b.y
```

Excepto que lo primero es mucho más eficiente y legible.

Las operaciones aritméticas regulares como adición, sustracción, multiplicación y división son soportadas.

La multiplicación y división también puede ser mezclada con números de digito único, también llamados **escalares**.

```
# multiplicación de vector por escalar
var c = a*2.0
# resultara en el vector c, con valor (4,10)
```

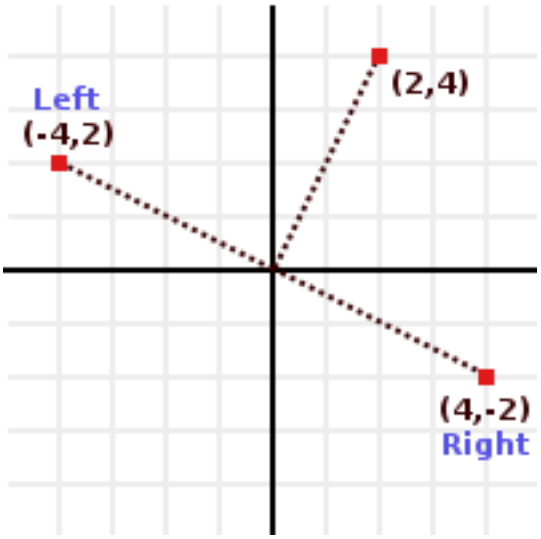
Es lo mismo que hacer

```
var c = Vector2()
c.x = a.x*2.0
c.y = a.y*2.0
```

Excepto que, nuevamente, lo primero es mucho más eficiente y legible.

Vectores perpendiculares

Rotar un vector 2D 90º para algún lado, izquierda o derecha, es realmente fácil, solo intercambia x e y, luego niega x o y (la dirección de la rotación depende en cual es negado).



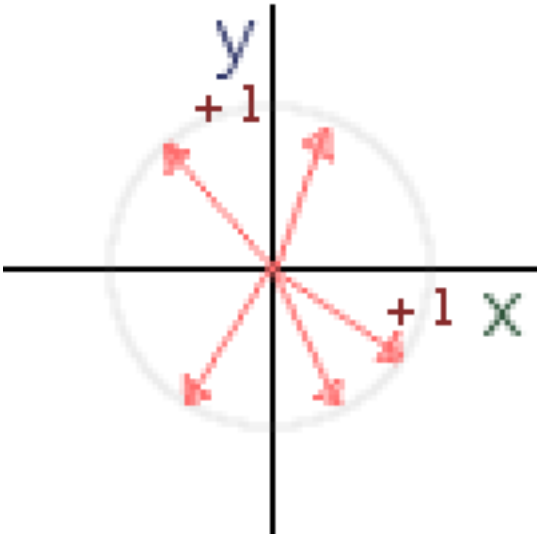
Ejemplo:

```
var v = Vector2(0,1)
# rotar a la derecha (horario)
var v_right = Vector2(-v.y, v.x)
# rotar a la izquierda (antihorario)
var v_right = Vector2(v.y, -v.x)
```

Este es un truco práctico que se usa a menudo. Es imposible de hacer con vectores 3D, porque hay un número infinito de vectores perpendiculares.

Vectores unitarios

OK, entonces sabemos lo que es un vector. Tiene una **dirección** y una **magnitud**. También sabemos cómo usarlos en Godot. El siguiente paso es aprender sobre **vectores unitarios**. Cualquier vector con una magnitud de largo 1 es considerado un **vector unitario**. En 2D, imagina dibujar un circulo de radio uno. Ese círculo contiene todos los vectores unitarios en existencia para dos dimensiones.



Entonces, que es tan especial sobre los vectores unitarios? Los vectores unitarios son asombrosos. En otras palabras, los vectores unitarios tienen **varias propiedades muy útiles**.

No puedes esperar a saber más sobre las fantásticas propiedades de los vectores unitarios, pero un paso a la vez. Así que, como se crea un vector unitario a partir de un vector regular?

Normalización

Tomar cualquier vector y reducir su **magnitud** a 1 mientras mantienes su **dirección** se llama **normalización**. La normalización es hecha al dividir los componentes x e y (y z en 3D) de un vector por su magnitud:

```
var a = Vector2(2,4)
var m = sqrt(a.x*a.x + a.y*a.y)
a.x /= m
a.y /= m
```

Como habrás adivinado, si el vector tiene magnitud 0 (lo que significa que no es un vector pero el **origen** también llamado *vector nulo*), ocurre una división por 0 y el universo atraviesa un segundo big bang, excepto que en polaridad reversa y luego de vuelta. Como resultado, la humanidad está a salvo pero Godot imprimira un error. Recuerda! El vector (0,0) no puede ser normalizado!.

Por supuesto, Vector2 y Vector3 ya tienen un método para esto:

```
a = a.normalized()
```

Producto escalar

OK, el **producto escalar** es la parte más importante de matemática de vectores. Sin el producto escalar, Quake nunca hubiera sido hecho. Esta es la sección más importante del tutorial, así que asegúrate que lo entiendes. La mayoria de la gente que intenta aprender matemática de vectores abandona acá, a pesar de lo simple que es, no le encuetran pie ni cabeza. Porque? Aquí está el porqué, es porque...

El producto escalar toma dos vectores y regresa un **escalar**:

```
var s = a.x*b.x + a.y*b.y
```

Si, básicamente eso. Multiplica **x** del vector **a** por **x** del vector **b**. Haz lo mismo con y, luego súmalo. En 3D es muy parecido:

```
var s = a.x*b.x + a.y*b.y + a.z*b.z
```

Lo se, no tienen ningún sentido! Hasta puedes hacerlo con una función incorporada:

```
var s = a.dot(b)
```

El orden de los dos vectores *no* importa, `a.dot(b)` retorna el mismo valor que `b.dot(b)`.

Aquí es donde empieza la desesperación y los libros y tutoriales te muestran esta fórmula:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta,$$

Y te das cuenta que es tiempo de abandonar el desarrollo de juegos 3D o 2D complejos. Como puede ser que algo tan simple sea tan complejo? Alguien más tendrá que hacer el próximo Zelda o Call of Duty. Los RPGs vistos de arriba no lucen tan mal después de todo. Sip, escuche que alguien lo ha hecho bastante bien con uno de esos en steam...

Así que este es tu momento, tu tiempo de brillar. **NO TE RINDAS**! En este punto, el tutorial tomara un giro rápido y se enfocara en lo que hace útil al producto escalar. Esto es, **porque** es útil. Nos enfocaremos uno por uno en los casos de uso del producto escalar, con aplicación en la vida real. No más fórmulas que no tienen ningún sentido. Las formulas tendrán un montón de sentido *una vez que aprendes* para que son útiles.

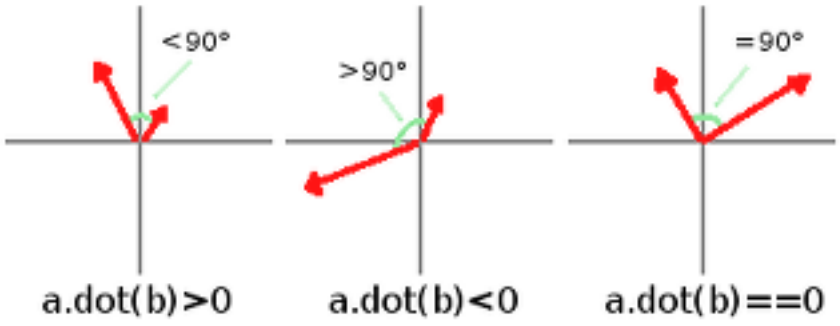
Lado

La primer utilidad y mas importante propiedad del producto escalar es para chequear para que lado miran las cosas. Imaginemos que tenemos dos vectores cualquiera, **a** y **b**. Cualquier **dirección** o **magnitud** (menos el **origen**). No importa lo que sean, solo imaginemos que computamos el producto escalar entre ellos.

```
var s = a.dot(b)
```

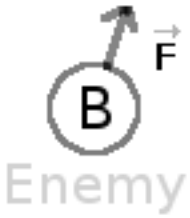
La operación retornara un único número de punto flotante (pero como estamos en el mundo de vectores, lo llamamos **scalar**, seguiremos usando ese término de ahora en más). El número nos dirá lo siguiente:

- Si el número es más grande que cero, ambos están mirando hacia la misma dirección (el ángulo entre ellos es < 90° grados).
- Si el número es menor que cero, ambos están mirando en direcciones opuestas (el ángulo entre ellos es > 90° grados).
- Si el número es cero, los vectores tienen forma de L (el ángulo entre ellos *es* 90° grados).



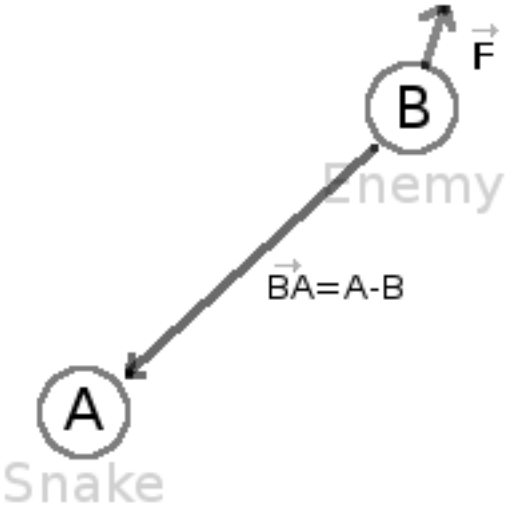
Así que pensemos de un escenario de caso de uso real. Imaginemos que una serpiente está yendo por un bosque, y esta el enemigo cerca. Como podemos decir rápidamente si el enemigo descubrió la serpiente? Para poder descubrirla, el enemigo debe poder *ver* la serpiente. Digamos, entonces que:

- La serpiente está en la posición **A**.
- El enemigo está en la posición **B**.
- El enemigo está *mirando* por el vector de dirección **F**.



Así que, vamos a crear un nuevo vector **BA** que va desde el guardia (**B**) hasta la serpiente (**A**), al restarlos:

```
var BA = A - B
```



Idealmente, si el guardia está mirando hacia la serpiente, para hacer contacto de ojo, necesitaría hacerlo en la misma dirección que el vector BA.

Si el producto escalar entre **F** y **BA** es mayor que 0, entonces la serpiente será descubierta. Esto sucede porque podremos saber si el guardia está mirando hacia ese lado:

```
if (BA.dot(F) > 0):
    print("!")
```

Parece que la serpiente está a salvo por ahora.

Lado con vectores unitarios

Bien, entonces ahora sabemos que el producto escalar entre dos vectores nos dejara saber si miran hacia el mismo lado, lados contrarios o son perpendiculares entre sí.

Esto funciona igual con todos los vectores, no importa la magnitud por lo que los **vectores unitarios** no son la excepción. Sin embargo, usando las mismas propiedades con los vectores unitarios brinda un resultado aun más interesante, ya que se agrega una propiedad extra:

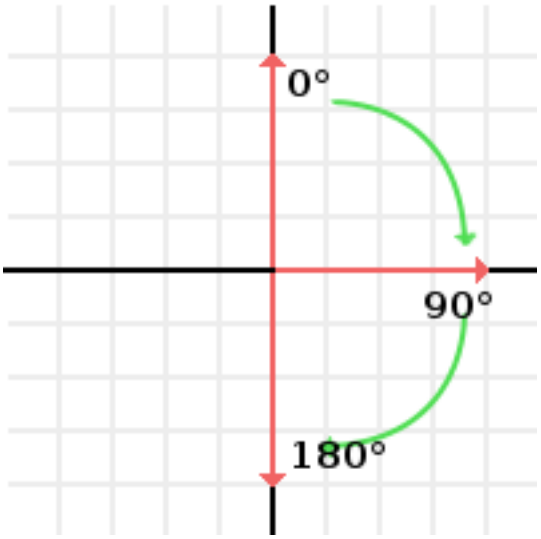
- Si ambos vectores están mirando exactamente hacia la misma dirección (paralelos, ángulo entre ellos de 0°), el escalar resultante es **1**.
- Si ambos vectores están mirando en dirección exactamente opuesta (paralelos, ángulo entre ellos es 180°), el escalar resultante es **-1**.

Esto significa que el producto escalar entre dos vectores unitarios siempre esta en el rango de 1 a -1. Asique de nuevo...

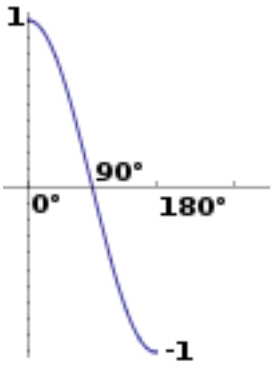
- Si su ángulo es **0**° el producto escalar es **1**.
- Si su ángulo es **90**° el producto escalar es **0**.
- Si su ángulo es **180**° el producto escalar es **-1**.

Mm... esto es extrañamente familiar... lo he visto antes... dónde?

Tomemos dos vectores unitarios. El primero apunta hacia arriba, el segundo también pero lo rotaremos desde arriba (0°) hasta abajo (180°)...



Mientras trazamos el escalar resultante!



Ahhh! Tiene sentido ahora, esto es la función Coseno

Podemos decir que, entonces, como regla...

El **producto escalar** entre dos **vectores unitarios** es el **coseno** del **ángulo** entre esos dos vectores. Entonces, para obtener el ángulo entre los dos vectores, debemos hacer:

```
var angulo_en_radianes = acos( a.dot(b) )
```

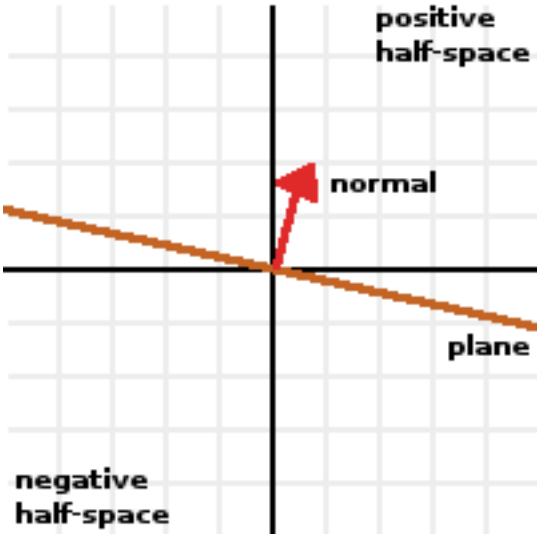
Para que sirve esto? Bueno obtener el ángulo directamente no es tan útil, pero solo poder saber el ángulo es útil como referencia. Un ejemplo es el demo Kinematic Character cuando el personaje se mueve en una cierta dirección y luego golpeamos un objeto. Como saber si lo que se golpeo es el suelo?

Comparando la normal del punto de colisión con un ángulo previamente computado.

La belleza de esto es que el mismo código funciona exactamente igual y sin modificación en 3D. La matemática vectorial es, en gran medida, independiente-de-la-cantidad-de-dimensiones, por lo que agregar o quitar ejes solo agrega muy poca complejidad.

Planos

El producto escalar tiene otra propiedad interesante con los vectores unitarios. Imagina que perpendicular al vector (y a través del origen) pasa un plano. Los planos dividen el espacio entero entre positivo (sobre el plano) y negativo (bajo el plano), y (contrario a la creencia popular) también puedes usar su matemática en 2D:



Los vectores unitarios que son perpendiculares a la superficie (por lo que, ellos describen la orientación de la superficie) son llamados **vectores unitarios normales**. Aunque, usualmente se abrevian solo como *normals. Las normales aparecen en planos, geometría 3D (para determinar hacia donde mira cada cara o vector), etc. Una **normal** *es* un **vector unitario**, pero se llama *normal* por su uso. (De la misma forma que llamamos Origen a (0.0)!).

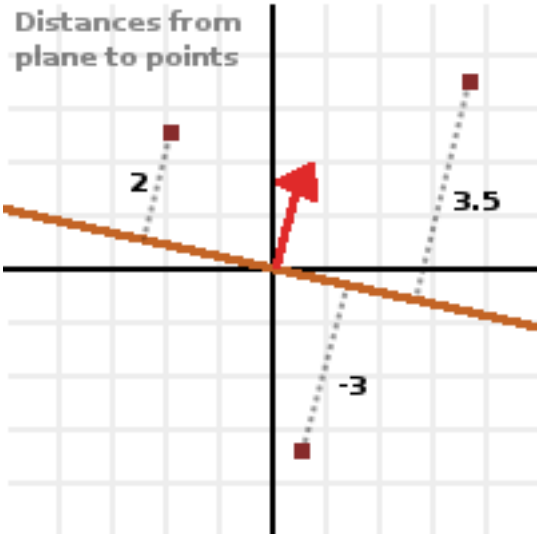
Es tan simple como luce. El plano pasa por el origen y la superficie de el es perpendicular al vector unitario (o *normal*). El lado hacia el cual apunta el vector es el medio-espacio positivo, mientras que el otro lado es el medio-espacio negativo. En 3D esto es exactamente lo mismo, excepto que el plano es una superficie infinita (imagina una pila de papel plana e infinita que puedes orientar y está sujeta al origen) en lugar de una línea.

Distancia a plano

Ahora que está claro lo que es un plano, vamos nuevamente al producto escalar. El producto escalar entre dos **vectores unitarios* y cualquier **punto en el espacio** (si, esta vez hacemos el producto escalar entre vector y posición), regresa la **distancia desde el punto al plano**:

```
var distance = normal.dot(point)
```

Pero no solo la distancia absoluta, si el punto está en en la mitad negativa del espacio la distancia será negativa, también:



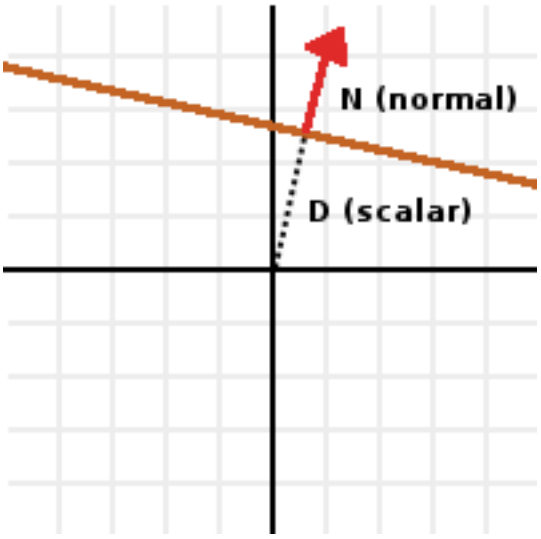
Esto nos permite saber de qué lado de un plano esta un punto.

Fuera del origen

Se lo que estás pensando! Hasta ahora viene bien, pero los planos *verdaderos* están por todos lados en el espacio, no solo pasando por el origen. Quieres acción con *planos* reales y lo quieres *ahora*.

Recuerda que los planos no solo dividen el espacio en dos, pero también tienen *polaridad*. Esto significa que es posible tener planos perfectamente superpuestos, pero sus espacios negativos y positivos están intercambiados.

Con esto en mente, vamos a describir un plano completo como una **normal** *N* y una **distancia desde origen** escalar *D*. Entonces, nuestro plano es representado por N y D. Por ejemplo:



Para matemática 3D, Godot provee un tipo incorporado *Plane* que maneja esto.

Básicamente, N y D pueden representar cualquier plano en el espacio, sea 2D o 3D (dependiendo de la cantidad de las dimensiones de N) y la matemática es la misma para ambos. Es lo mismo que antes, pero D es la distancia desde el origen al plano, viajando en dirección N. Como un ejemplo, imagina que quieres llegar a un punto en el plano, solo harías:

```
var point_in_plane = N*D
```

Esto va a estirar (cambiar el tamaño) el vector normal y lo hará tocar el plano. Esta matemática puede parecer confusa, pero es en realidad mucho más simple de lo que parece. Si queremos saber, nuevamente, la distancia de un punto al plano, hacemos lo mismo pero ajustándolo para la distancia:

```
var distance = N.dot(point) - D
```

Lo mismo, usando una función incorporada:

```
var distance = plane.distance_to(point)
```

Esto, nuevamente, regresara una distancia positiva o negativa.

Dar vuelta la polaridad del plano también es muy simple, solo niega ambos N y D. Esto resultara en un plano en la misma posición, pero invirtiendo las mitades positivas y negativas del espacio:

```
N = -N
D = -D
```

Por supuesto, Godot también implementa el operador en *Plane*, por lo que haciendo:

```
var inverted_plane = -plane
```

Funcionará como se espera.

Así que, recuerda, un plano es solo eso y su uso práctico más importante es calcular la distancia a él. Entonces, por qué es útil calcular la distancia desde un punto al plano? Es extremadamente útil! Vamos a ver algunos ejemplos simples..

Construyendo un plano en 2D

Los planos claramente no vienen de la nada, así que deben ser construidos. Construirlos en 2D es fácil, esto puede ser hecho ya sea de una normal (vector unitario) y un punto, o de dos puntos en el espacio.

En el caso de la normal y el punto, la mayor parte del trabajo está hecho, porque la normal ya está computada, por lo q es solo calcular D desde el producto escalar de la normal y el punto.

```
var N = normal
var D = normal.dot(point)
```

Para dos puntos en el espacio, hay en realidad dos planos que pasan por ellos, compartiendo el mismo espacio pero con la normal apuntando en direcciones opuestas. Para computar la normal desde los dos puntos, el vector de dirección debe ser obtenido en primer lugar, y luego necesita ser rotado 90° para cualquiera de los dos lados:

```
# calcular vector desde a to b
var dvec = (point_b - point_a).normalized()
# rotatr 90 grados
var normal = Vector2(dvec.y,-dev.x)
# o alternativamente
# var normal = Vector2(-dvec.y,dev.x)
# dependiendo del lado deseado de la normal
```

El resto es lo mismo que en el ejemplo previo, tanto point_a o point_b funcionará ya que están en el mismo plano:

```
var N = normal
var D = normal.dot(point_a)
```

```
# esto funciona igual
# var D = normal.dot(point_b)
```

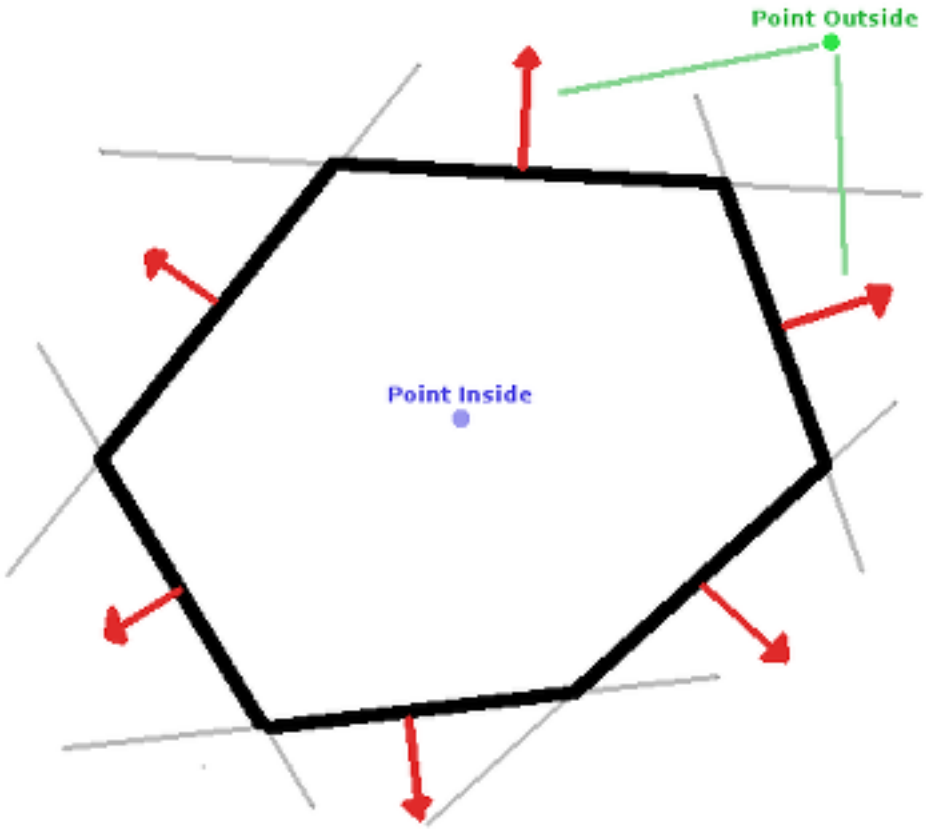
Hacer lo mismo en 3D es un poco más complejo y será explicado mas abajo.

Algunos ejemplos de planos

Aquí hay un ejemplo simple sobre para que son útiles los planos. Imagina que tienes un polígono convexo Por ejemplo, un rectángulo, un trapezoide, un triángulo, o el polígono que sea donde las caras no se doblan hacia adentro.

Para cada segmento del polígono, computamos el plano que pasa por dicho segmento. Una vez tenemos la lista de planos, podemos hacer cosas interesantes, por ejemplo chequear si un punto está dentro de un polígono.

Vamos a través de todos los planos, si podemos encontrar un plano donde la distancia al punto es positiva, entonces el punto está fuera del polígono. Si no podemos, está dentro



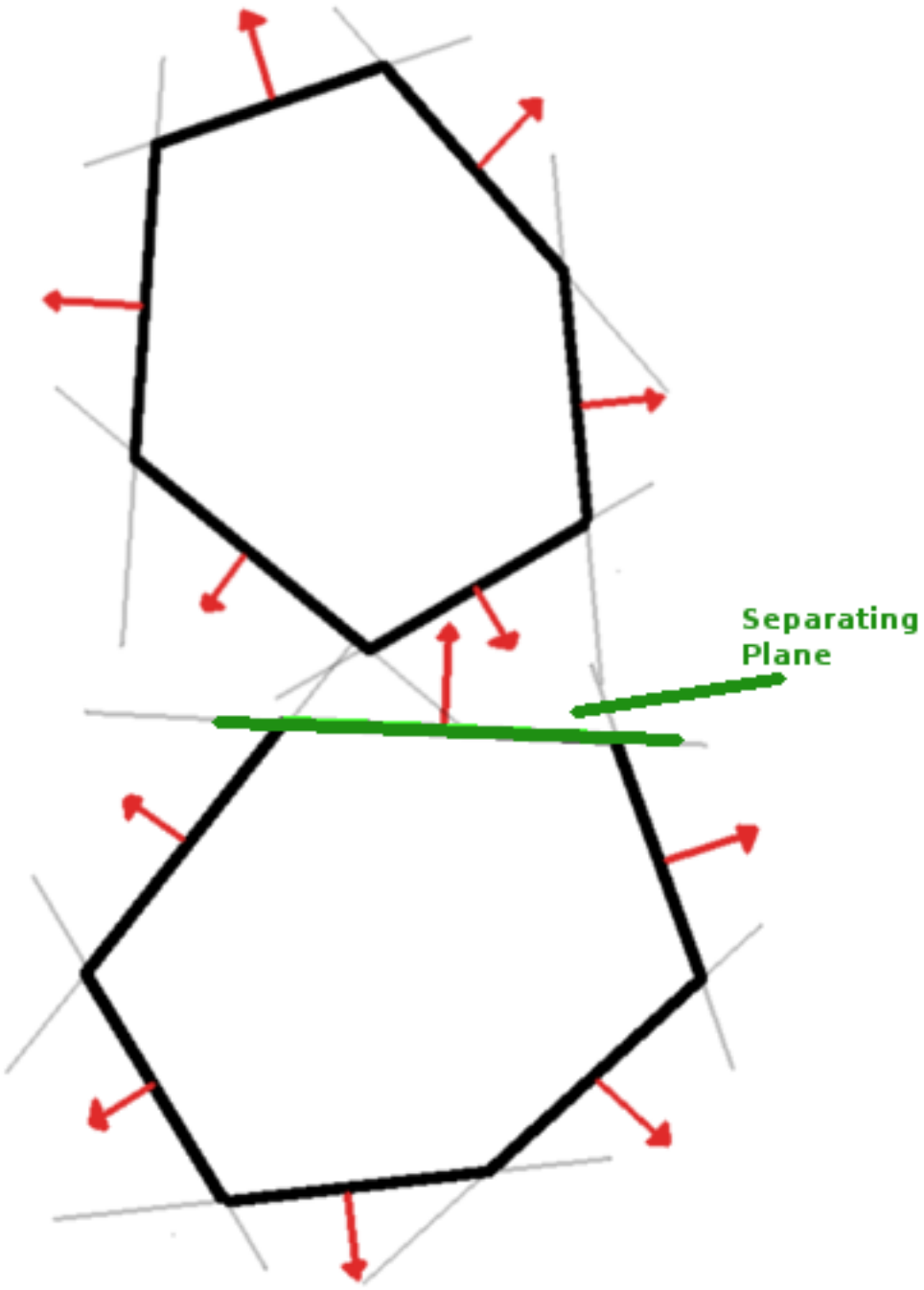
El código sería algo como esto:

```
var inside = true
for p in planes:
    # chequear si la distancia al plano es positiva
    if (N.dot(point) - D > 0):
```

```
        inside = false
        break # si falla uno, es suficiente
```

Bastante copado, eh? Pero se pone mucho mejor! Con un poco más de esfuerzo, lógica similar nos permitirá saber cuándo dos polígonos convexos están superpuestos también. Esto se llama "Teorema de separación de ejes" (Separating Axis Theorem SAT) y la mayoría de los motores de física lo usan para detectar colisiones.

La idea es realmente simple! Con un punto, solo chequear si un plano retorna una distancia positiva es suficiente para saber si el punto esta afuera. Con otro polígono, debemos encontrar un plano donde *todos los **demás** puntos del polígono* retornan una distancia positiva a él. Este chequeo es hecho con los planos A contra los puntos de B, y luego con los planos B contra los puntos de A:



El código debería ser parecido a esto:

```
var overlapping = true

for p in planes_of_A:
    var all_out = true
    for v in points_of_B:
        if (p.distance_to(v) < 0):
            all_out = false
            break
```

```
    if (all_out):
        # se encontró un plano separador
        # no continuar probando
        overlapping = false
        break

if (overlapping):
    # solo haz este chequeo si no se encontraron planos
    # de separación en los planos de A
    for p in planes_of_B:
        var all_out = true
        for v in points_of_A:
            if (p.distance_to(v) < 0):
                all_out = false
                break

        if (all_out):
            overlapping = false
            break

if (overlapping):
    print("Los polígonos colisionaron!")
```

Como puedes ver, los planos son bastante útiles, y esto es la punta del iceberg. Puedes estarte preguntando que sucede con los polígonos no convexos. Esto en general se hace dividiendo el polígono concavo en polígonos convexos mas pequeños, o usando una técnica como BSP (la cual ya no se usa mucho hoy en día).

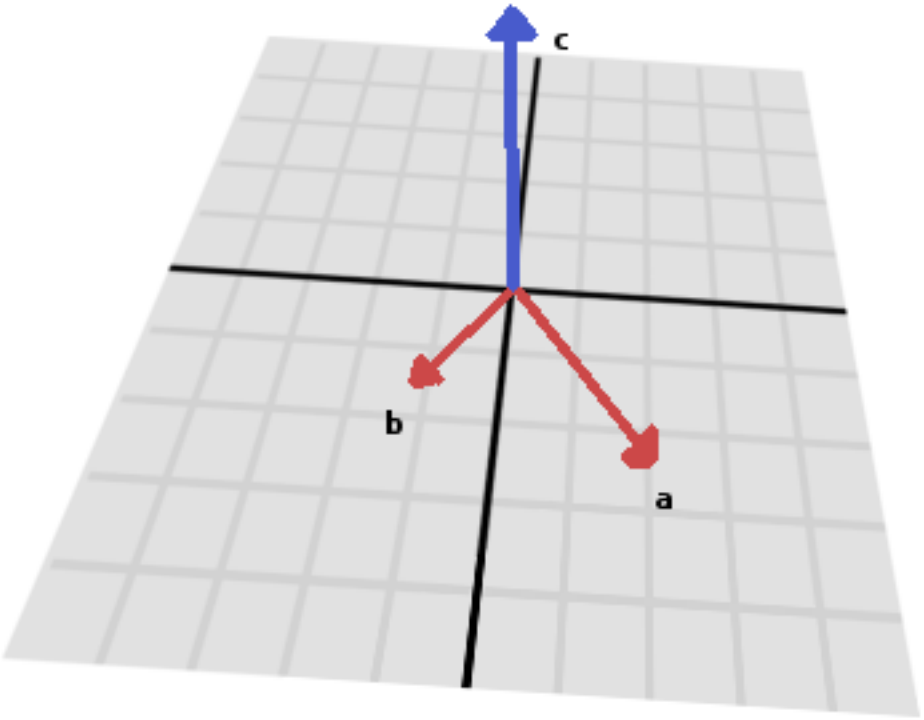
Producto Vectorial

Un montón puede ser hecho con el producto escalar! Pero la fiesta no seria completa sin el producto vectorial. Recuerdas al comienzo de este tutorial? Específicamente como obtener un vector perpendicular (rotado 90 grados) al intercambiar x e y, luego negando uno para rotación derecha (horario) o izquierda (anti-horario)? Eso termino siendo útil para calcular un plano 2D normal desde dos puntos.

Como se mencionó antes, no existe tal cosa en 3D porque un vector 3D tiene infinitos vectores perpendiculares. Tampoco tendría sentido obtener un plano 3D con 2 puntos, ya que en su lugar se necesitan 3 puntos.

Para ayudarnos con este tipo de cosas, las mentes más brillantes de los principales matemáticos nos trajo el **producto vectorial**.

El producto vectorial toma dos vectores y retorna otro vector. El tercer vector retornado siempre es perpendicular a los dos primeros. Los vectores fuente, por supuesto, no deben ser iguales, y no deben ser paralelos u opuestos, de lo contrario el vector resultante será (0,0,0):



La formula para el producto vectorial es:

```
var c = Vector3()
c.x = (a.y + b.z) - (a.z + b.y)
c.y = (a.z + b.x) - (a.x + b.z)
c.z = (a.x + b.y) - (a.y + b.x)
```

Esto puede ser simplificado, en Godot, a:

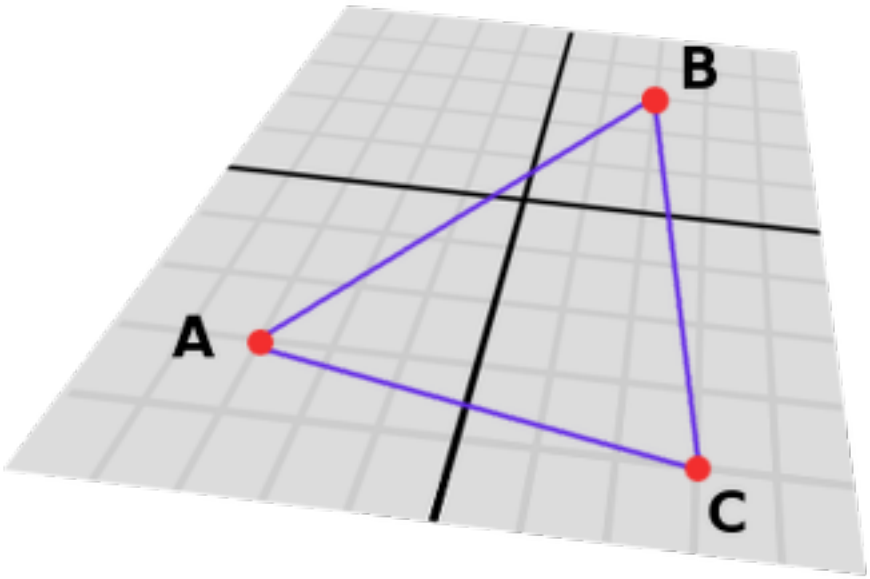
```
var c = a.cross(b)
```

Sin embargo, a diferencia del producto escalar, hacer `a.cross(b)` y `b.cross(a)` producirá resultados diferentes. Específicamente, el vector retornado será negado para el segundo caso. Como te habrás dado cuenta, esto coincide con crear planos perpendiculares en 2D. En 3D, también hay dos posibles vectores perpendiculares a un par de vectores 2D.

Además, el resultado del producto vectorial de dos vectores unitarios *no* es un vector unitario. El resultado deberá ser re normalizado.

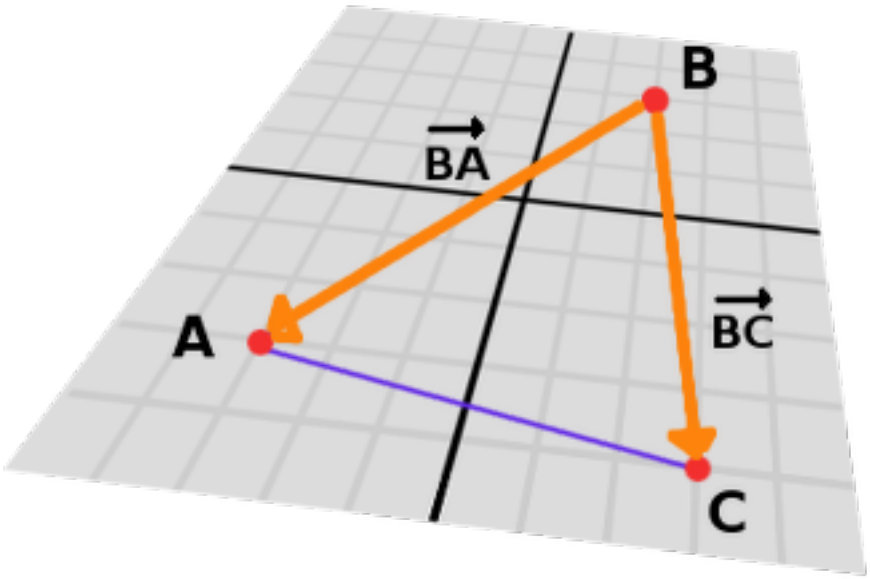
área de un triángulo

El producto vectorial puede ser usado para obtener la superficie de un triángulo en 3D. Dado que un triángulo consiste de 3 puntos, **A**, **B** y **C**:



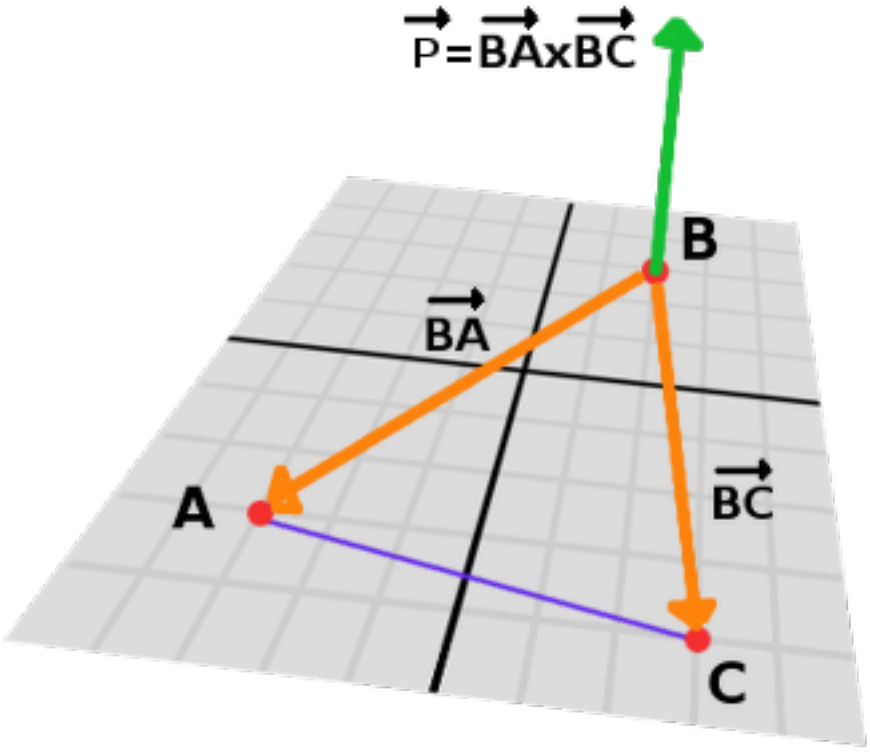
Toma cualquiera de ellos como un pívot y computa los vectores adyacentes a los otros dos puntos. A modo de ejemplo: usaremos B como pívot.

```
var BA = A - B
var BC = C - B
```



Computa el producto vectorial entre **BA** y **BC** para obtener el vector perpendicular **P**:

```
var P = BA.cross(BC)
```



El largo (magnitud) de **P** es la superficie del área del paralelogramo construido por los dos vectores **BA** y **BC**, por lo cual el área de superficie del triángulo es la mitad de él.

```
var area = P.length()/2
```

Plano de un triángulo

Con **P** computado desde el paso previo, normalizalo para obtener la normal del plano.

```
var N = P.normalized()
```

Y obtiene la distancia al hacer el producto escalar de P con cualquiera de los 3 puntos del triángulo **ABC**:

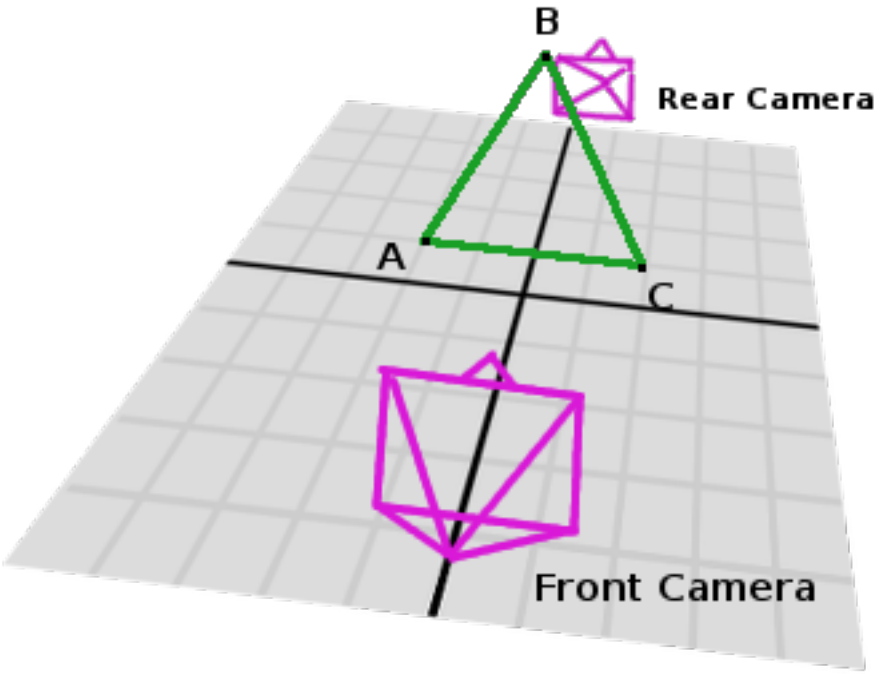
```
var D = P.dot(A)
```

Fantástico! Computaste el plano desde un triángulo!

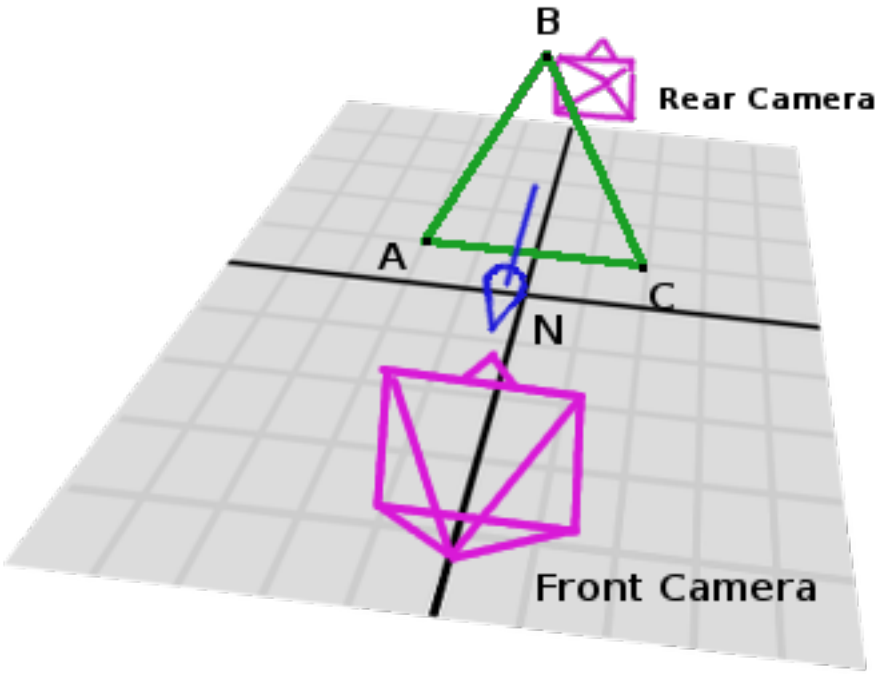
Aquí un poco de información útil (que puedes encontrar en el código fuente de Godot de todas formas). Computar un plano desde un triángulo puede resultar en 2 planos, por lo que alguna convención debe ser ajustada. Esto usualmente depende (en juegos de video y para visualización 3D) en usar el lado del triángulo que mira al frente.

En Godot, los triángulos que miran al frente son aquellos que, cuando cuando están mirando a la cámara, están en orden horario. Los triángulos que miran de forma anti horaria a la cámara no son dibujados (esto ayuda a dibujar menos, así la parte trasera de los objetos no es dibujada).

Para hacerlo un poco más claro, en la imagen de abajo, el triángulo **ABC** aparece de forma horaria cuando se lo mira desde la *Cámara Frontal*, pero a la *Cámara trasera* aparece como anti horario por lo que no será dibujado.



Las normales de los triángulos a menudo están hacia el lado de dirección que se pueden ver, por lo que en este caso, la normal del triángulo ABC apuntaría hacia la cámara frontal:



Así que, para obtener N, la formula correcta es:

```
#  normal horaria de la fórmula del triángulo
var N = (A-C).cross(A-B).normalized()
# para anti horario:
# var N = (A-B).cross(A-C).normalized()
var D = N.dot(A)
```

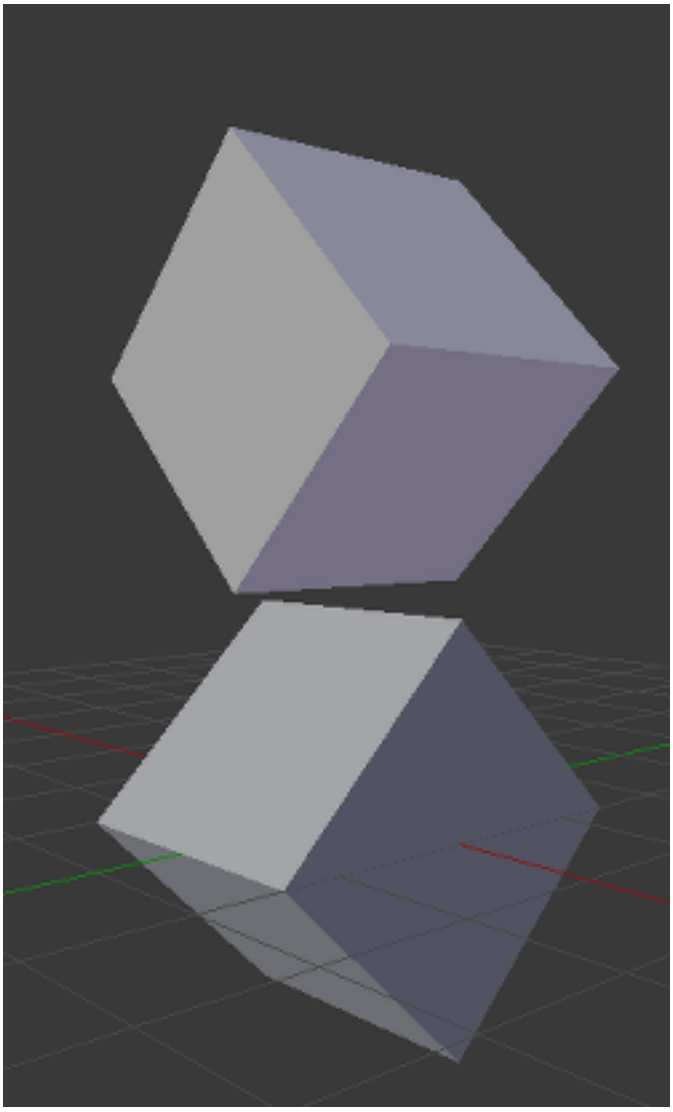
Detección de colisión en 3D

Este es otro pequeño bono, una recompensa por ser paciente y mantenerse en este largo tutorial. Aquí hay otra pieza de sabiduría. Esto puede no ser algo con un caso de uso directo (Godot ya hace la detección de colisión bastante bien) pero es un algoritmo realmente copado para entender de todas formas, porque es usado por casi todos los motores físicos y librerías de detección de colisión :)

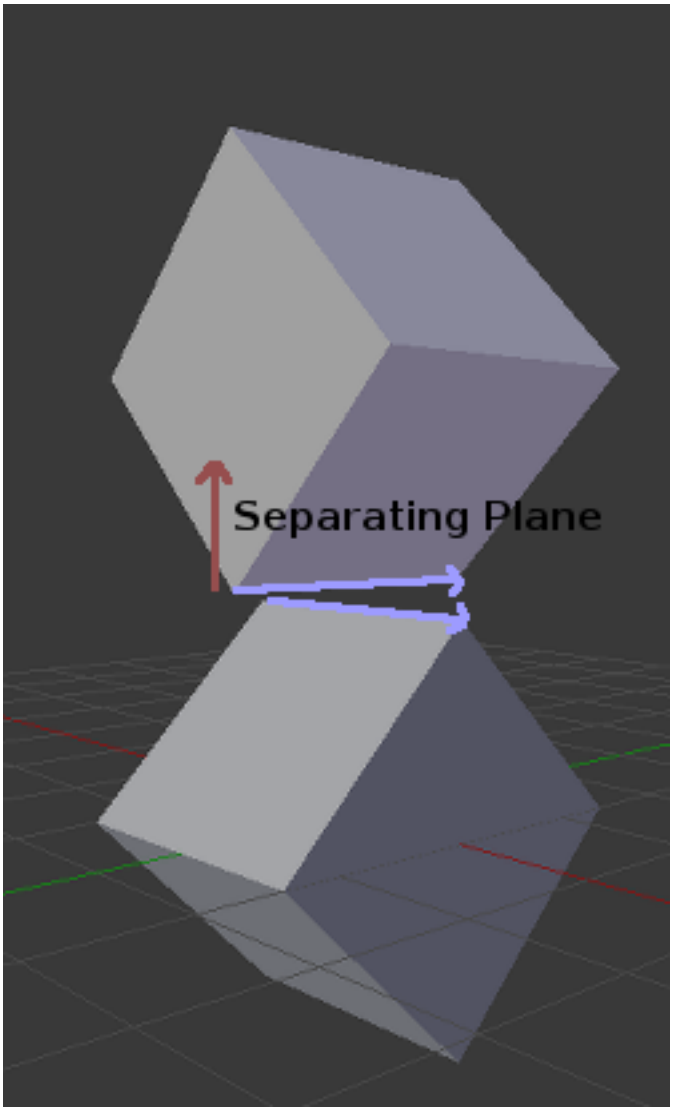
Recuerdas que convertir una figura 2D convexa a un arreglo de planos 2D fue útil para la detección de colisión? Puedes detectar si un punto estaba dentro de una figura convexa, o si dos figuras 2D convexas están superpuestas.

Bueno, esto funciona en 3D también, si dos figuras 3D poliedros están colisionando, no podrás encontrar un plano separador. Si un plano separador se encuentra, entonces las formas definitivamente no están colisionando.

Para refrescar un poco un plano separador significa que todos los vértices del polígono A están en un lado del plano, y todos los vértices del polígono B están en el otro lado. Este plano es siempre uno de los planos de las caras del polígono A o B.



Para evitarlo, algunos planos extra deben ser probados como separadores, estos planos con el producto vectorial entre los lados del polígono A y los lados del polígono B:



Por lo que el algoritmo final es algo así:

```
var overlapping = true

for p in planes_of_A:
    var all_out = true
    for v in points_of_B:
        if (p.distance_to(v) < 0):
            all_out = false
            break

    if (all_out):
        # un plano separador fue encontrado
        # no continuar probando
        overlapping = false
        break

if (overlapping):
    # solo haz este chequeo si no fue encontrado
    # un plano separador en los planos de A
```

```
    for p in planes_of_B:
        var all_out = true
        for v in points_of_A:
            if (p.distance_to(v) < 0):
                all_out = false
                break

        if (all_out):
            overlapping = false
            break

if (overlapping):
    for ea in edges_of_A:
        for eb in edges_of_B:
            var n = ea.cross(eb)
            if (n.length() == 0):
                continue

            var max_A = -1e20 # numero diminuto
            var min_A = 1e20 # numero enorme

            # estamos usando el producto escalar directamente
            # por lo que podemos mapear un rango máximo y mínimo
            # para cada polígono, luego chequear si se superponen

            for v in points_of_A:
                var d = n.dot(v)
                if (d > max_A):
                    max_A = d
                if (d < min_A):
                    min_A = d

            var max_B = -1e20 # numero diminuto
            var min_B = 1e20 # numero enorme

            for v in points_of_B:
                var d = n.dot(v)
                if (d > max_B):
                    max_B = d
                if (d < min_B):
                    min_B = d

            if (min_A > max_B or min_B > max_A):
                # no se superponen!
                overlapping = false
                break

        if (not overlapping):
            break

if (overlapping):
   print("Los polígonos colisionaron!")
```

Esto fue todo! Espero que haya sido de ayuda, y por favor danos tu feedback y déjanos saber si algo en este tutorial no es claro! Deberías estar pronto para el siguiente desafío... *Matrices y transformaciones*!

7.1.2 Matrices y transformaciones

Introducción

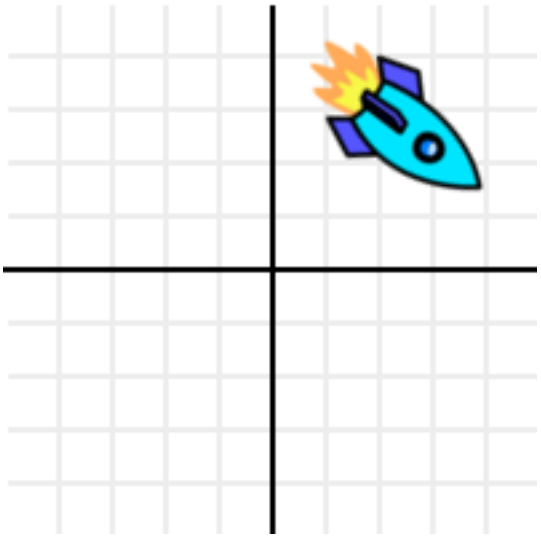
Antes de leer este tutorial, es recomendable leer el anterior sobre *Matemática vectorial* ya que este es una continuación directa.

Este tutorial será sobre *transformaciones* y cubrirá algo sobre matrices (pero no en profundidad).

Las transformaciones son aplicadas la mayor parte del tiempo como traslación, rotación y escala por lo que serán consideradas como una prioridad aquí.

Sistema orientado de coordenadas (OCS)

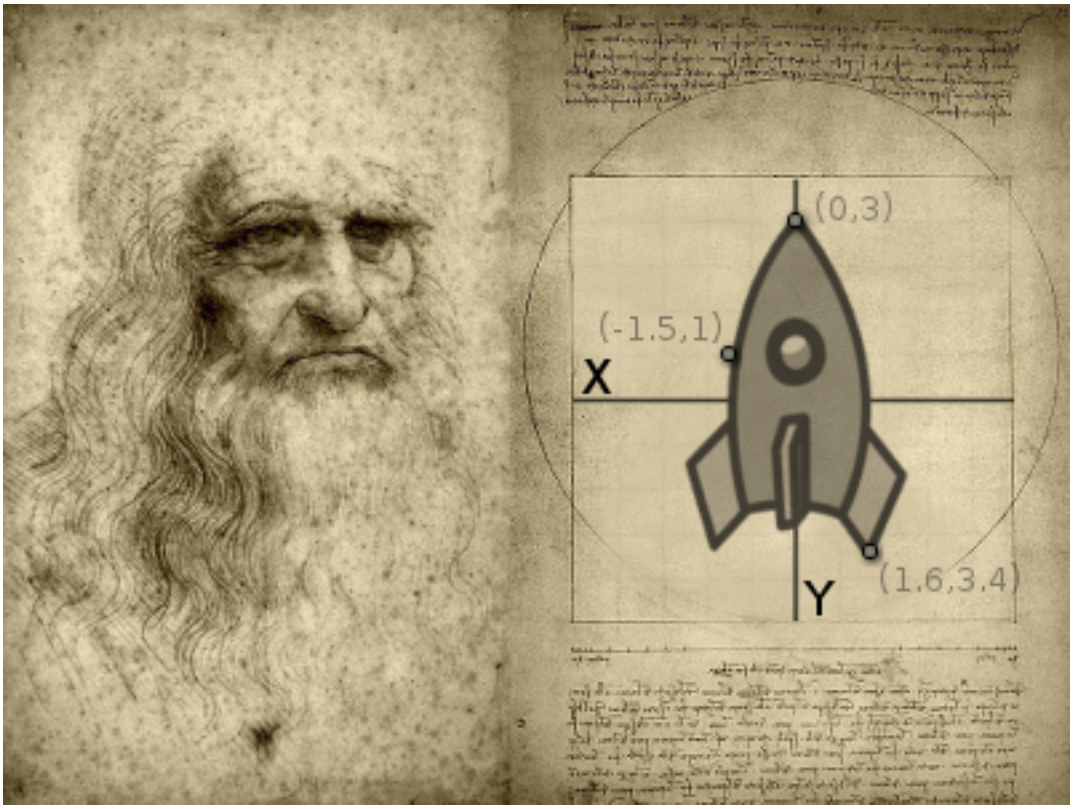
Imagina que tenemos una nave en algún lugar del espacio. En Godot esto es fácil, solo mueve la nave hacia algún lado y rótala:



Bien, entonces en 2D esto luce simple, una posición y un ángulo para una rotación. Pero recuerda, somos adultos aquí y no usamos ángulos (además, los ángulos ni siquiera son tan útiles cuando trabajamos en 3D).

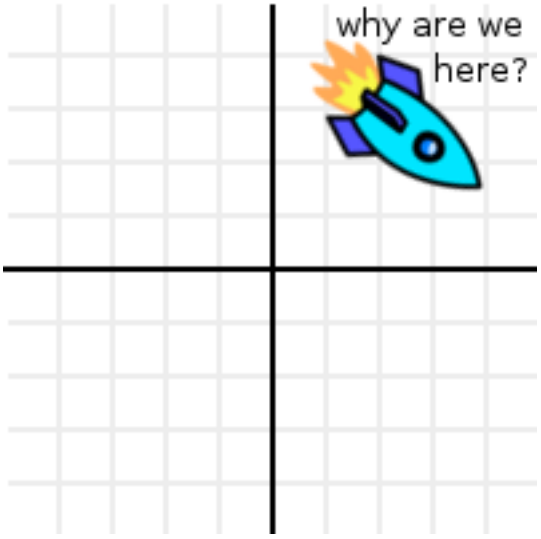
Debemos darnos cuenta que en algún punto, alguien *diseño* esta nave. Sea un dibujo 2D hecho con Paint.net, Gimp, Photoshop, etc. o en 3D a través de herramientas DCC 3D como Blender, Max, Maya, etc.

Cuando fue diseñada, no estaba rotada. Fue diseñada en su propio *sistema de coordenadas*.



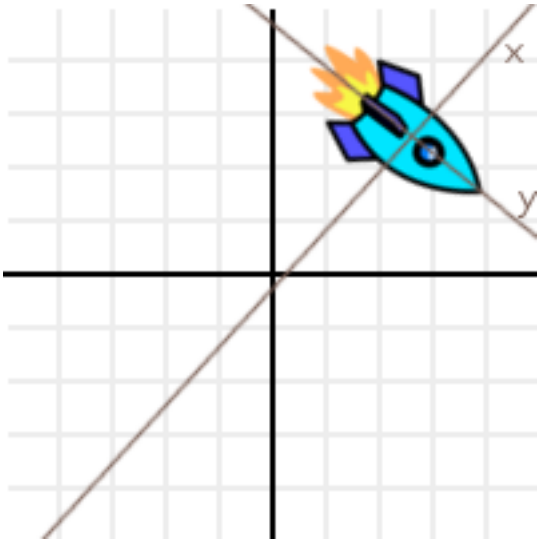
Esto significa que el extremo de la nave tiene una coordenada, el alerón tiene otra, etc. Sea en pixels (2D) o vértices (3D).

Así que, recordemos nuevamente que la nave esta en algún lugar del espacio:

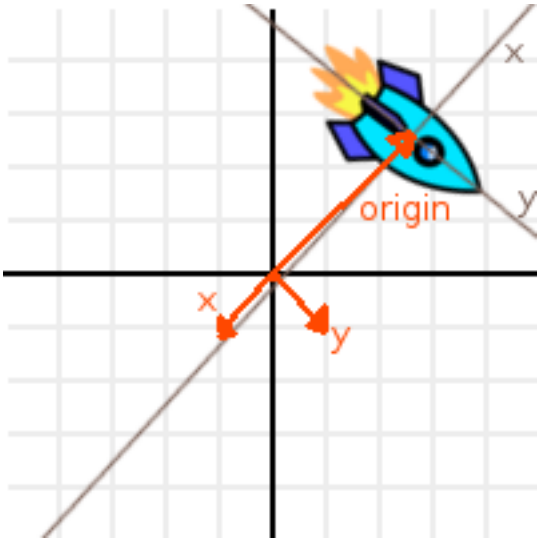


Cómo llego allí? Que la movió y roto desde el lugar que fue diseñada a su posición actual? La respuesta es... un **transform**, la nave fue transformada desde su posición original a la nueva. Esto permite que la nave sea mostrada donde está.

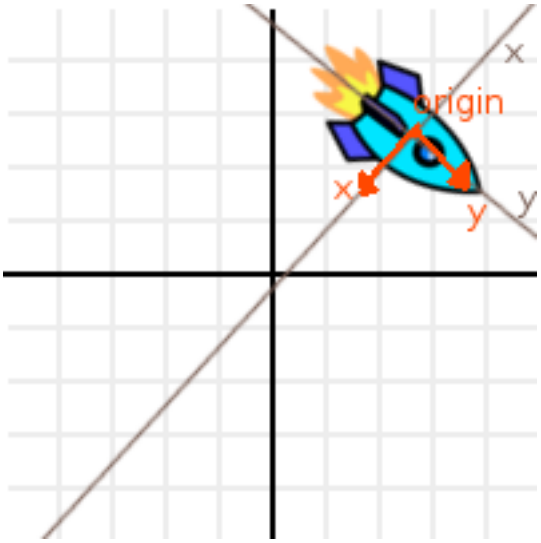
Pero transform es un término muy genérico para describir este proceso. Para resolver este puzzle, vamos a superponer la posición del diseño original a su posición actual:



Entonces, podemos ver que el "espacio de diseño" ha sido transformado también. Como podemos representar mejor esta trasformación? Vamos a usar 3 vectores para esto (en 2D), un vector unitario apuntando hacia X positivo, un vector unitario apuntando hacia Y positivo y una traslación.

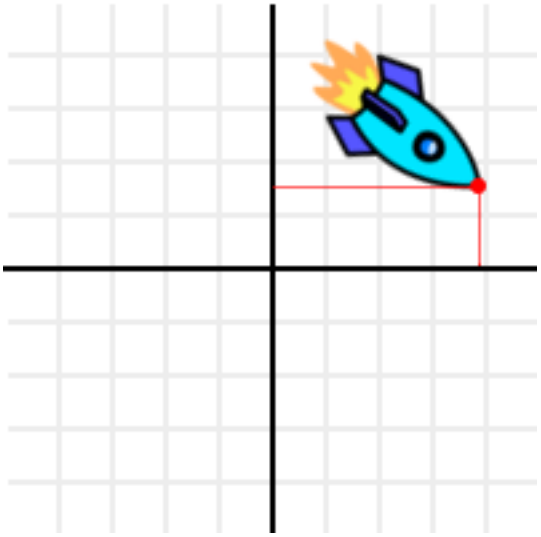


Llamemos a estos 3 vectores "X", "Y" y "Origen", y vamos a superponerlos sobre la nave para que tenga más sentido:



Bien, esto es más lindo, pero aún no tiene sentido. Que tienen que ver X, Y y Origen con como la nave llego allí?

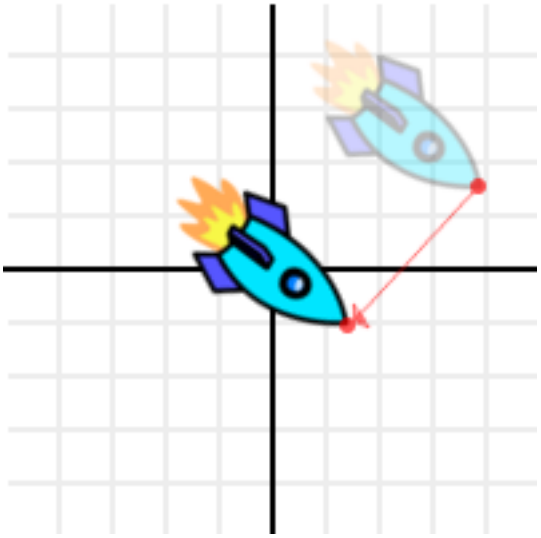
Bueno, tomemos el punto desde el extremo superior de la nave como referencia:



Y le aplicamos la siguiente operación (y a todos los puntos en la nave también, pero vamos a seguir el extremo superior como punto de referencia):

```
var new_pos = pos - origin
```

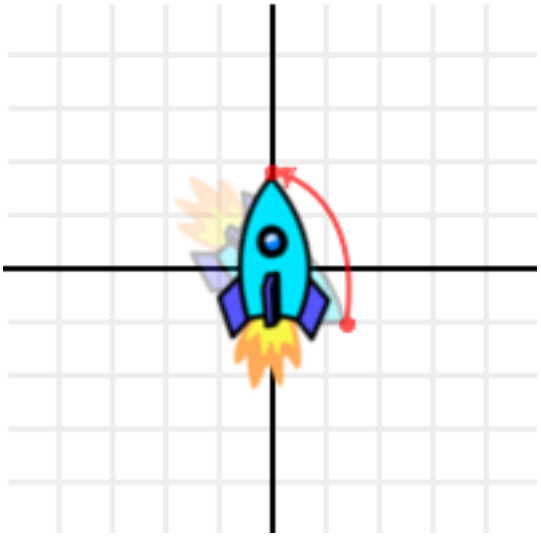
Haciendo esto al punto seleccionado lo moverá de nuevo al centro:



Esto esa esperable, pero luego hagamos algo mas interesante. Usando el producto escalar de X y el punto, y agrégalo al producto escalar de Y y el punto:

```
var final_pos = x.dot(new_pos) + y.dot(new_pos)
```

Entonces lo que tenemos es.. espera un minuto, es la nave en su posición de diseño!



Como sucedió esta magia negra? La nave estaba perdida en el espacio, y ahora esta de nuevo en casa!

Puede parecer extraño, pero tiene mucha lógica. Recuerda, como vimos en *Matemática vectorial*, lo que sucedió es que la distancia al eje X, y la distancia al eje Y fue computada. Calcular distancia en una dirección o plano era uno de los usos para el producto escalar. Esto fue suficiente para obtener de vuelta las coordenadas de diseño para cada punto en la nave.

Entonces, con lo que ha estado trabajando hasta ahora (con X, Y y Origen) es un *Sistema ordenado de coordenadas*. *X e Y son la **Base*, y *Origen* es el offset (compensación).

Base

Sabemos lo que es el Origen. Es donde termino el 0,0 (origen) del sistema de coordenadas luego de ser transformado a una nueva posición. Por esto es llamado *Origen*, pero en la práctica, es solo un offset hacia la nueva posición.

La base es más interesante. La base es la dirección de X e Y en el OCS (sistema de coordenadas) de la nueva posición transformada. Nos dice que ha cambiado, ya sea en 2D o 3D. El Origen (offset) y la Base (dirección) comunican "Oye, el X e Y original de tu diseño están *aquí*, apuntando hacia *estas direcciones*."

Entonces, cambiemos la representación de la base. En lugar de 2 vectores, usemos una *matriz*.

$$M = \begin{Bmatrix} X_x & X_y \\ Y_x & Y_y \end{Bmatrix}$$

Los vectores están allí en la matriz, horizontalmente. El siguiente problema ahora es que.. es esto de una matriz? Bueno, vamos a asumir que nunca escuchaste de una matriz.

Transforms en Godot

Este tutorial no explicara matemática de matrices (y sus operaciones) en profundidad, solo su uso práctico. Hay mucho material sobre eso, el cual debería ser mucho más simple de entender luego de completar este tutorial. Vamos a explicar solo como usar los transforms.

Matrix32

Matrix32 es una matriz 2x3. Tiene 3 elementos Vector2 y es usada para 2D. El eje "X" es el elemento 0, el eje "Y" es el elemento 1 y "Origen" es el elemento 2. No está dividido en base/origen por conveniencia, debido a su simplicidad.

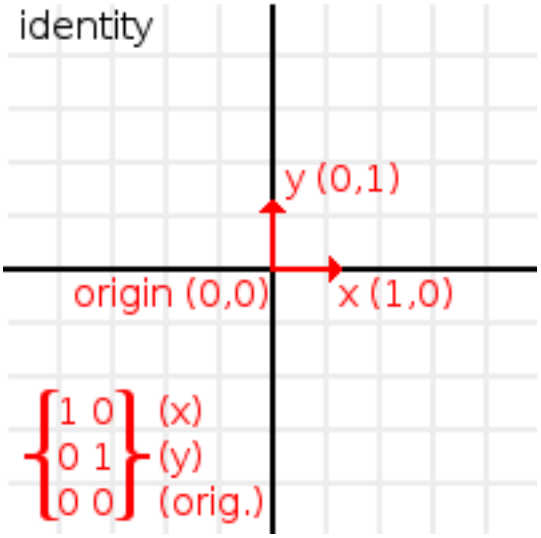
```
var m = Matrix32()
var x = m[0] # 'X'
var y = m[1] # 'Y'
var o = m[2] # 'Origin'
```

La mayoría de las operaciones serán explicadas con este tipo de datos (Matrix32), pero la misma lógica aplica a 3D.

Identidad

Por defecto, Matrix32 es creada como una matriz de "identidad". Esto significa:

- 'X' Apunta a la derecha: Vector2(1,0)
- 'Y' Apunta arriba (o abajo en pixels): Vector2(0,1)
- 'Origen' es el origen Vector2(0,0)



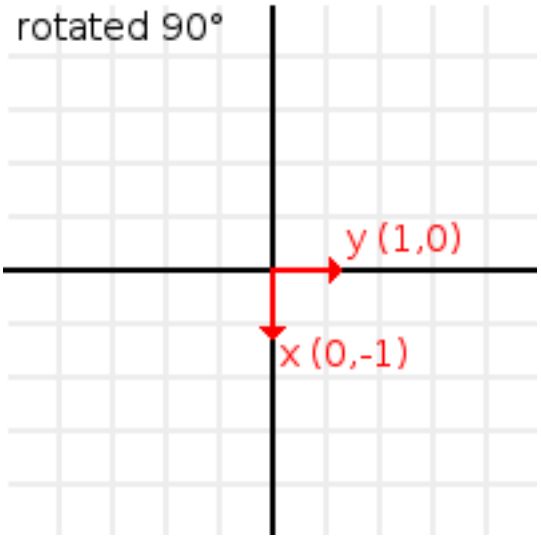
Es fácil adivinar que una matriz *identidad* es solo una matriz que alinea el transform a su sistema de coordenadas padre. Es un *OCS* que no ha sido trasladado, rotado o escalado. Todos los tipos de transforms en Godot son creados con *identidad*.

Operaciones

Rotación

Rotar Matrix32 es hecho usando la función "rotated":

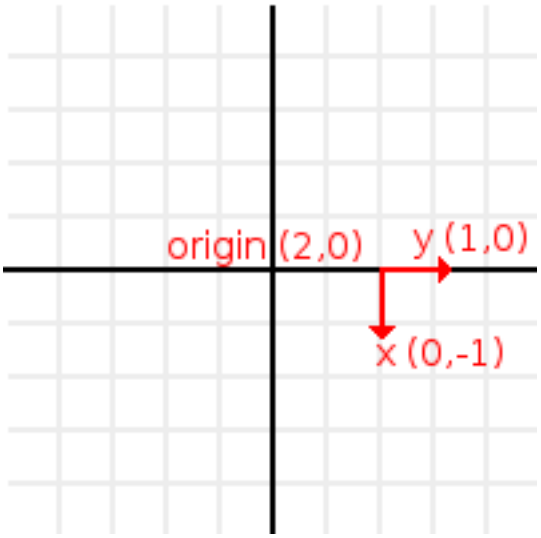
```
var m = Matrix32()
m = m.rotated(PI/2) # rotar 90°
```



Traslación

Hay dos formas de trasladar una Matrix32, la primera es solo mover el origen:

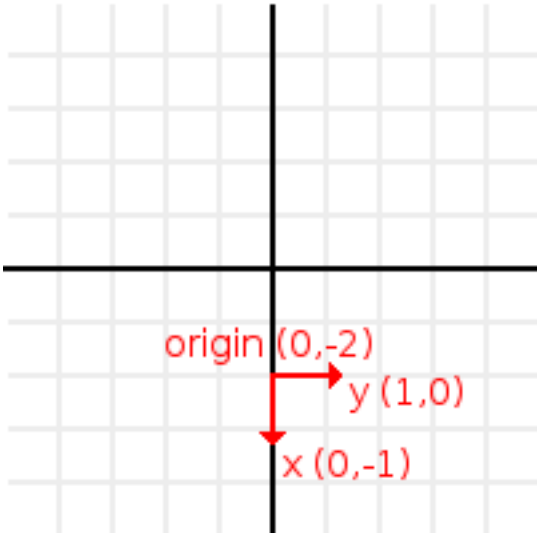
```
# Mover 2 unidades a la derecha
var m = Matrix32()
m = m.rotated(PI/2) # rotar 90°
m[2]+=Vector2(2,0)
```



Esto siempre funcionara en coordenadas globales.

Si en su lugar, la traslación es deseada en coordenadas *locales* de la matriz (hacia donde se orienta la *base*), está el método *Matrix32.translated()* :

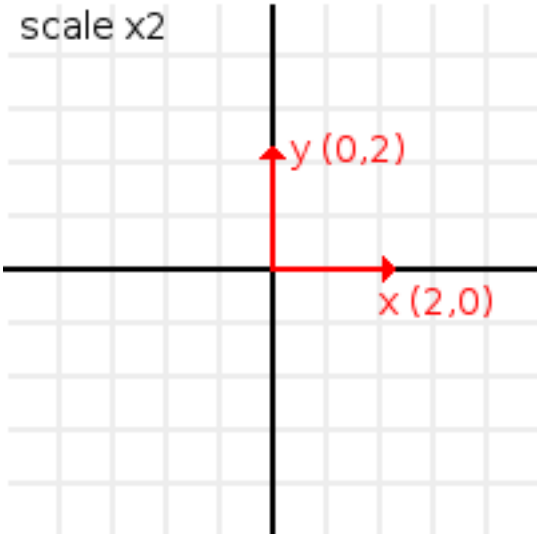
```
# Mover 2 unidades hacia donde está orientada la base
var m = Matrix32()
m = m.rotated(PI/2) # rotar 90°
m=m.translated( Vector2(2,0) )
```



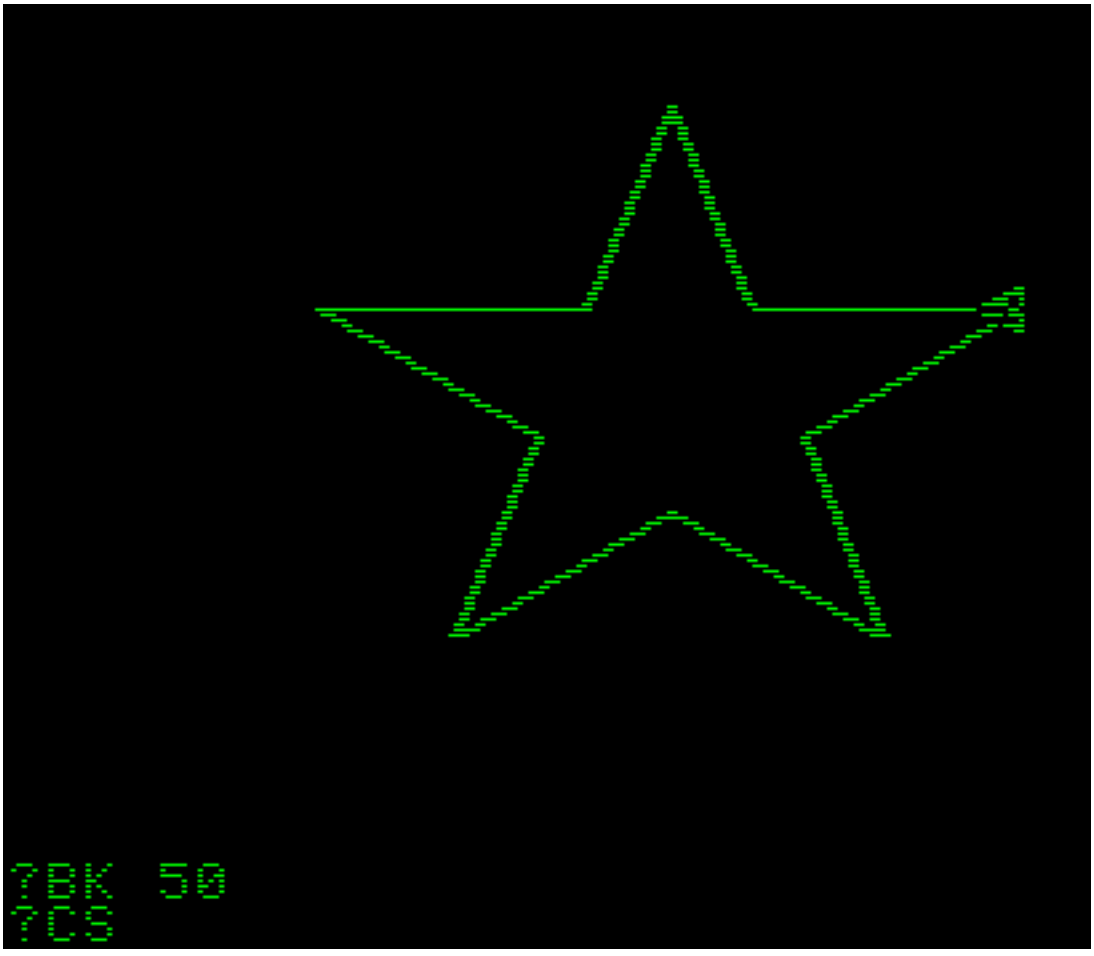
Escala

Una matriz puede ser escalada también. Escalar multiplicara los vectores base por un vector (vector X por componente x de la escala, vector Y por el componente y de la escala). Dejará igual el origen:

```
# Llevar al doble el tamaño de la base.
var m = Matrix32()
m = m.scaled( Vector2(2,2) )
```



Este tipo de operaciones en matrices es acumulativo. Significa que cada una empieza relativa a la anterior. Para aquellos que han estado viviendo en el planeta lo suficiente, una buena referencia de cómo funciona transform es esta:



Una matriz es usada en forma similar a una tortuga. La tortuga muy probablemente tenía una matriz en su interior (y estas descubriendo esto muchos años *después* de descubrir que Santa no es real).

Transform

Transform es el acto de conmutar entre sistemas de coordenadas. Para convertir una posición (sea 2D o 3D) desde el sistema de coordenadas de "diseño" al OCS, el método "xform" es usado:

```
var new_pos = m.xform(pos)
```

Y solo para la base (sin traslación):

```
var new_pos = m.basis_xform(pos)
```

Ademas - multiplicar también es válido:

```
var new_pos = m * pos
```

Transform inversa

Para hacer la operación opuesta (lo que hicimos arriba con el cohete), se usa el método "xform_inv":

```
var new_pos = m.xform_inv(pos)
```

Solo para la base:

```
var new_pos = m.basis_xform_inv(pos)
```

O pre-multiplicación:

```
var new_pos = pos * m
```

Matrices ortonormales

Sin embargo, si la Matrix ha sido escalada (los vectores no tienen largo de unidad), o los vectores base no son ortogonales (90°), el transform inverso no funcionara.

En otras palabras, el transform inverso solo es válido en matrices *ortonormales*. Por ello, en estos casos se debe computar un inverso afín.

El transform, o el transform inverso de una matriz de identidad retornara la posición sin cambio:

```
# No hace nada, pos no cambia
pos = Matrix32().xform(pos)
```

Inverso afín

El inverso afín es la matriz que hace la operación inversa de otra matriz, no importa si la matriz tiene escala o los ejes de vectores no son ortogonales. El inverso afín es calculado con el método affine_inverse():

```
var mi = m.affine_inverse()
var pos = m.xform(pos)
pos = mi.xform(pos)
# pos no cambia
```

Si la matriz es ortonormal, entonces:

```
# si m es ortonormal, entonces
pos = mi.xform(pos)
# es lo mismo que
pos = m.xform_inv(pos)
```

Multiplicación de matrices

Las matrices pueden ser multiplicadas. La multiplicación de dos matrices "encadena" (concatena) sus transforms.

Sin embargo, por convención, la multiplicación toma lugar en orden reverso.

Ejemplo:

```
var m = more_transforms * some_transforms
```

Para hacerlo un poco más claro, esto:

```
pos = transform1.xform(pos)
pos = transform2.xform(pos)
```

Es lo mismo que:

```
# nota el orden inverso
pos = (transform2 * transform1).xform(pos)
```

Sin embargo, esto no es lo mismo:

```
# devuelve resultados diferentes
pos = (transform1 * transform2).xform(pos)
```

Porque en matemática de matrices, A + B no es lo mismo que B + A.

Multiplicación por inverso

Multiplicar una matriz por su inverso, resulta en identidad

```
# No importa lo que A sea, B será identidad
B = A.affine_inverse() * A
```

Multiplicación por identidad

Multiplicar una matriz por identidad, resultara en una matriz sin cambios:

```
# B sera igual que A
B = A * Matrix32()
```

Consejos de Matrices

Cuando usamos una jerarquía de transform, recuerda que la multiplicación de matrices es reversa! Para obtener el transform global para una jerarquía, haz:

```
var global_xform = parent_matrix * child_matrix
```

Para 3 niveles:

```
# debido al orden reverso, se necesitan paréntesis
var global_xform = gradparent_matrix + (parent_matrix + child_matrix)
```

Para hacer una matriz relativa al padre, usa el inverso afín (o el inverso regular para matrices ortonormales).

```
# transformar B desde una matriz global a una local a A
var B_local_to_A = A.affine_inverse() * B
```

Revertirlo es justo como el ejemplo de arriba:

```
# transformar de vuelta B local a B global
var B = A * B_local_to_A
```

Bien, esto debería ser suficiente! Completemos el tutorial moviéndonos a matrices 3D.

Matrices & transforms en 3D

Como mencionamos antes, para 3D, nos manejamos con 3 vectores *Vector3* para la matriz de rotación, y uno extra para el origen.

Matrix3

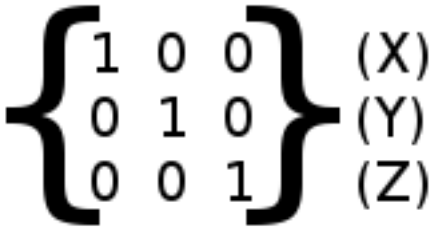
Godot tiene un tipo especial para una matriz 3x3, llamada *Matrix3*. Puede ser usada para representar una rotación y escala 3D. Los sub vectores pueden ser accedidos así:

```
var m = Matrix3()
var x = m[0] # Vector3
var y = m[1] # Vector3
var z = m[2] # Vector3
```

O, alternativamente como:

```
var m = Matrix3()
var x = m.x # Vector3
var y = m.y # Vector3
var z = m.z # Vector3
```

Matrix3 también es inicializado a Identidad por defecto:



Rotación in 3D

Rotación en 3D es mas complejo que en 2D (traslación y escala son iguales), porque rotación es una operación 2D implícita . Para rotar en 3D, un *eje*, debe ser seleccionado. La rotación, entonces, sucede alrededor de dicho eje.

El eje para la rotación debe ser un *vector normal*. Es decir, un vector que puede apuntar en cualquier dirección, pero cuyo largo debe ser uno (1.0).

```
#rotar en el eje Y
var m3 = Matrix3()
m3 = m3.rotated( Vector3(0,1,0), PI/2 )
```

Transform

Para agregar el componente final a la mezcla, Godot provee el tipo *Transform* . Transform tiene dos miembros:

- *basis* (base, de tipo *Matrix3*)
- *origin* (origen, de tipo *Vector3*)

Cualquier transformación 3D puede ser representada con Transform, y la separación de base y origen hace más sencillo trabajar con traslación y rotación por separado.

Un ejemplo:

```
var t = Transform()
pos = t.xform(pos) # transformar posición 3D
pos = t.basis.xform(pos) # (solo rotar)
pos = t.origin + pos # (solo trasladar)
```

7.2 Shaders

7.2.1 Generación de mallas con heightmap y shaders

Introducción

Este tutorial te ayudara a usar los shaders de Godot para deformar una malla de plano de forma que parezca terreno básico. Recuerda que esta solución tiene pros y contras.

Pros:

- Bastante fácil de hacer.
- Este enfoque permite la computación LOD de terreno.

- El heightmap puede ser usado en Godot para crear un normal map.

Contras:

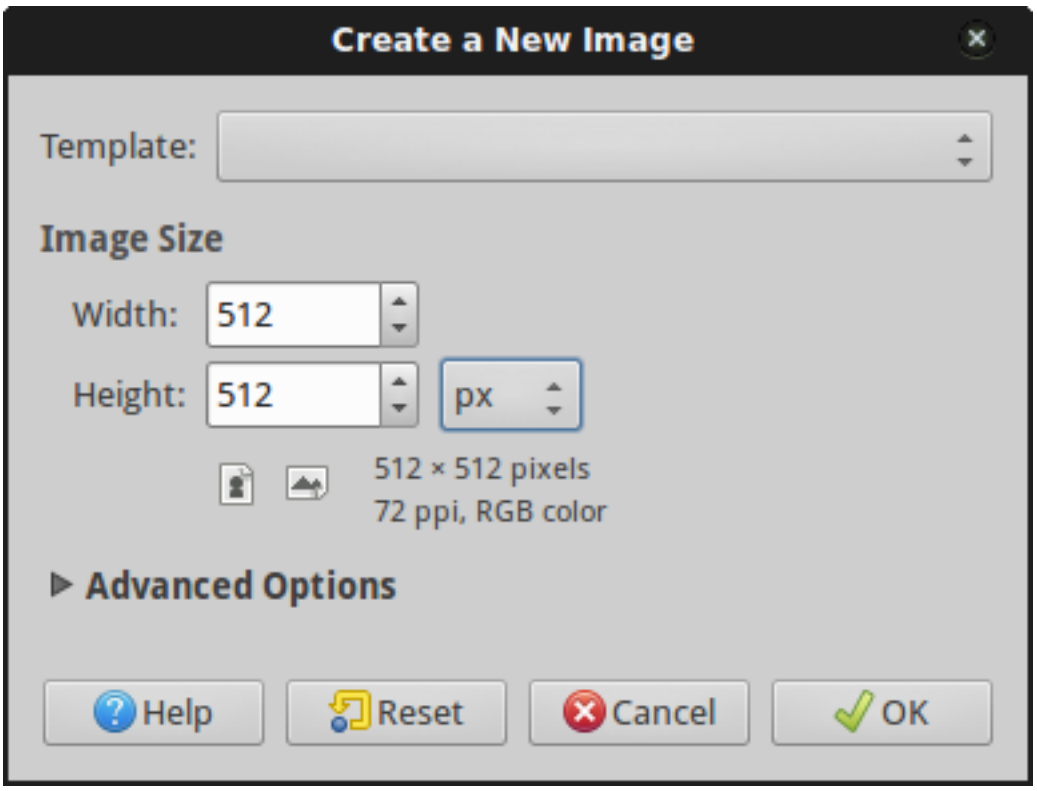
- El Shader de Vértices no puede re-computar las normales de las caras. Por lo que, si tu malla no es estática, este método no funcionara en materiales con sombras.
- Este tutorial usa una malla de plano importada de Blender al motor Godot. Godot puede crear mallas también.

Ve este tutorial como una introducción, no un método que deberías utilizar en tus juegos, excepto si piensas hacer LOD. De otra forma, esta probablemente no es la mejor forma.

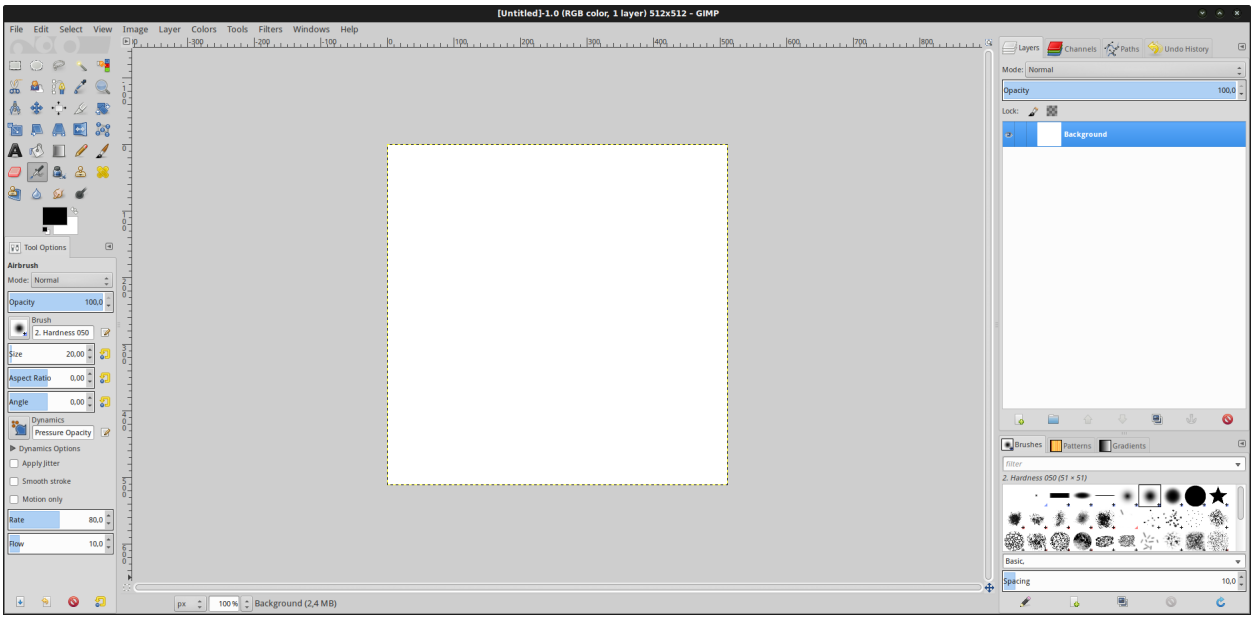
Sin embargo, vamos primero a crear un heightmap, o una representación del terreno. Para hacer esto, voy a usar GIMP, pero puedes usar el editor que te guste.

El heightmap (mapa de altura)

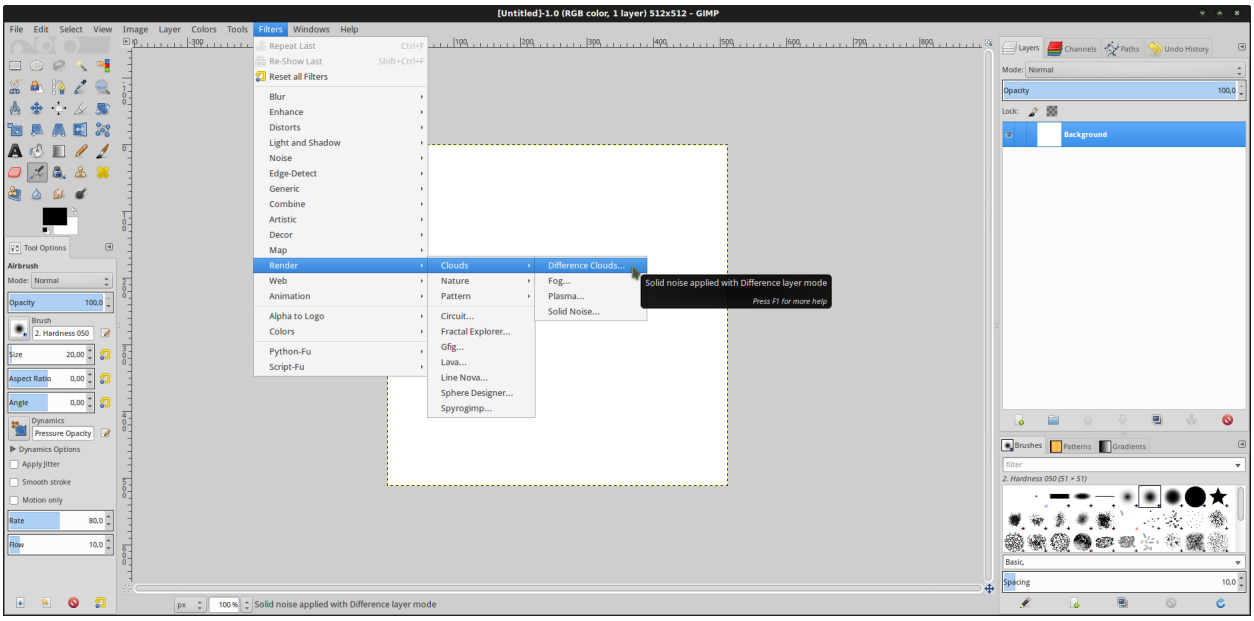
Usaremos unas pocas funciones del editor de imágenes GIMP para producir un simple heightmap. Ejecuta GIMP y crea una imagen cuadrada de 512x512 pixels.



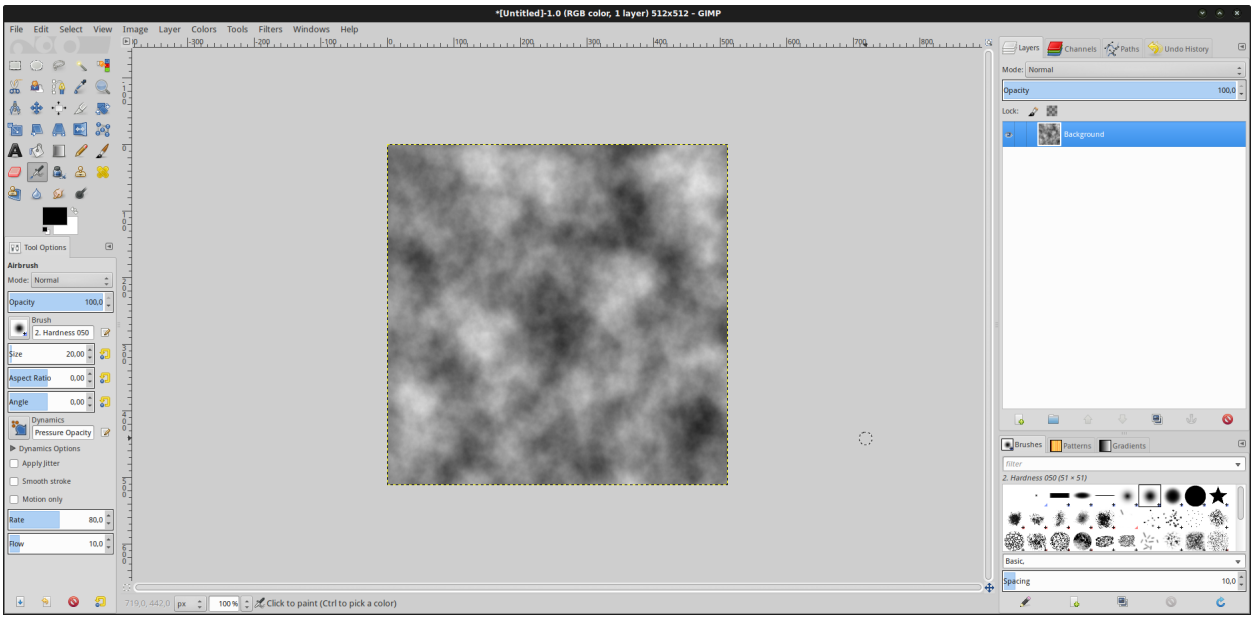
Ahora te encuentras frente a una nueva imagen cuadrada y en blanco.



Luego, usa un filtro para renderizar algunas nubes en este nueva imagen.



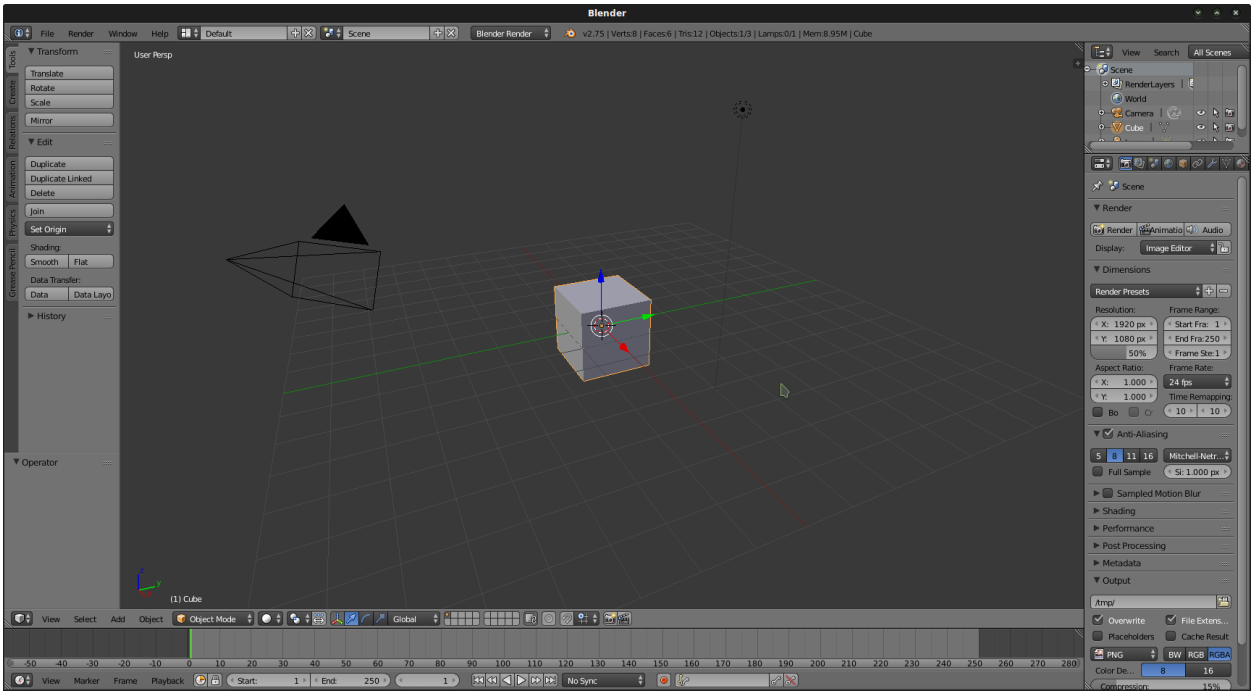
Parametriza este filtro de la forma que gustes. Un pixel blanco corresponde con el punto más alto del heightmap, un pixel negro corresponde al más bajo. Entonces, las regiones más oscuras son valles y las más brillantes montañas. Si quieres, puedes chequear “tileable” para renderizar un heightmap que puede ser clonado y embaldosados cerca uno del otro. El tamaño de X e Y no importa mucho mientras que sea lo suficientemente grande para proveer un terreno decente. Un valor de 4.0 o 5.0 para ambos está bien. Haz clic en el botón “New Seed” para tirar el dado y GIMP va a crear un nuevo heightmap aleatorio. Una vez que estés feliz con el resultado, haz clic en “OK”.



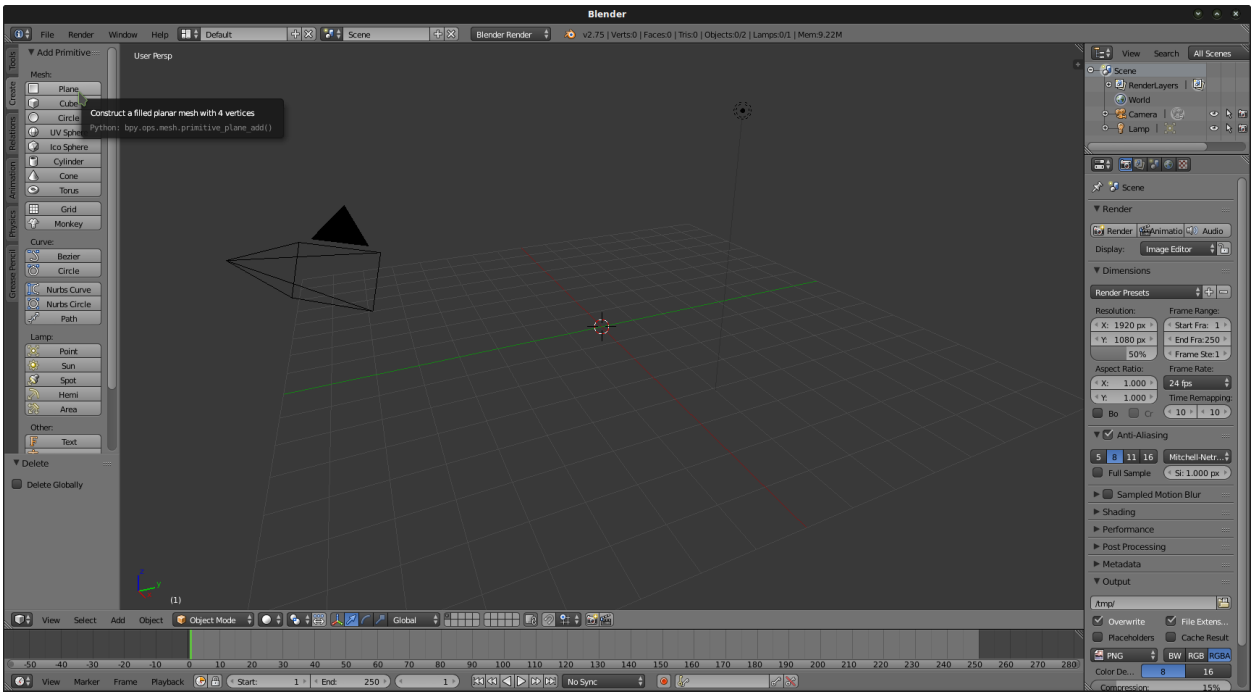
Puedes continuar editando tu imágen si deseas. Para nuestro ejemplo, vamos a mantener el heightmap com oesta, y vamos a exportarlo como un archivo PNG, como “heightmap.png”. Guárdalo en tu carpeta de proyecto de Godot.

La malla de plano

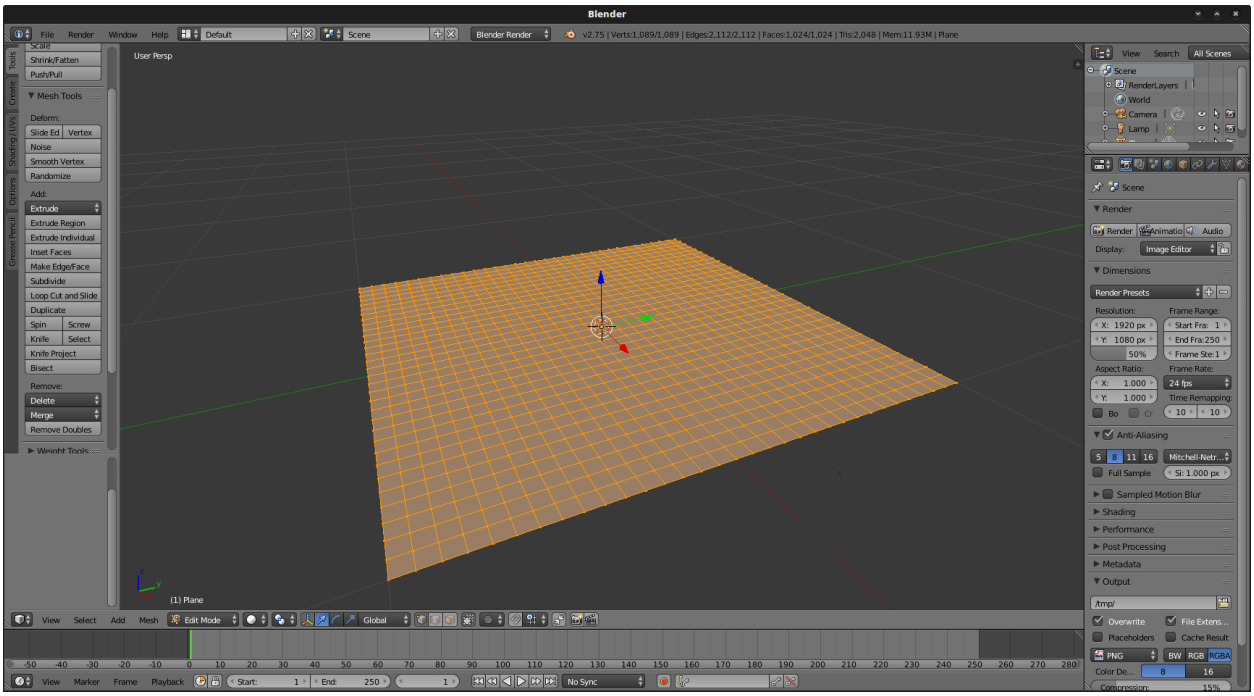
Ahora, vamos a necesitar una malla de plano para importar en Godot. Vamos a ejecutar Blender.



Remueve la malla de cubo inicial, luego agrega un nuevo plano a la escena.



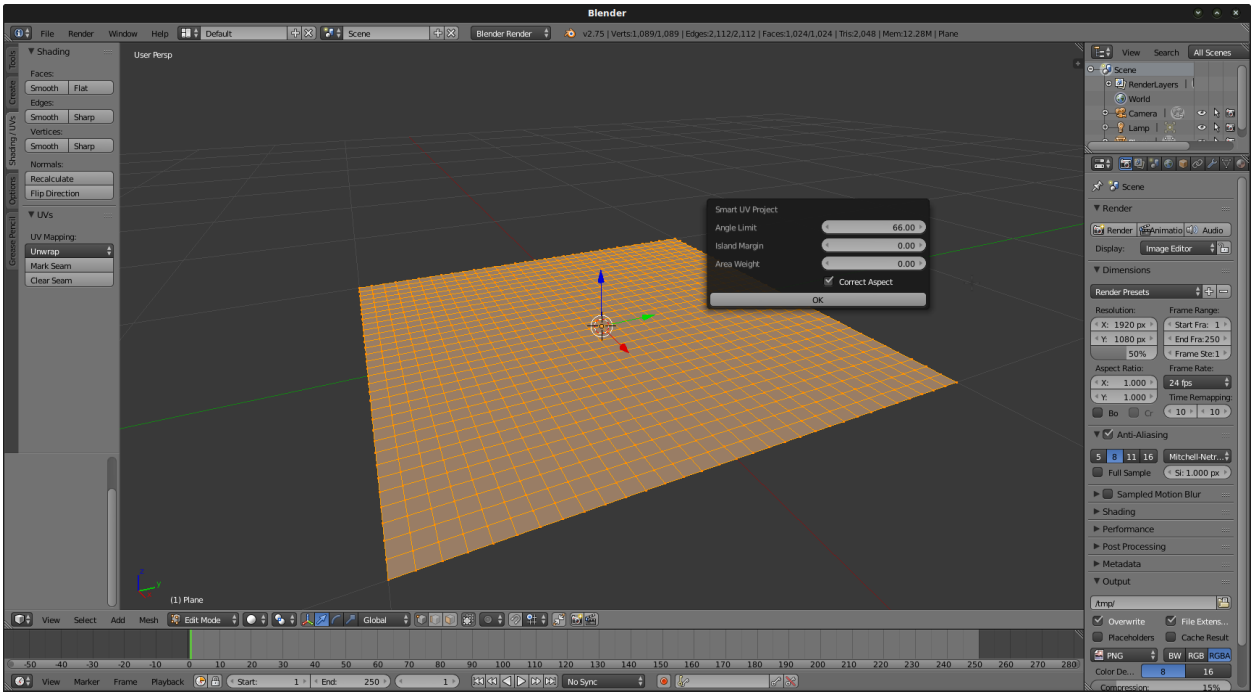
Haz un poco de zoom, luego cambia a Edit mode (tecla Tab) y en el grupo de botones de herramientas en el lado izquierda, oprime “Subdivide” 5 o 6 veces.



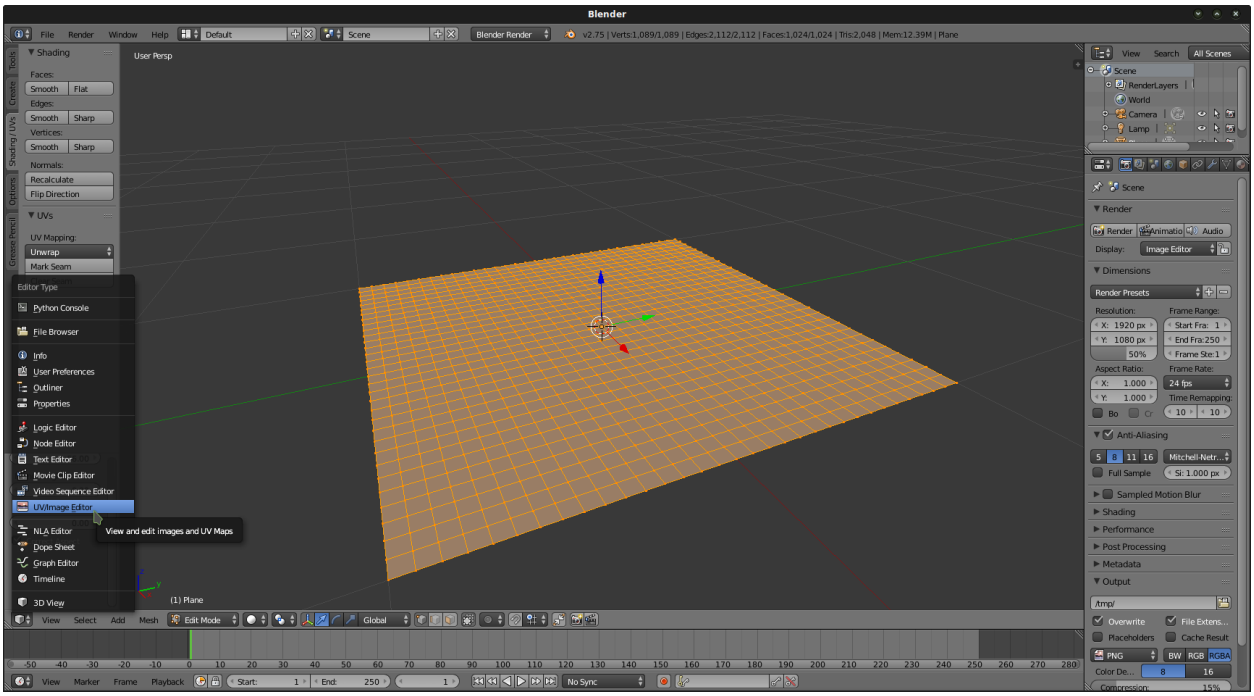
Tu malla ahora está subdividida, lo cual significa que agregamos vértices a la malla de plano que luego podremos mover. El trabajo aun no finalizo: para poder texturizar esta malla es necesario un mapa UV adecuado. Actualmente, el mapa UV por defecto contiene solo los 4 vértices que teníamos al principio. Sin embargo, ahora tenemos más, y queremos poder texturizar correctamente sobre la malla entera.

Si todos los vértices de tu malla no están seleccionados, selecciónalos todos (toca “A”). Deben aparecer anaranjados, no negros. Luego, en el grupo de botones Shading/UVs a la izquierda, haz clic en el botón “Unwrap” (o simplemente

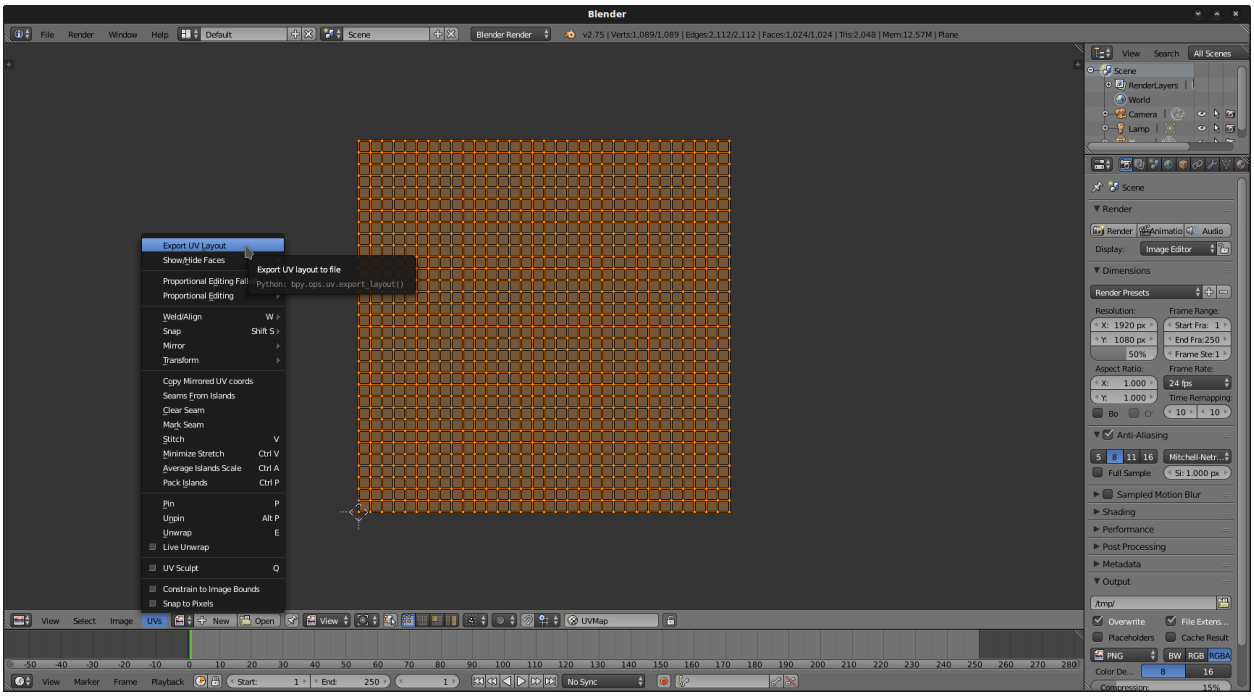
toca “U”) y selecciona “Smart UV Project”. Mantén las opciones por defecto y dale “Ok”.



Ahora, necesitamos cambiar nuestra vista a “UV/Image editor”.



Selecciona todos los vértices de nuevo (“A”) luego en el menú UV, selecciona “Export UV Layout”.

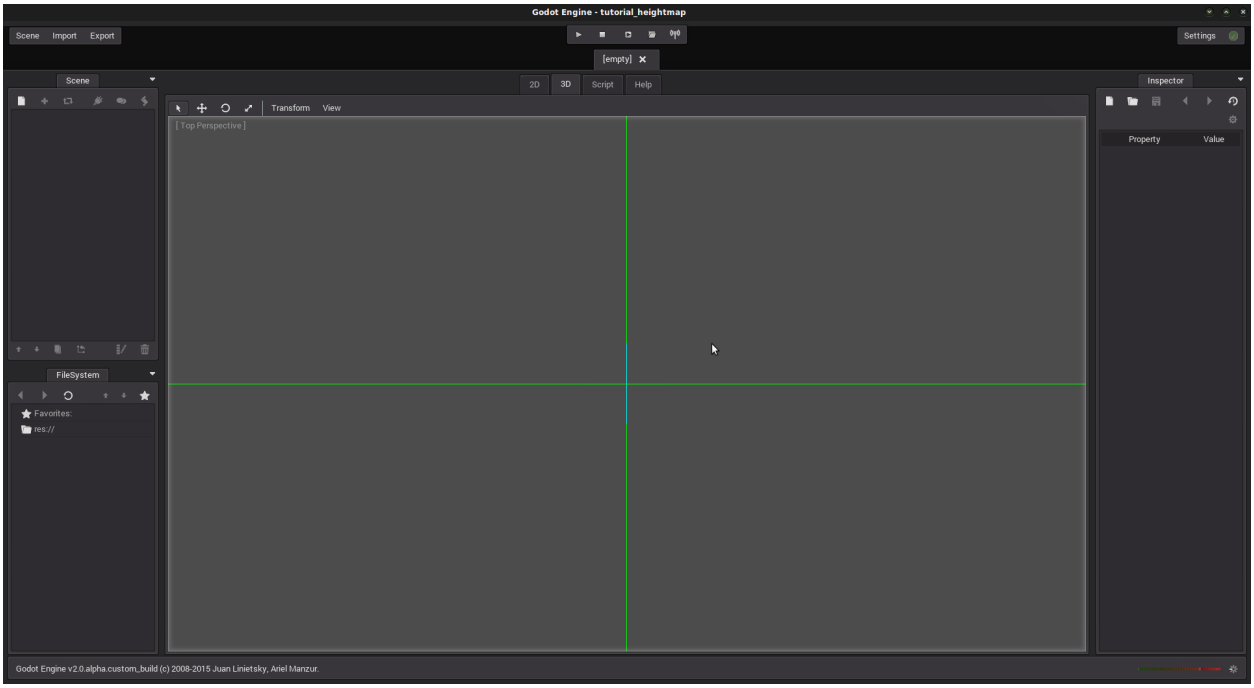


Exporta el layout como un archivo PNG. Nómbralo "plane.png" y guárdalo en tu carpeta de proyecto Godot. Ahora, vamos a exportar nuestra malla como un archivo OBJ. En la parte superior de la pantalla, haz clic en "File/Export/Wavefront (obj)". Guarda tu proyecto como "plane.obj" en tu carpeta de proyecto Godot.

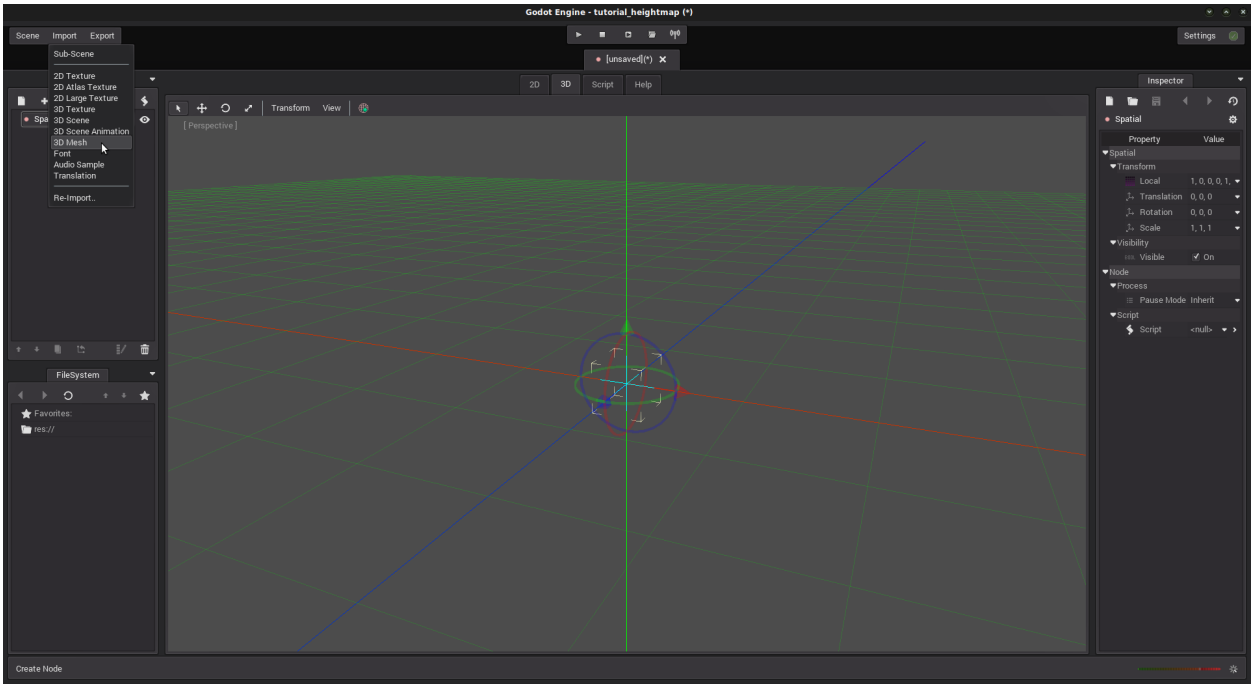
Magia de shaders

Ahora abramos el Editor Godot.

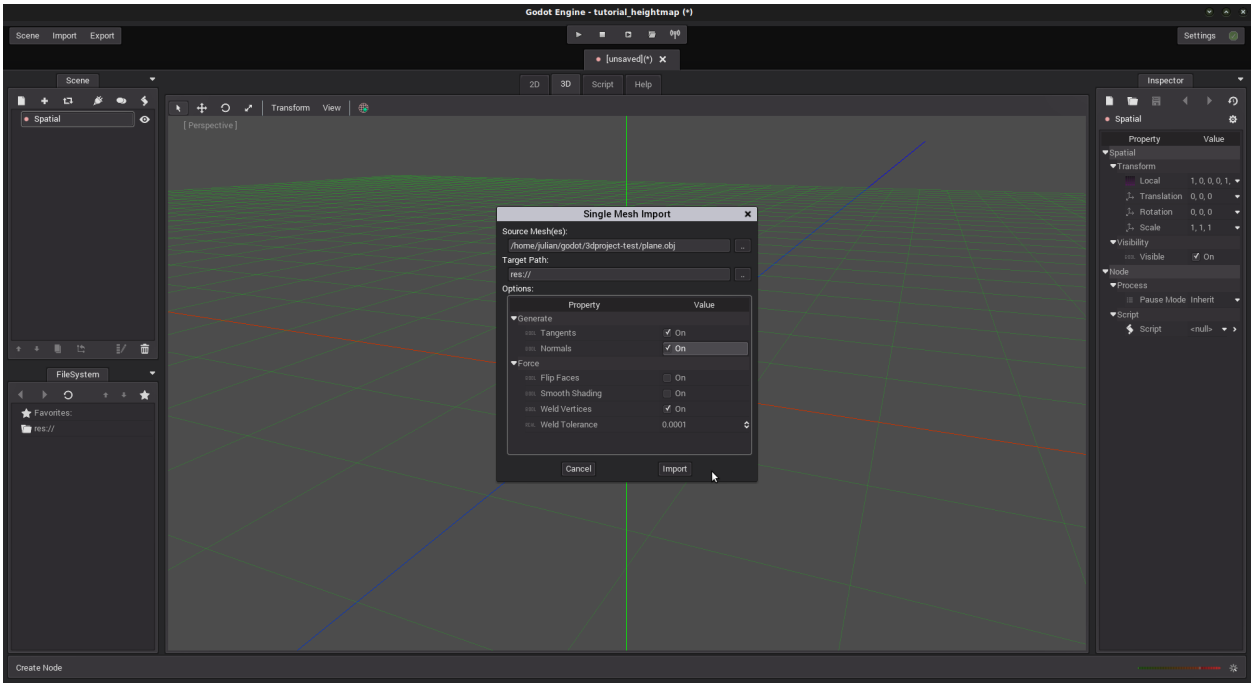
Crea un nuevo proyecto en la carpeta que creaste previamente y nómbralo como quieras.



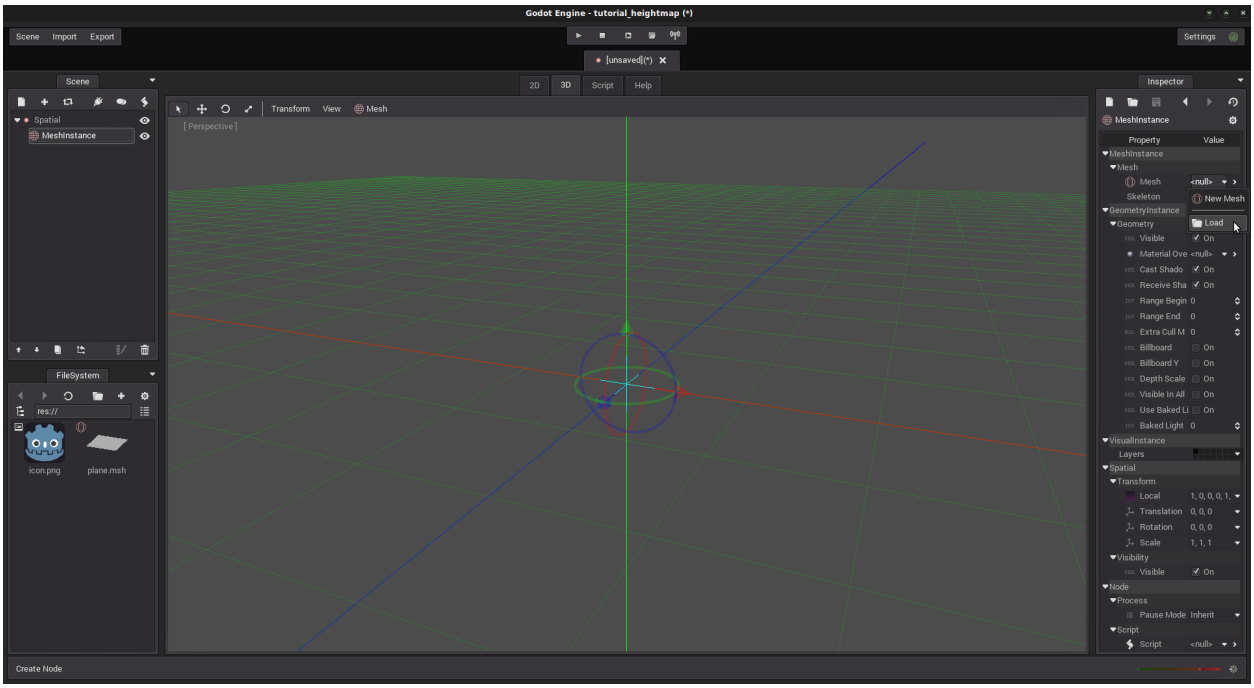
En nuestra escena por defecto (3D), crea un nodo raíz “Spatial”. Luego, importa el archivo de malla OBJ. Haz clic en “Import”, selecciona “3D Mesh” y selecciona tu archivo plane.obj, ajusta el camino de destino como “/” (o lo que sea que tengas en tu carpeta de proyecto).



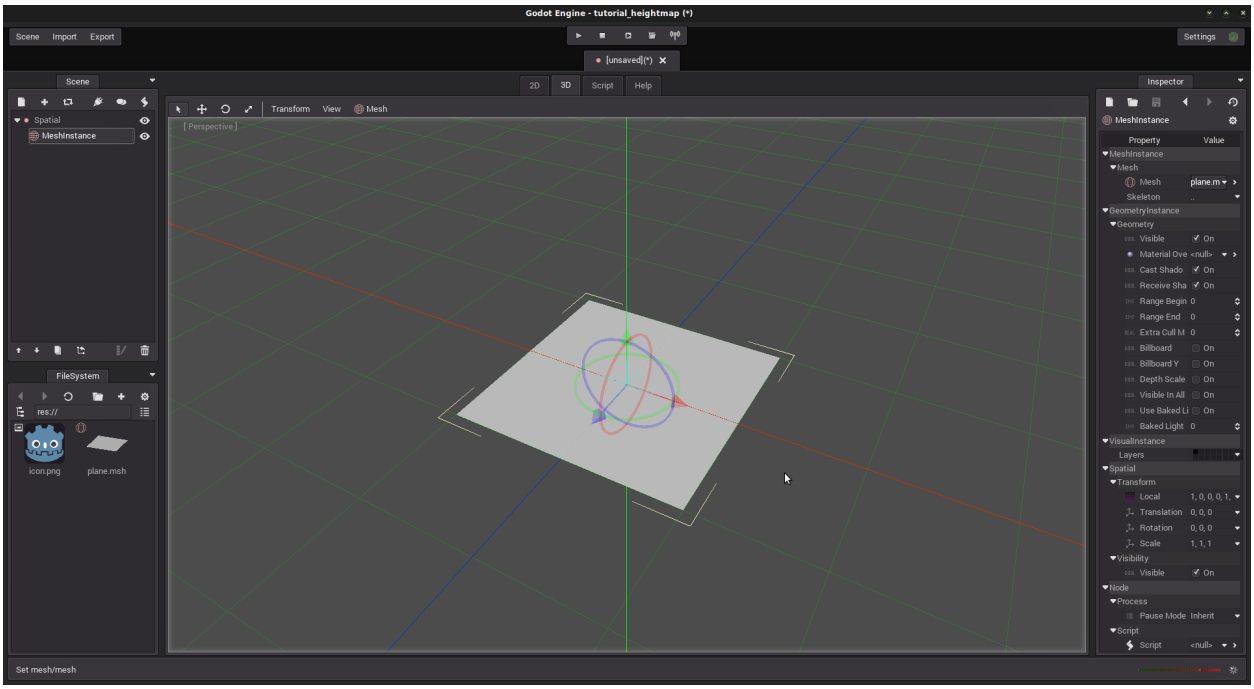
Me gusta chequear “Normals” en la ventana emergente de importación de modo que el importador también considere las normales de las caras, lo que puede ser útil (aun si no las usamos en este tutorial). Tu malla ahora es mostrada en el FileSystem en “res://”.



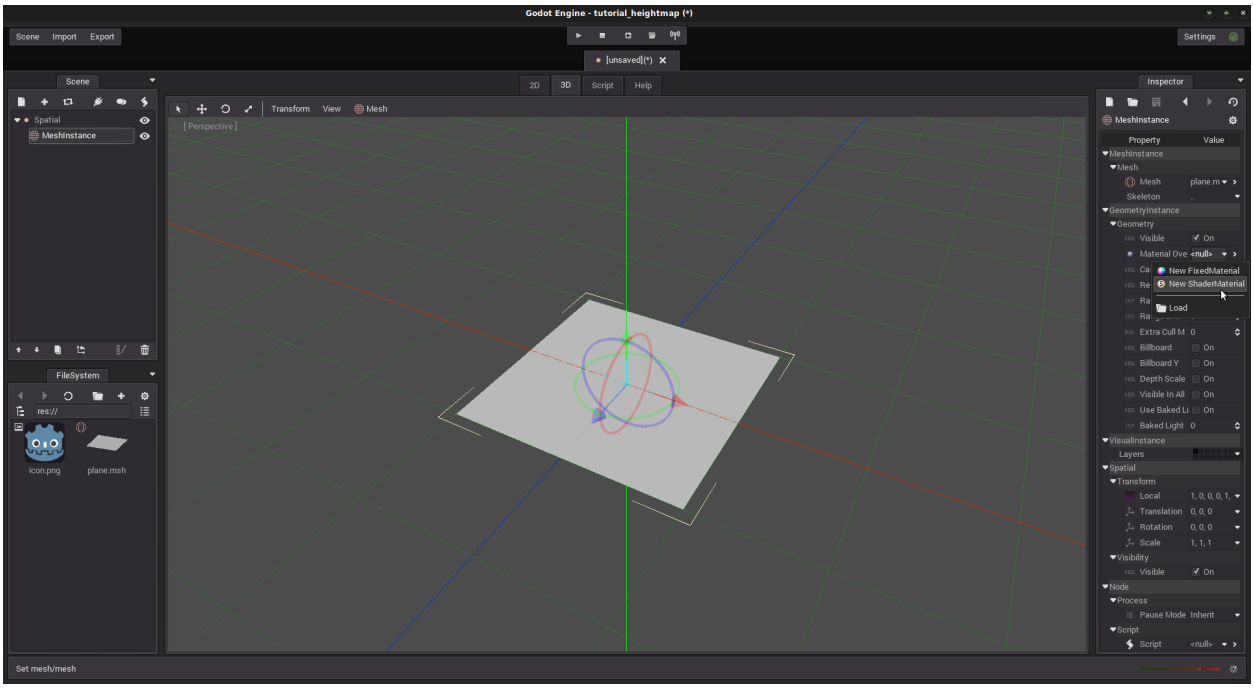
Crea un nodo MeshInstance. En el Inspector, carga la malla de recién importamos. Selecciona “plane.msh” y dale OK.



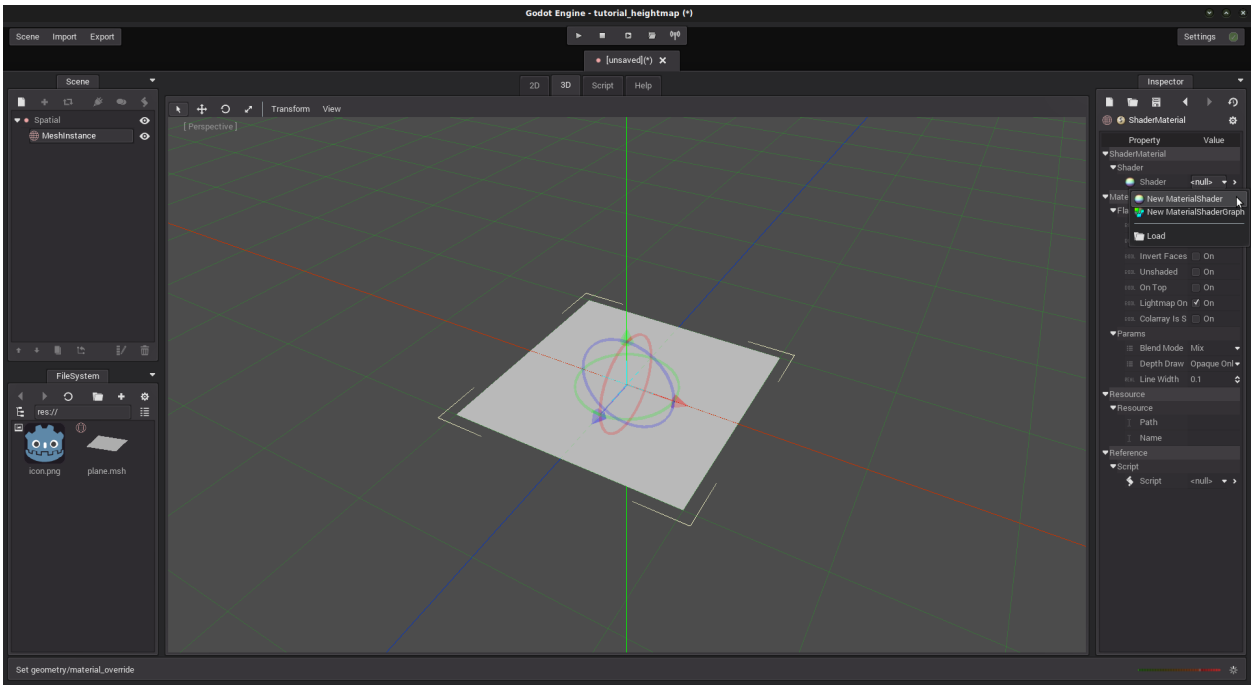
Genial! Nuestro plano esta ahora renderizado en la vista 3D.



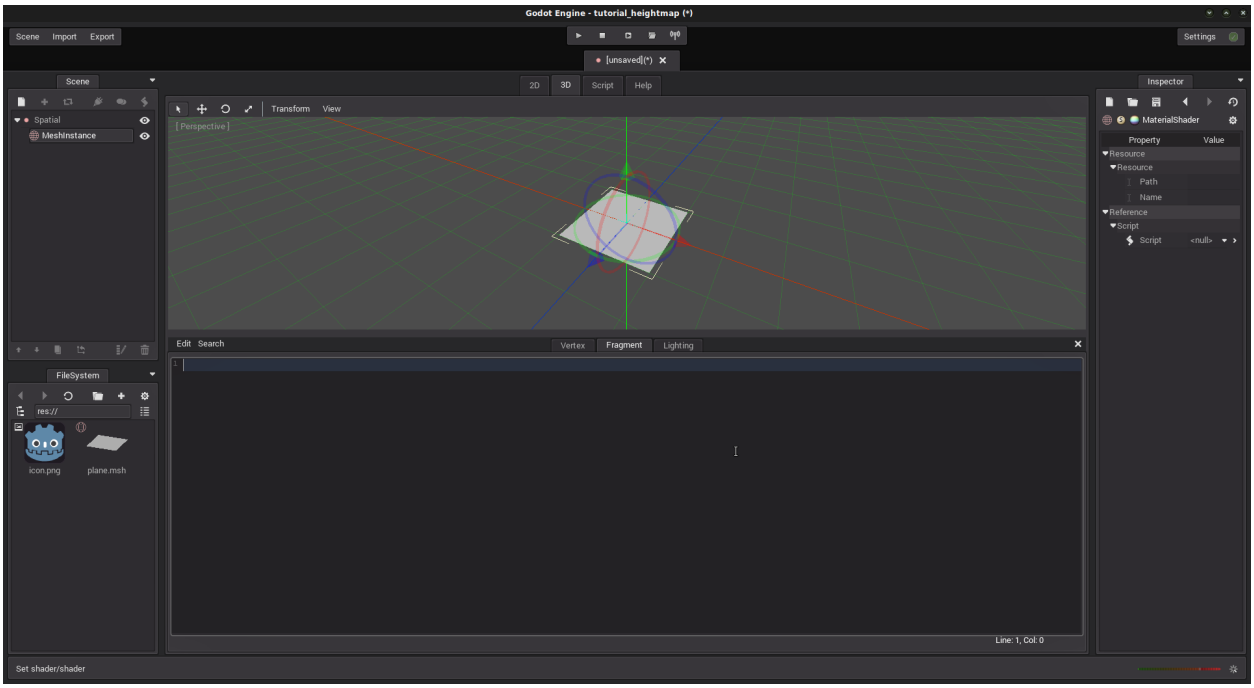
Es momento de agregar otras cosas de shaders. En el inspector, en la línea “Material Override”, agrega un “New ShaderMaterial”. Edítalo al hacer clic en el botón “>” justo a su derecha.



Tienes dos formas de crear un shader: por código (MaterialShader), o usando un shader graph (MaterialShaderGraph). La segunda es más visual, pero no la cubriremos por ahora. Crea un "New MaterialShader".



Edítalo haciendo clic en el botón ">" justo a su derecha. El editor de Shaders se abre.



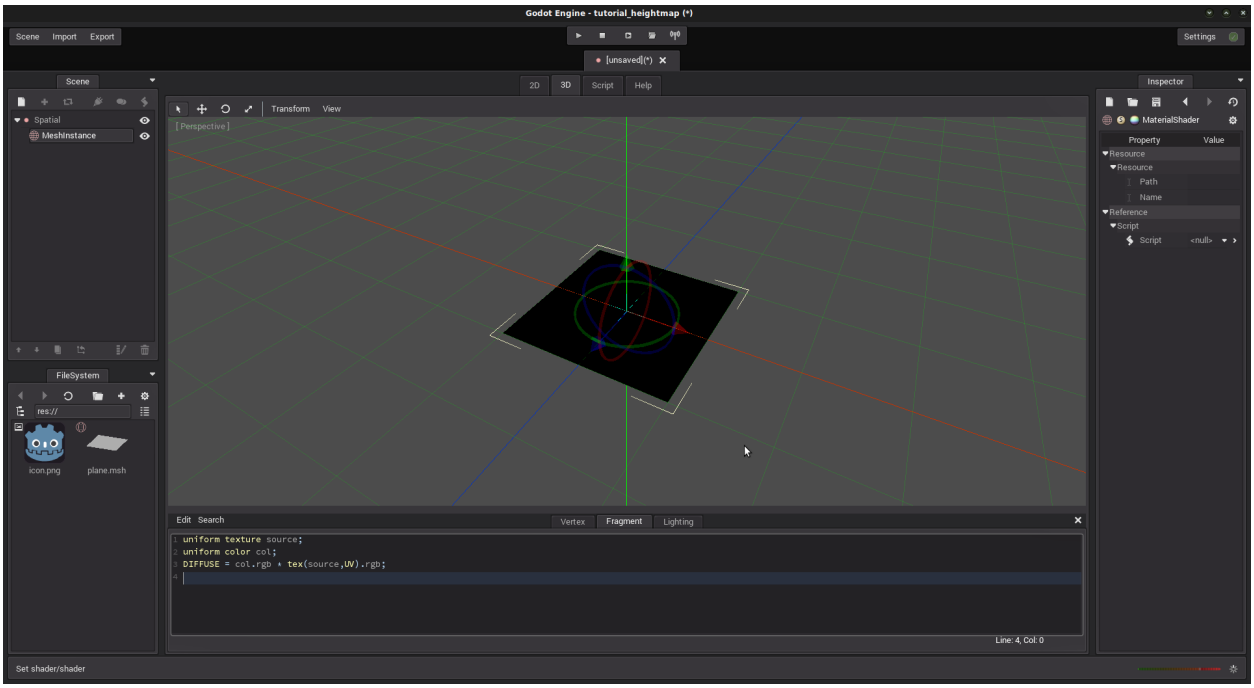
La pestaña Vertex es para el shader de vértice, y la pestaña Fragment es para el shader de fragmento. No hay necesidad de explicar que hacen ambos, correcto? Si no, ve a la página *Shading language*. De otra forma, empecemos con el Fragment shader. Este es usado para texturizar el plano usando una imagen. Para este ejemplo, lo texturizaremos con la imagen de heightmal propia, de forma que veamos las montañas como regiones más brillantes y los cañones como regiones más oscuras. Usa este código:

```
uniform texture source;
uniform color col;
DIFFUSE = col.rgb * tex(source,UV).rgb;
```

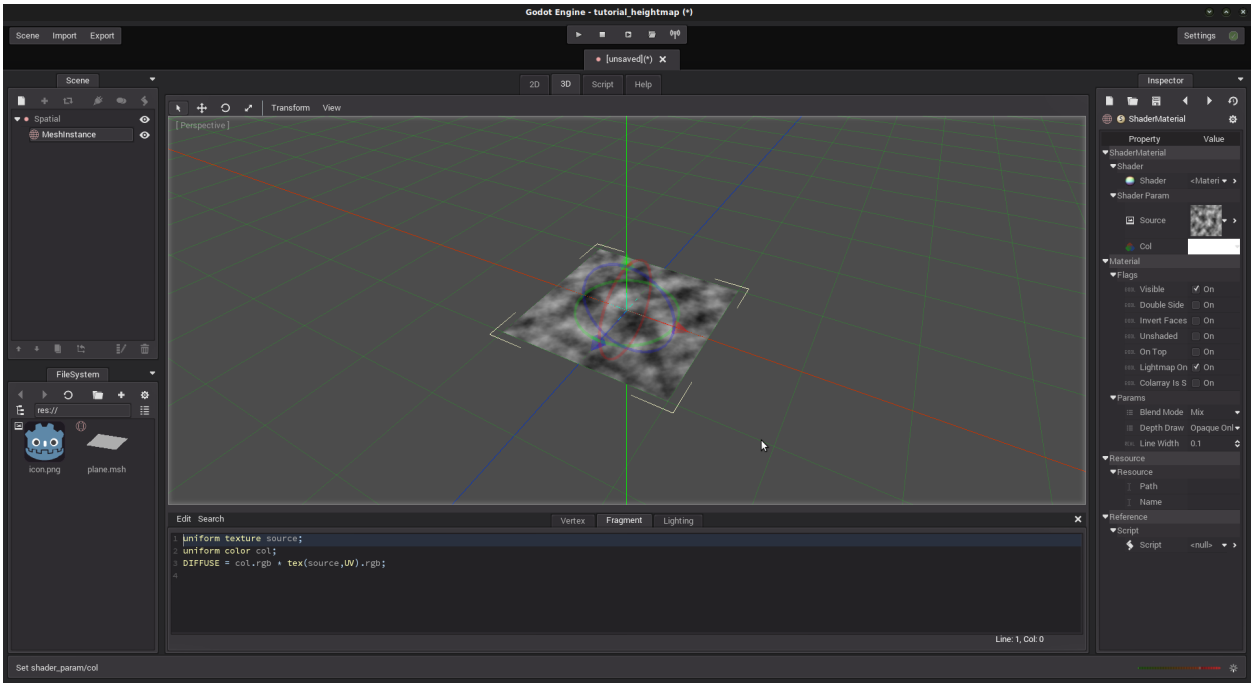
Este shader es muy simple (en realidad viene de la página *Shading language*). Lo que hace básicamente es tomar 2 parámetros que tenemos que proveer desde fuera del shader ("uniform"):

- el archivo de textura
- un color
- a color Luego, vamos a multiplicar cada pixel de la imagen por un `tex(source, UV).rgb` por el color definido `col` y lo ajustamos a la variable DIFFUSE, que es el color renderizado. Recuerda que la variable `UV` es una variable de shader que retorna la posición 2D de la variable DIFFUSE, la cual es el color renderizado. Recuerda el pixel en la imagen de textura, de acuerdo al vértice con el que estamos tratando. Ese es el uso del UV Layout que hicimos antes. El color `col` no es en realidad necesario para mostrar la textura, pero es interesante jugar y ver lo que hace, correcto?

Sin embargo, el plano se muestra negro! Esto es porque no ajustamos el archivo de textura y el color a usar.



En el Inspector, haz clic en el botón “Previous” para ir atrás al ShaderMaterial. Aquí es donde tu quieres ajustar la textura y el color. En “Source”, haz clic en “Load” y selecciona el archivo de textura “heightmap.png”. Pero la malla aun es negra! Esto es porque nuestro Fragment shader multiplica cada valor de pixel de la textura por el parámetro `col`. Sin embargo, este color está actualmente ajustado a negro (0,0,0), y como sabes, 0*x = 0 ;) . Solo cambia el parámetro `col` a otro color para que tu textura aparezca.



Bien. ahora, el Vertex Shader.

El Vertex Shader es el primer shader en ser ejecutado. Trata con los vértices.

Haz clic en la pestaña “Vertex” para cambiar, y pega este código:

```
uniform texture source;
uniform float height_range;
vec2 xz = SRC_VERTEX.xz;
float h = tex(source, UV).g * height_range;
VERTEX = vec3(xz.x, h, xz.y);
VERTEX = MODELVIEW_MATRIX *  VERTEX;
```

Este shader usa dos parámetros “uniform”. El parámetro‘‘source‘‘ ya esta ajustado para el fragment shader. Entonces, la misma imagen será usada en este shader como el heightmap. El parámetro `height_range` es un parámetro que usaremos para incrementar el efecto de la altura.

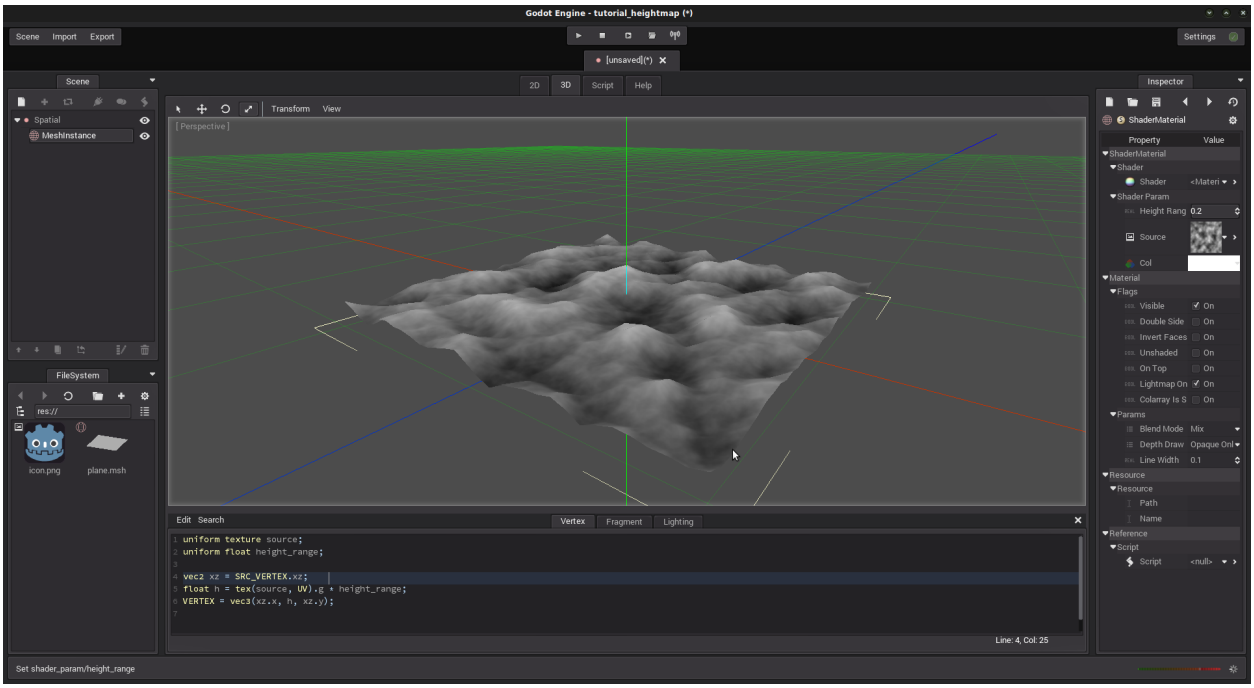
En la línea 3, guardamos la posición x y z de el SRC_VERTEX, porque no queremos que cambien : el plano debe permanecer cuadrado. Recuerda que el eje Y corresponde a la “altura”, la cual es lo único que queremos cambiar con el heightmap.

En la línea 4, computamos una variable `h` al multiplicar el valor de pixel por la posición UV y el `height_range`. Como el heightmap es una imagen en escala de grises, todos los canales r, g y b contienen el mismo valor. Use `g`, pero cualquiera de r, g y b tendría el mismo efecto.

En la línea 5, ajustamos la posición actual del vértice a la posición (xz.x, h, xz.y). sobre xz.y recuerda que su tipo es “vec2”. Entonces, sus componentes son x e y. El componente y simplemente contiene la posición z que ajustamos en la línea 3.

Finalmente, en la línea 6, multiplicamos el vértice por la matriz model/view de forma de ajustar su posición de acuerdo a la posición de la cámara. Si tratas de comentar esta línea, veras que la malla se comporta de forma extraña cuando mueves y rotas la cámara.

Eso esta todo bien, pero nuestro plano permanece chato. Esto es porque el valor de `height_range` es 0. Incrementa dicho valor para observar la malla distorsionarse y tomar la forma del terreno que ajustamos antes:



CAPÍTULO 8

Conducto de Asset

8.1 General

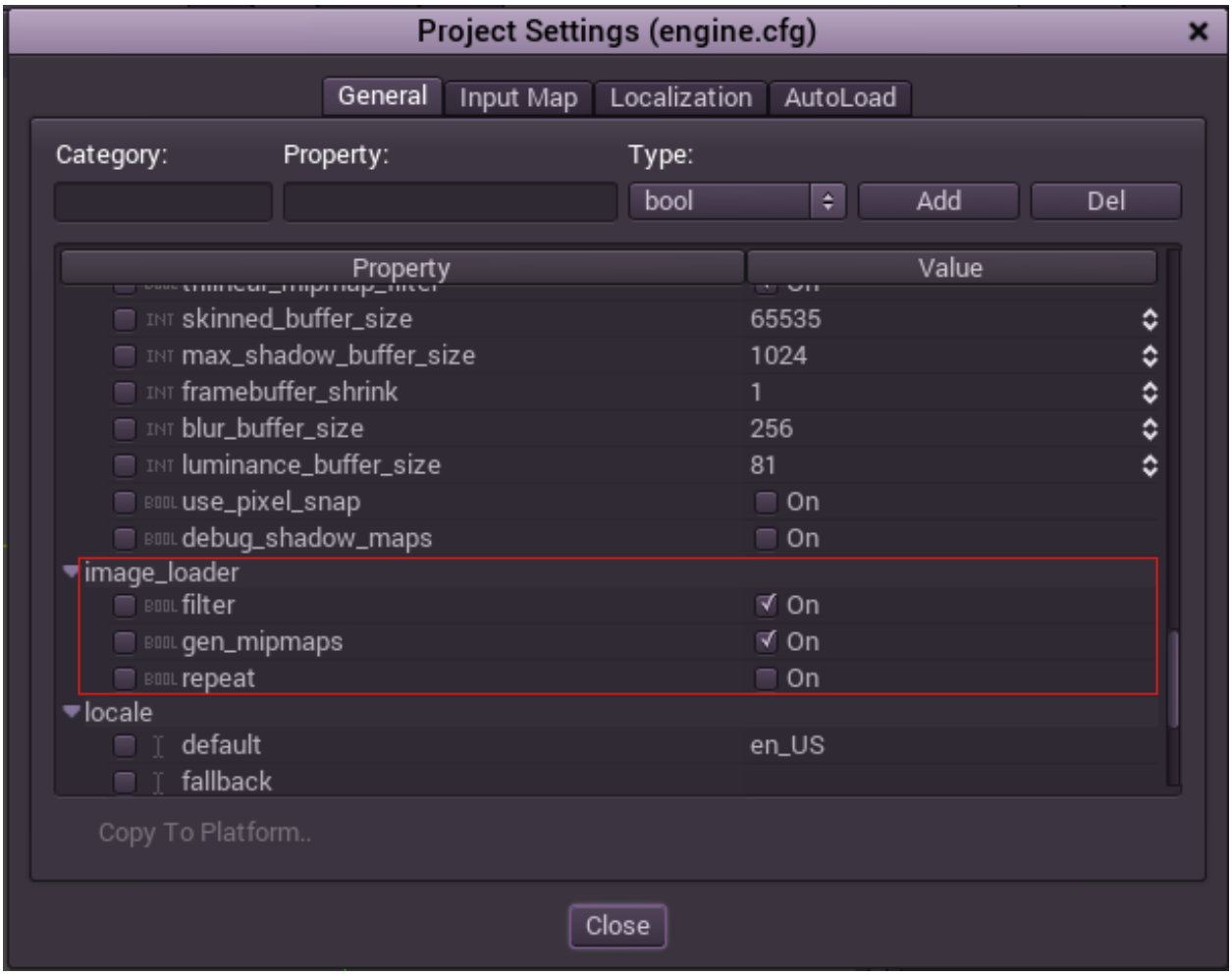
8.1.1 Administrando archivos de imagen

Si haz leído los tutoriales previos en doc_resources‘y :ref:‘doc_filesystem, en este punto tu sabes que las imágenes regulares (.png, .jpg, etc.) son tratadas como recursos regulares en Godot.

A diferencia de los recursos de texturas (.tex files), los archivos de imagen no contienen información extra de mosaico (repetición de texturas), mipmaps o filtrado. Editar esta información y guardar la textura de nuevo no tendrá ningún efecto, ya que dichos formatos no pueden contener esa información.

Cargador de imágenes

La carga de imágenes es hecha por el “image loader”. El comportamiento del cargador de imágenes para todos los archivos de imagenes puede ser cambiado en el dialogo de Configuración de Proyecto (Escena -> Configuración de proyecto), Hay una sección con valores que son usados para todos los recursos de imágenes:



Opciones del cargador de imágenes

Filter (Filtro)

Filter es usado cuando la imagen es estirada más que tu tamaño original, por lo que un texel en la imagen es más grande que el pixel en la pantalla. Apagarlo genera un efecto estilo retro:

Repeat (Repetir)

Repeat es usado principalmente para texturas 3D, por lo que está apagado por defecto (las texturas son importadas con la escena y usualmente no están en el proyecto como archivos). Cuando se usan coordenadas UV (algo que no es común en 2D), y el valor UV va más allá de 0,0,1,1 rect, la textura se repite en lugar de restringirse al borde.

Mipmaps

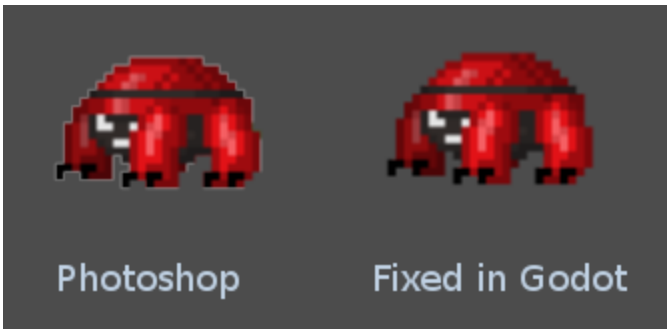
Cuando la opción mipmaps es habilitada, Godot va a generar mipmaps. Mipmaps son versiones de una imagen encogidas a la mitad en ambos ejes, recursivamente, hasta que la imagen tiene un tamaño de 1 pixel. Cuando el hardware 3D necesita encoger la imagen, encuentra el mipmap más grande desde el cual puede escalar, y escala desde allí. Esto mejora el rendimiento y la calidad de imagen.

Cuando los mipmaps son deshabilitados, las imágenes empiezan a distorsionarse mucho cuando se encojen excesivamente:

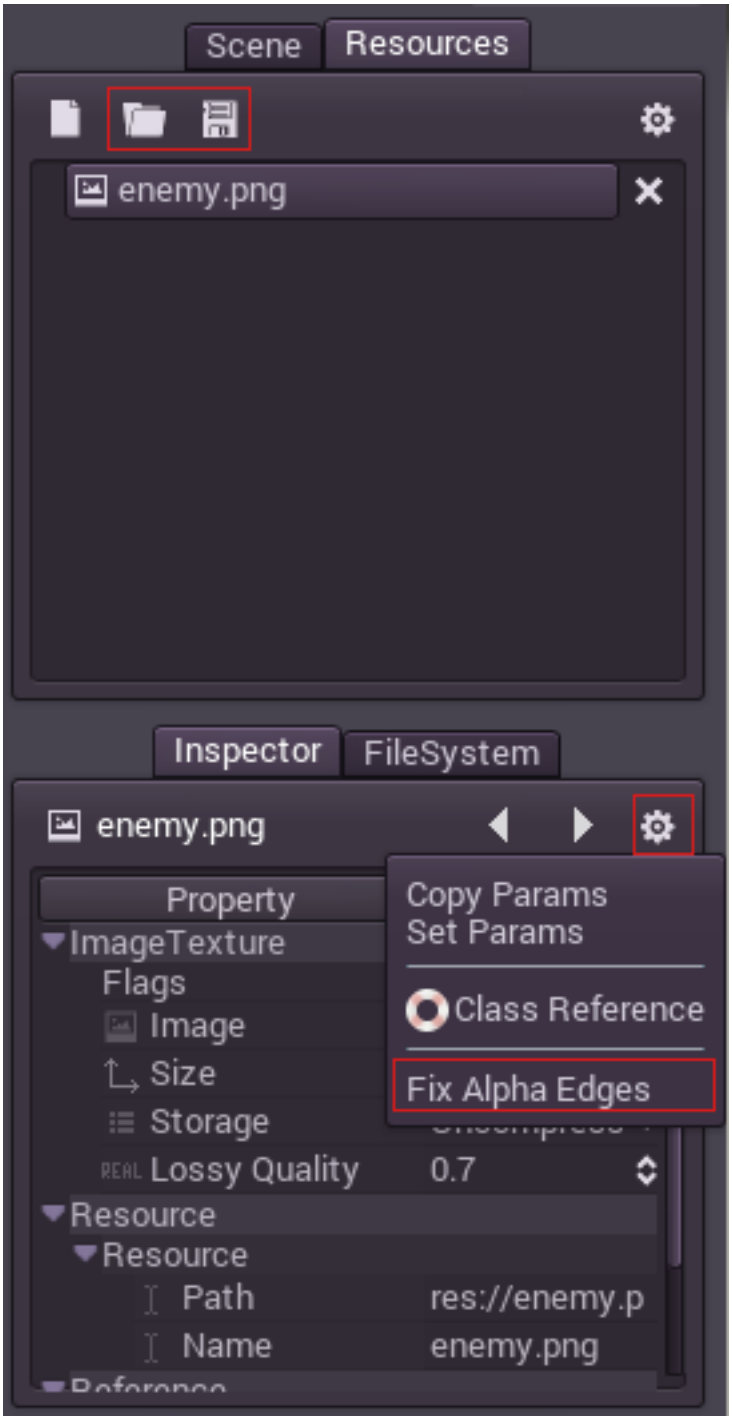
Alpha blending (Mezcla Alpha)

La ecuación de mezcla usada por aplicaciones como Photoshop es demasiado compleja para tiempo real. Hay aproximaciones mejores como alpha pre-multiplicado, pero imponen más estrés en el conducto de assets. Al final, terminamos con texturas que tienen artefactos en los bordes, porque las aplicaciones como Photoshop guardan los pixels blancos en áreas completamente transparentes. Esos pixels blancos se terminan mostrando gracias al filtro de textura (cuando está activo).

Godot tiene una opción para arreglar los bordes de una imagen (al pintar pixels invisibles con el mismo color de sus vecinos visibles).



Para hacer esto, abre la imagen desde la pestaña recursos, o edítala desde las propiedades de editor de otro nodo o recurso, luego ve a las opciones del objeto y selecciona “Fix Alpha Edges”, y guardalo.



Debido a que arreglar esto en tantas imágenes puede ser algo molesto, tanto Texture Import y Image Export pueden también hacer esta operación.

Importando texturas

A veces, puede desearse cambiar los ajustes de arriba por imagen. Desafortunadamente, los ajustes de image loader son globales. Los flags de textura no pueden ser guardados en un archivo regular .png o .jpg.

Para esos casos, la imagen puede ser importada como una textura (.tex), donde los falgs individuales pueden ser

cambiados. Godot también mantiene el seguimiento del archivo original y lo re-importara si cambia.

Importar también permite la conversión a otros formatos (WebP, o compresión RAM) la cual puede ser usada en algunos casos. Más información en la pagina *Importing textures*.

Exportando texturas

También es posible convertir imágenes a otros formatos (WebP o compresión RAM) al exportar, así como instruir al exportador para crear un Atlas para un conjunto de imágenes. También es posible pedir al exportador que escale todas las imágenes (o grupos seleccionados).

Mas información en la paginae *Exporting images*.

8.2 Import

8.2.1 Import process

What is it for?

When Godot was created, it was probably after several failed and not so failed engine attempts (well, each attempt failed a little less.. and so on). One of the most difficult areas of creating game engines is managing the import process. That means, getting the assets that artists make into the game, in a way that functions optimally.

Artists use certain tools and formats, and programmers would rather have their data into a different format. This is because artists put their focus on creating assets with the best quality possible, while programmers have to make sure they actually run at decent speed (or run at all), use a certain amount of memory, and don't take ages loading from disk.

One would think that just writing a converter/importer would be enough, but this is not all there is to it. The same way programmers iterate several times over their code, artists keep making changes to their assets. This generates some bottleneck, because *someone* has to keep re-importing that artwork right? And importing assets is often something that has to be agreed by both parties, as the programmer needs to decide how the artwork is imported and the artists needs to see how it looks.

The goal to establishing an import process is that both can agree on how the rules under which the assets are going to be imported the first time, and the system will apply those rules automatically each time the asset is re-imported.

Godot does not do the re-import process automatically, though. It gives the team the option to do it at any time (a red icon on the top right of the screen, allows the ability to do it at any desired time).

Does it always work?

The aim of the import system is that it works well enough for most common cases and projects. What is there has been tested and seems to cover most needs.

However, as mentioned before, this is one of the most difficult areas of writing a game engine. It may happen often (specially on large projects, ports, or projects with unusual requirement) that what is provided is not enough. It's easy to say that the engine is open source and that the programmer should make their own if they don't like what is there, but that would be making a huge disservice to the users and not the right attitude. Because of that, we made sure to provide as many tools and helpers as possible to support a custom import process, for example:

- Access to the internals of almost all data structures is provided to the scripting and C++ API, as well as saving and loading in all supported file formats.
- Some importers (like the 3D asset importer) support scripts to modify the data being imported.

- Support for creating custom import plugins is also provided, even for replacing the existing ones.
- If all else fails, Godot supports adding custom resource loaders, to load data in alternative formats, without intermediate conversion.

Both the import system and the custom tools provided will improve over time as more use cases are revealed to us.

Importing assets

Source asset location

To begin, it is a good idea to define where the original assets created by the artists (before they are imported) will be located. Normally, Godot does not mind much about the location, but if the project has several developers, it is a good idea to understand the simple rule for it to work for everyone.

First of all, it would be really good for this location to **not** be inside the project path (where engine.cfg is located, or any sub-folder). Godot expects regular resources in there, and may consider many of the files used as source art as regular resources. This would lead to it bundling all of them when the project is exported, something which is undesired.

Now that it is clear that this location must be outside the project folder, the rule that Godot uses to reference external assets can be explained. When an asset is imported, the engine stores a relative path from the project path to the asset (In windows, this works as long as they are on the same drive, otherwise an absolute path is stored). This ensures that the same asset can be re-imported in another computer.

The usual approach to this, when using a VCS such as Subversion, Perforce or GIT, is to create the project in a subfolder, so both it and the source assets can be committed to a same repository. For example:

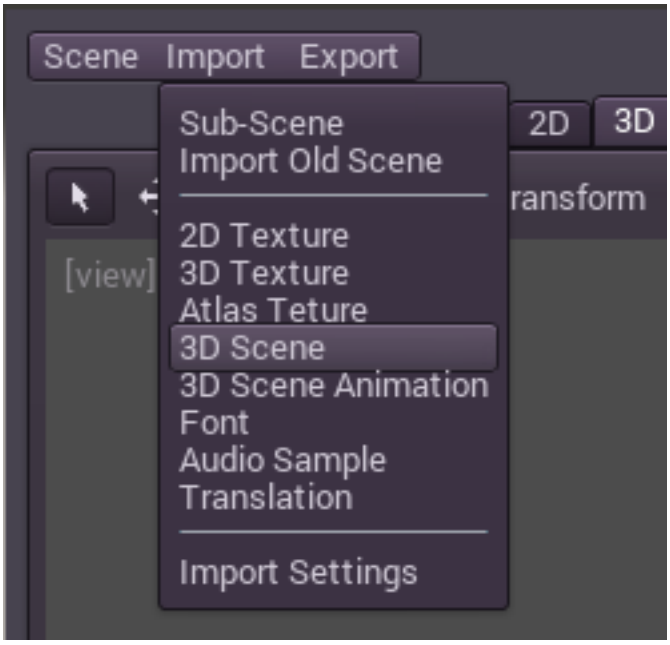
Repository layout:

```
source_assets/sfx/explosion.wav
source_assets/sfx/crash.wav
source_assets/fonts/myfont.ttf
source_assets/translation/strings.csv
source_assets/art/niceart.psd
game/engine.cfg
```

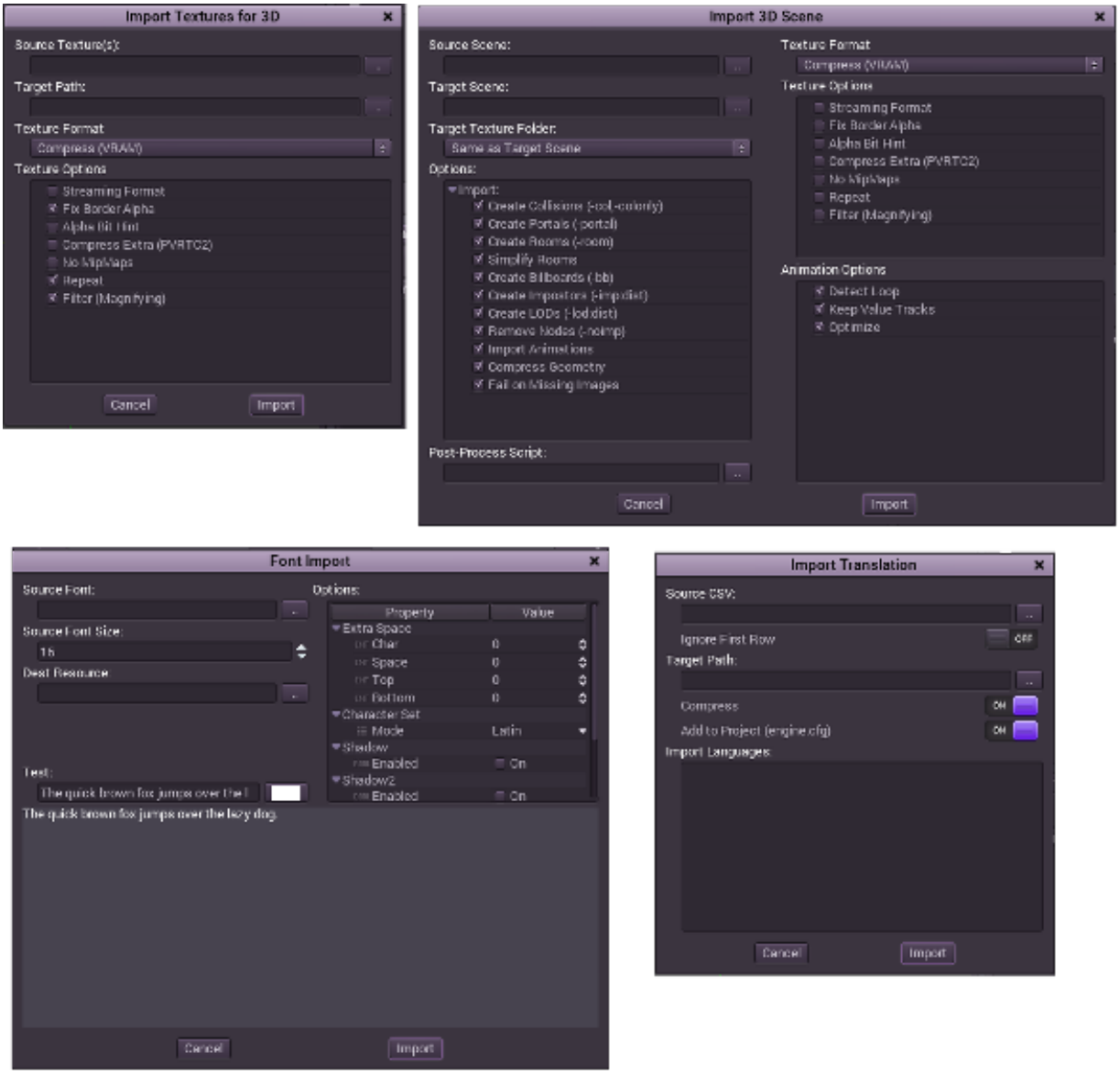
In the above example, artists, musician, translators, etc. can work in the source_assets/ folder, then import the assets to the game/ folder. When the repository is updated, anyone can re-import the assets if they changed.

Import dialogs

Godot provides for importing several types of assets, all of them can be accessed from the import dialog:



Each of the dialog shares a similar function, a source file (or several of them) must be provided, as well as a target destination inside the project folders. Once imported, Godot saves this information as metadata in the imported asset itself.



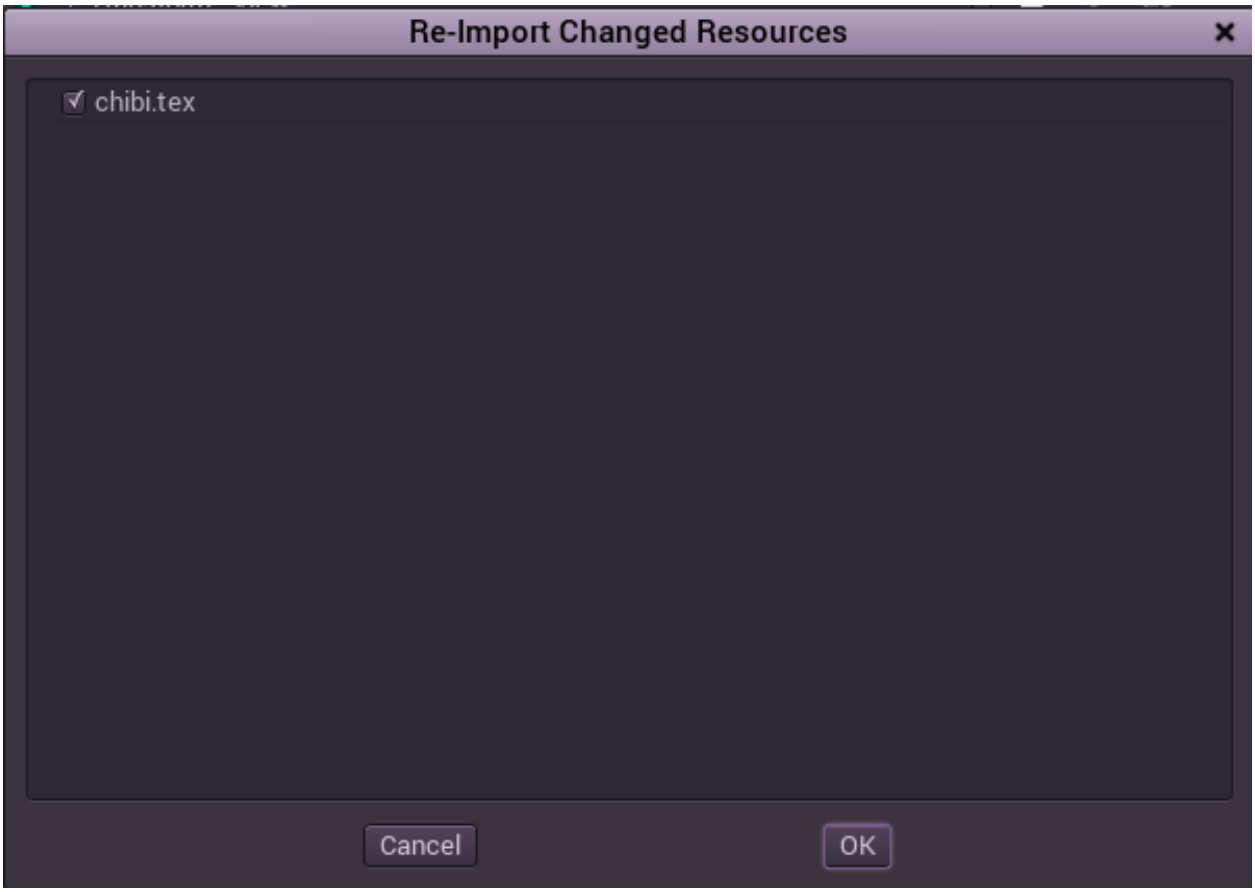
More information about each specific type of asset can be found in specific sections, such as Importing Textures.

Tracking changes and re-importing

Godot tracks changes in the source assets constantly. If at least one asset has been found to be modified (md5 is different than when it was imported), a small red indicator will appear in the top right corner of the screen.

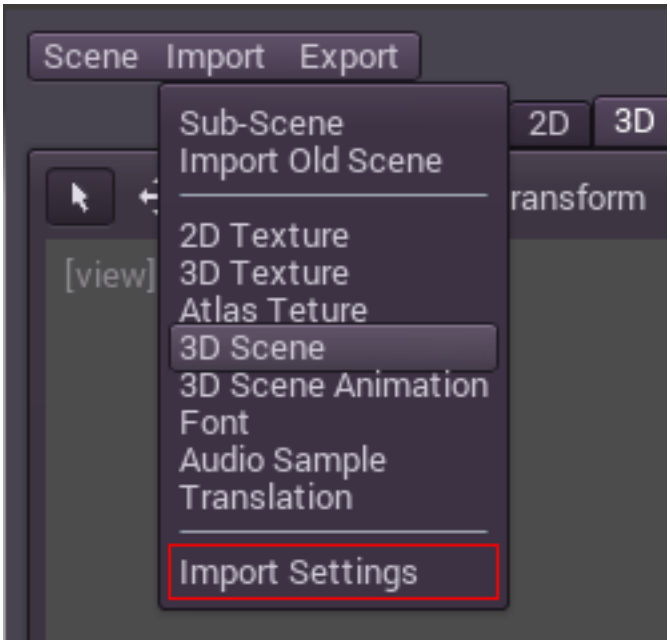
From that moment onward, the user can choose to re-import at any given time by clicking on the red-icon. When this action is done, a dialog will pop-up showing which resources can be re-imported (all selected by default).

Accepting that dialog will immediately re-import the resources and will update any of them currently in use in the editor (like a texture, model or audio file).

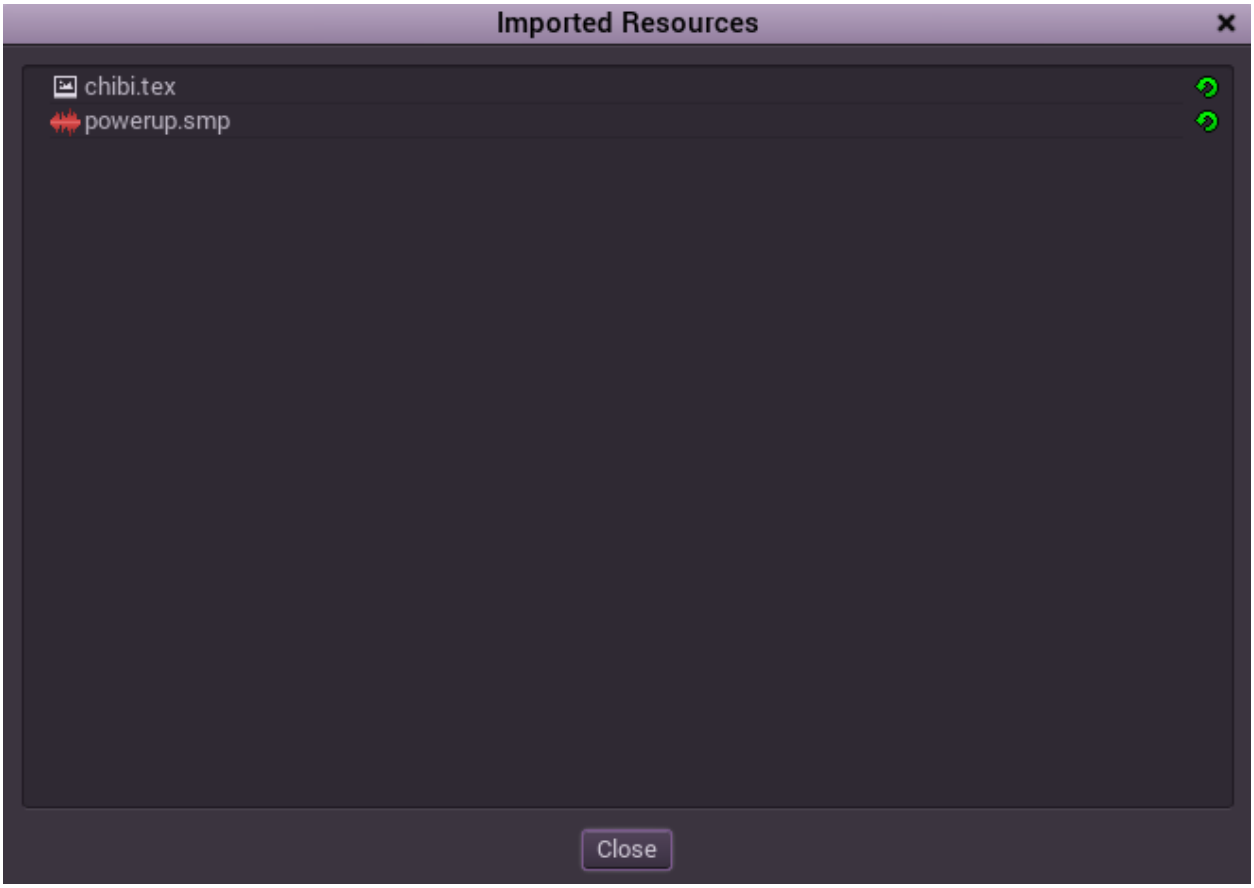


Manually re-importing

The re-import process is automatic, but it may be desired at some point to change the settings of an already imported file, so it can be re-imported differently. For this, the Import Settings window is provided.



This screen allows the user to re-open the corresponding import-window to re-import that asset again, with the ability to change any of the settings.



8.2.2 Importing textures

Do NOT import them in most cases

In most cases you **don't** want images imported when dealing with 2D and GUI. Just copy them to the filesystem. Read the tutorial on *Administrando archivos de imagen* before continuing! For 3D, textures are always imported by the 3D scene importer, so importing those is only useful when importing a texture used for 3D that doesn't come with the 3D scene (for example, in a shader). The flags and options are the same as here, so reading the rest of the document might help too.

OK, you *might* want to import them

So, if you have read the previous tutorial on the texture exporter, the texture importer gives you more fine-grained control on how textures are imported. If you want to change flags such as repeat, filter, mipmaps, fix edges, etc. ***PER texture***, importing them is the best way to accomplish this (since you can't save such flags in a standard image file).

Lack of MipMaps

Images in 3D hardware are scaled with a (bi)linear filter, but this method has limitations. When images are shrunk too much, two problems arise:

- **Aliasing**: Pixels are skipped too much, and the image shows discontinuities. This decreases quality.
- **Cache Misses**: Pixels being read are too far apart, so texture cache reads a lot more data than it should. This decreases performance.

To solve this, mipmaps are created. Mipmaps are versions of the image shrunk by half in both axis, recursively, until the image is 1 pixel of size. When the 3D hardware needs to shrink the image, it finds the largest mipmap it can scale from, and scales from there. This improves performance and image quality.

Godot automatically creates mipmaps upon load for standard image files. This process is time consuming (although not much) and makes load times a little worse. Pre-importing the textures allows the automatic generation of mipmaps.

Unwanted MipMaps

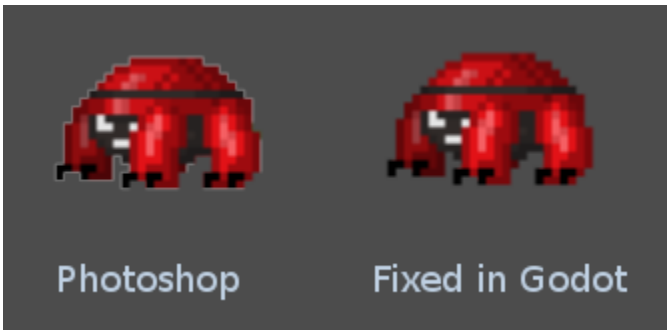
Remember the previous point about mipmaps? Yes, they are cool, but mobile GPUs only support them if the textures are in power of 2 dimensions (i.e. 256x256 or 512x128). In these platforms, Godot will stretch and enlarge the texture to the closest power of 2 size and then generate the mipmaps. This process takes more of a performance hit and it might degrade the quality a little more.

Because of this, there are some scenarios when it may be desirable to not use them, and just use a linear filter. One of them is when working with graphical user interfaces (GUIs). Usually they are made of large images and don't stretch much. Even if the screen resolution is in a larger or smaller value than original art, the amount of stretch is not as much and the art can retain the quality. Pre-importing the textures also allows the disabling of mipmap generation.

Blending artifacts

The blending equation used by applications like Photoshop is too complex for realtime. There are better approximations such as pre-multiplied alpha, but they impose more stress in the asset pipeline. In the end, we are left with textures that have artifacts in the edges, because apps such as Photoshop store white pixels in completely transparent areas. Such white pixels end up showing thanks to the texture filter.

Godot has an option to fix the edges of the image (by painting invisible pixels the same color as the visible neighbours):



However, this must be done every time the image changes. Pre-Importing the textures makes sure that every time the original file changes, this artifact is fixed upon automatic re-import.

Texture flags

Textures have flags. The user can choose for them to repeat or clamp to edges (when UVs exceed the 0,0,1,1 boundary). The magnifying filter can also be turned off (for a Minecraft-like effect). Such values can not be edited in standard file formats (png, jpg, etc.), but can be edited and saved in Godot .tex files. Then again, the user may not want to change the values every time the texture changes. Pre-Importing the textures also takes care of that.

Texture compression

Aside from the typical texture compression, which saves space on disk (.png, jpg, etc.), there are also texture compression formats that save space in memory (more specifically video memory. This allows to have much better looking textures in games without running out of memory, and decrease memory bandwidth when reading them so they are a big plus.

There are several video texture compression formats, none of which are standard. Apple uses PVRTC. PC GPUs, consoles and nVidia Android devices use S3TC (BC), other chipsets use other formats. OpenGL ES 3.0 standardized on ETC format, but we are still a few years away from that working everywhere.

Still, when using this option, Godot converts and compresses to the relevant format depending on the target platform (as long as the user pre-imported the texture and specified video ram compression!).

This kind of compression is often not desirable for many types of 2D games and UIs because it is lossy, creating visual artifacts. This is especially noticeable on games that use the trendy vectory social game artwork. However, the fact that it saves space and improves performance may make up for it.

The 3D scene importer always imports textures with this option turned on.

Atlases

Remember how mobile GPUs have this limitation of textures having to be in power of 2 sizes to be able to generate mimpmaps for optimum stretching? What if we have a lot of images in different random sizes? All will have to be scaled and mipmapped when loaded (using more CPU and memory) or when imported (taking more storage space). This is probably still OK, but there is a tool that can help improve this situation.

Atlases are big textures that fit a lot of small textures inside efficiently. Godot supports creating atlases in the importer, and the imported files are just small resources that reference a region of the bigger texture.

Atlases can be a nice solution to save some space on GUI or 2D artwork by packing everything together. The current importer is not as useful for 3D though (3D Atlases are created differently, and not all 3D models can use them).

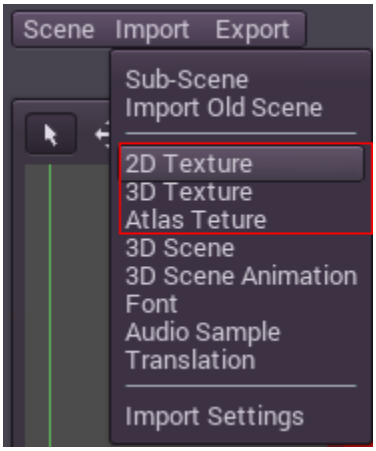
As a small plus, atlases can decrease the amount of "state changes" when drawing. If a lot of objects that are drawn using several different textures are converted to an atlas, then the texture rebinds per object will go from dozens or hundreds to one. This will give the performance a small boost.

Artists use PSD

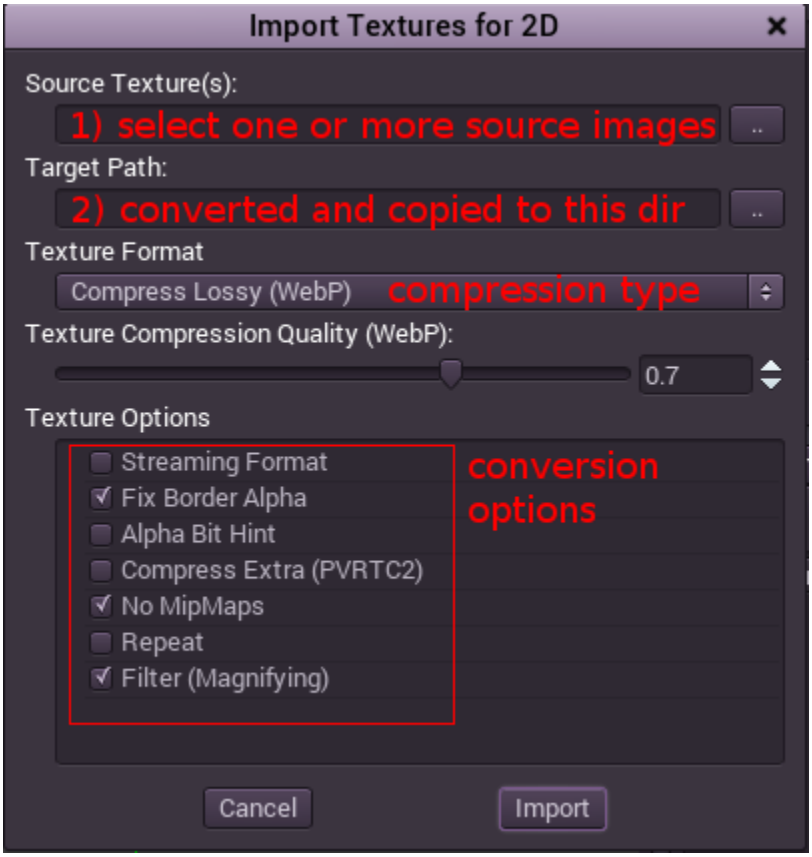
Still wondering whether to use the texture importer or not? Remember that in the end, artists will often use Photoshop anyway, so it may be wiser to just let the import subsystem to take care of importing and converting the PSD files instead of asking the artist to save a png and copy it to the project every time.

Texture importer

Finally! It's time to take a look at the texture importer. There are 3 options in the import menu. They are pretty much (almost) the same dialog with a different set of defaults.



When selected, the texture import dialog will appear. This is the default one for 2D textures:



Each import option has a function, explained as follows:

Source texture(s)

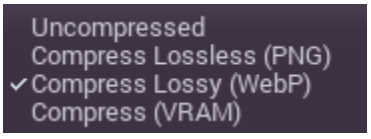
One or more source images can be selected from the same folder (this importer can do batch-conversion). This can be from inside or outside the project.

Target path

A destination folder must be provided. It must be inside the project, as textures will be converted and saved to it. Extensions will be changed to .tex (Godot resource file for textures), but names will be kept.

Texture format

This combo allows to change the texture format (compression in this case):



Each of the four options described in this table together with their advantages and disadvantages (= Best, =Worst):

	Uncompressed	Compress Lossless (PNG)	Compress Lossy (WebP)	Compress VRAM
Description	Stored as raw pixels	Stored as PNG	Stored as WebP	Stored as S3TC/BC,PVRTC/ETC, depending on platform
Size on Disk	Large	Small	Very Small	Small
Memory Usage	Large	Large	Large	Small
Performance	Normal	Normal	Normal	Fast
Quality Loss	None	None	Slight	Moderate
Load Time	Normal	Slow	Slow	Fast

Texture options

Provided are a small amount of options for fine grained import control:

- **Streaming Format** - This does nothing as of yet, but a texture format for streaming different mipmap levels is planned. Big engines have support for this.
- **Fix Border Alpha** - This will fix texture borders to avoid the white auras created by white invisible pixels (see the rant above).
- **Alpha Bit Hint** - Godot auto-detects if the texture needs alpha bit support for transparency (instead of full range), which is useful for compressed formats such as BC. This forces alpha to be 0 or 1.
- **Compress Extra** - Some VRAM compressions have alternate formats that compress more at the expense of quality (PVRTC2 for example). If this is ticked, texture will be smaller but look worse.
- **No MipMaps** - Force imported texture to NOT use mipmaps. This may be desirable in some cases for 2D (as explained in the rant above), though it's NEVER desirable for 3D.
- **Repeat** - Texture will repeat when UV coordinates go beyond 1 and below 0. This is often desirable in 3D, but may generate artifacts in 2D.
- **Filter** - Enables linear filtering when a texture texel is larger than a screen pixel. This is usually turned on, unless it's required for artistic purposes (Minecraft look, for example).

8.2.3 Importing fonts

What is a font?

Fonts in modern operating systems are created as scalable vector graphics. They are stored as a collection of curves (usually one for each character), which are independent of the screen resolution, and stored in standardized file formats, such as TTF (TrueType) or OTF (OpenType).

Rendering such fonts to bitmaps is a complex process, which employs different methods to convert curves to pixels depending on context and target size. Due to this, this rendering process must be done by using the CPU. Game engines use the GPU to render, and 3D APIs don't really support the means to do this efficiently, so fonts have to be converted to a format that is friendly to the GPU when imported to a project.

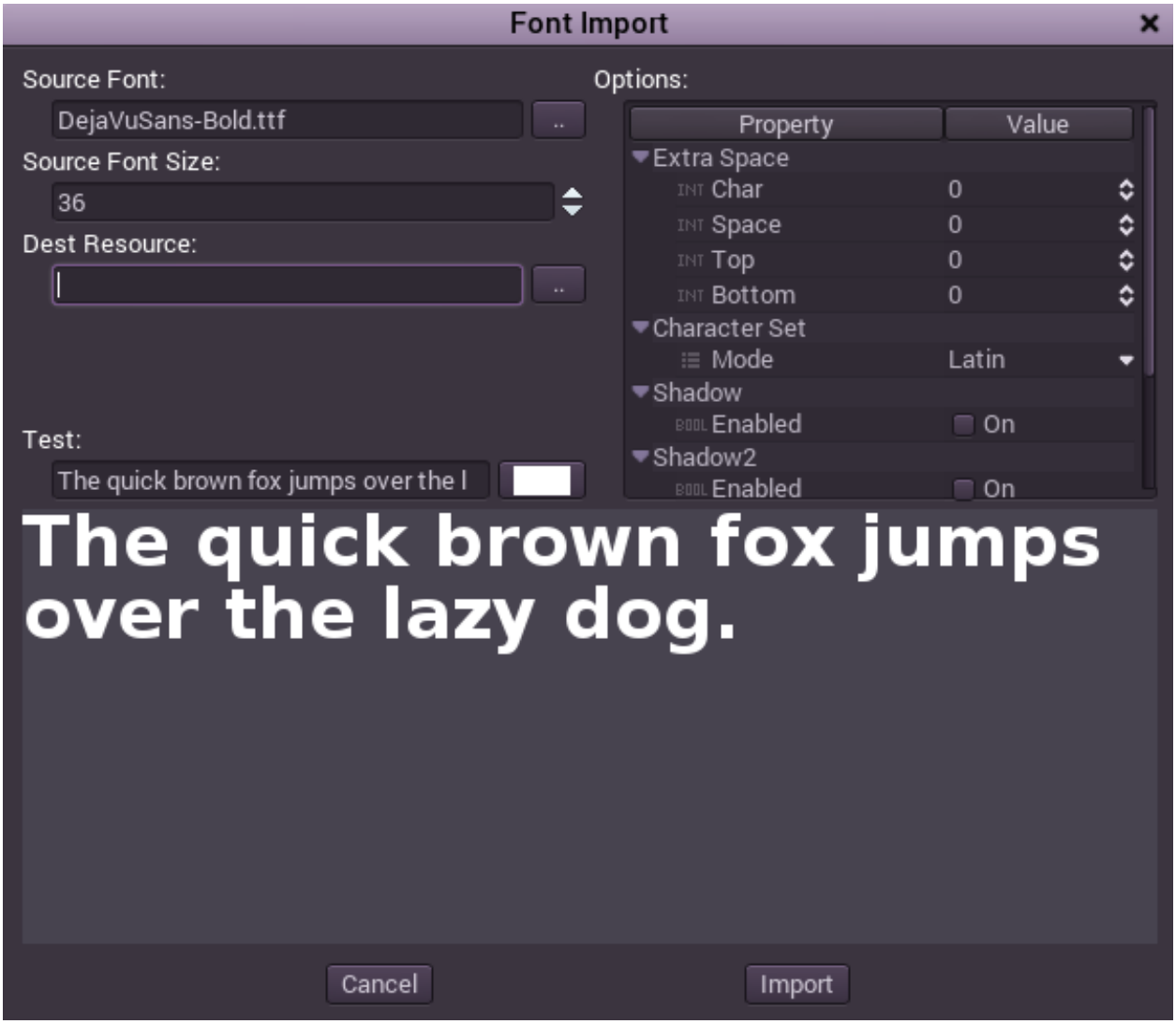
Converting fonts

This conversion process consists of rendering a vector font to a given point size and storing all the resulting characters in a bitmap texture. The bitmap texture is then used by the GPU to draw a small quad for each character and form readable strings.

The drawback of this process is that fonts must be pre-imported in the specific sizes that they will use in the project. However, given that that bitmap fonts compress really well, this is not as bad as it sounds.

Importing a font

Fonts are imported via the Font import dialog. The dialog will ask for a font, a size, some options and a target resource file to save.



The dialog is fully dynamic, which means that any change will be reflected in the font preview window. The user can tweak almost every parameter and get instant feedback on how the font will look.

Since the resulting font is a bitmap, a few more options were added to make the imported font look even nicer. These options were added to please graphic designers, who love putting gradients, outlines and shadows in fonts, as well as changing all the inter-spaces available :). These options will be explained in the next section.

Extra spacing

It is possible to add more space for:

- **Characters**, the space between them can be varied.
- **"space" character**, so the distance between words is bigger.
- **Top and Bottom margins**, this changes the spacing between lines as well as the space between the top and bottom lines and the borders.

Shadows & outline

Fonts can have a shadow. For this, the font is drawn again, below the original, in a different color, and then blurred with a Gaussian kernel of different sizes. The resulting shadow can be adjusted with an exponential function to make it softer or more like an outline. A second shadow is also provided to create some added effects, like a bump or outline+shadow.

Gradients

Gradients are also another of the visual effects that graphic designers often use. To show how much we love them, we added those too. Gradients can be provided as a simple curve between two colors, or a special png file with a hand drawn gradient.

Internationalization

Colors, shadows and gradients are beautiful, but it's time we get to serious business. Developing games for Asian markets is a common practice in today's globalized world and app stores.

Here's when things get tricky with using bitmap fonts. Asian alphabets (Chinese, Japanese and Korean) contain dozens of thousands of characters. Generating bitmap fonts with every single of them is pretty expensive, as the resulting textures are huge. If the font size is small enough, it can be done without much trouble, but when the fonts become bigger, we run out of video ram pretty quickly!

To solve this, Godot allows the user to specify a text file (in UTF-8 format) where it expects to find all the characters that will be used in the project. This seems difficult to provide at first, and more to keep up to date, but it becomes rather easy when one realizes that the .csv with the translations can be used as such source file (see the *Importing translations* section). As Godot re-imports assets when their dependencies change, both the translation and font files will be updated and re-imported automatically if the translation csv changes.

Another cool trick for using a text file as limit of which characters can be imported is when using really large fonts. For example, the user might want to use a super large font, but only to show numbers. For this, he or she writes a numbers.txt file that contains "1234567890", and Godot will only limit itself to import data, thus saving a lot of video memory.

8.2.4 Importing audio samples

Why importing?

Importing Audio Samples into the game engine is a process that should be easier than it really is. Most readers are probably thinking "Why not just copy the wav files to a folder inside the project and be over with it?"

It's not usually that simple. Most game engines use uncompressed audio (in memory, at least) for sound effects. The reason for this is because it's really cheap to play back and resample. Compressed streamed audio (such as ogg files) takes a large amount of processor to decode so no more than one or two are streamed simultaneously. However, with sound effects, one expects a dozen of them to be playing at the same time in several situations.

Because of this, sound effects are loaded uncompressed into memory, and here is where the problems begin.

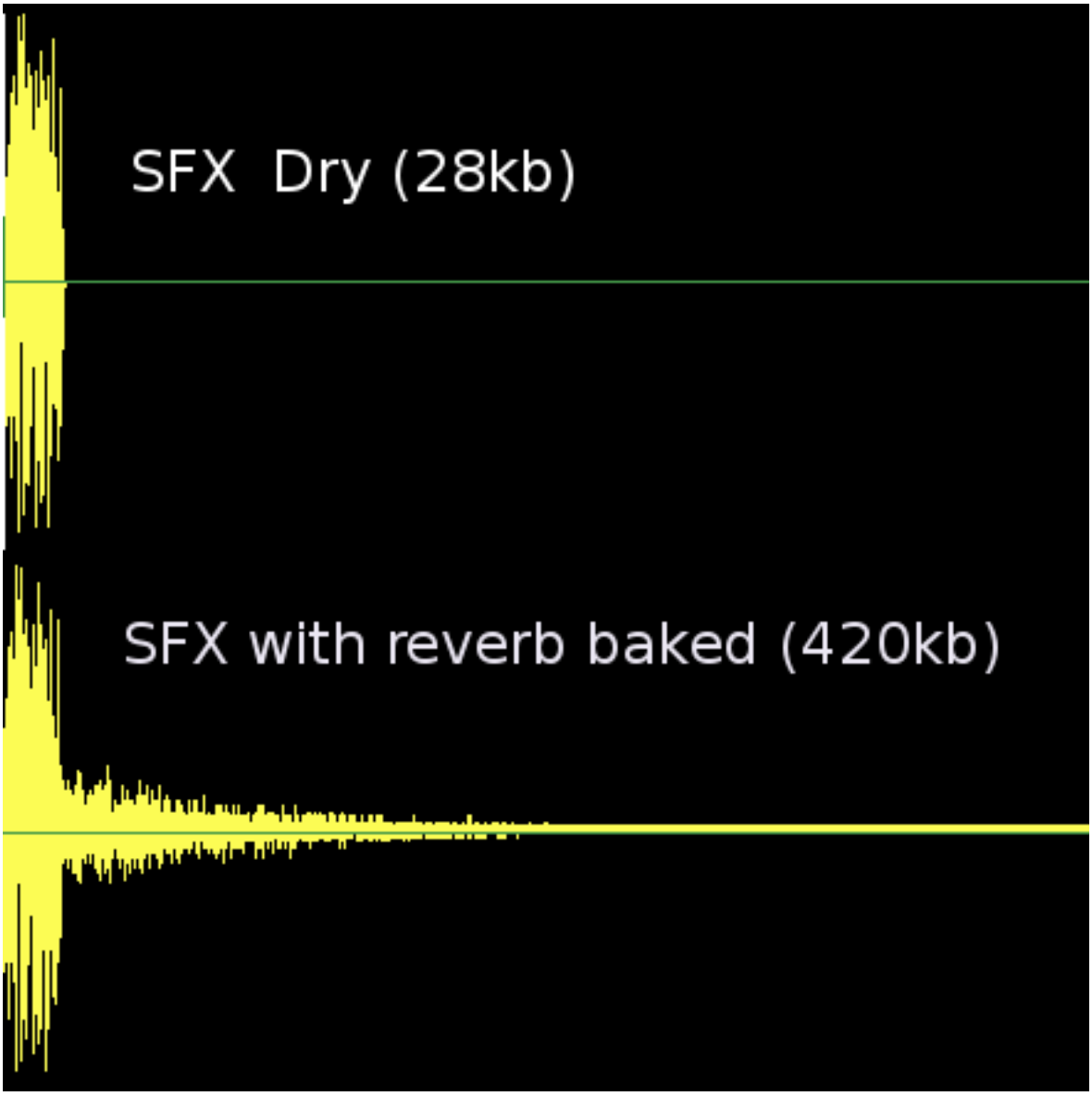
As is usual with graphics, the situation where programmers don't really know about audio and audio engineers don't know about programming is also common in the industry. This leads to a scenario where a project ends up wasting resources unnecessarily.

To be more precise, SFX artists tend to work with audio formats that give them a lot of room for tweaking the audio with a low noise floor and minimum aliasing, such as 96kHz, 24 bits. In many cases, they work in stereo too. Added to that, many times they add effects with an infinite or really long fadeout, such as reverb, which leads to apparent trailing silences. Finally, many DAWs also add silence at the beginning when normalizing to wav.

These often result in extremely large files to integration into a game engine with sound effects taking dozens of megabytes.

How much does quality matter?

First of all, it is important to know that Godot has an internal reverb generator. Sound effects can go to four different setups (small, medium and large room, as well as hall), with different send amounts. This saves SFX artists the need to add reverb to the sound effects, reducing their size greatly and ensuring correct trimming. Say no to SFX with baked reverb!



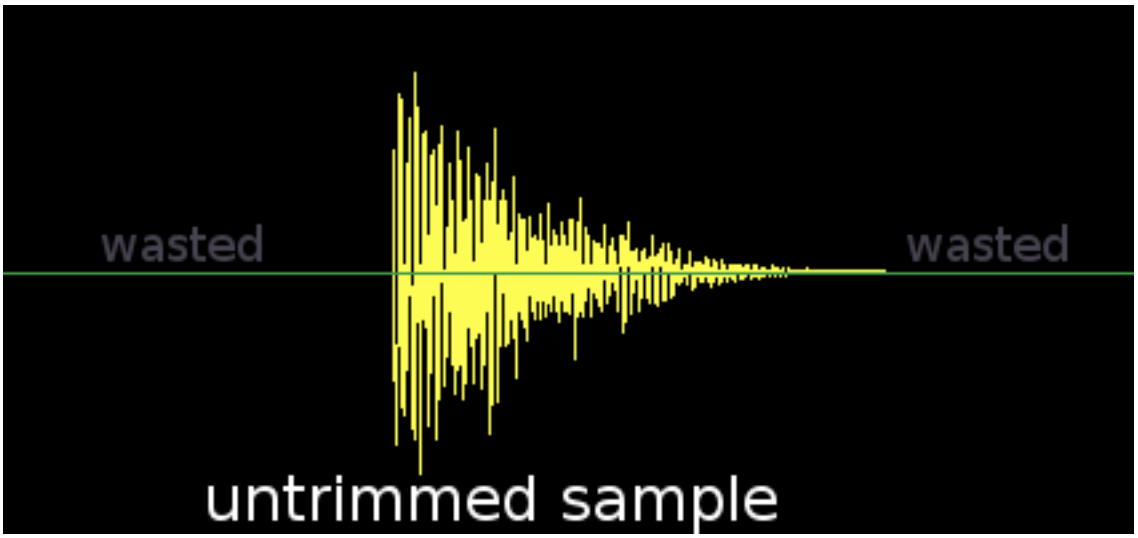
Another common problem is that, while it's useful for working inside a DAW, high bit depths (24 bits) and high sampling rate (96kHz) are completely unnecessary for use in a game, as there is no audible difference. If positional sound is going to be used (for 2D and 3D), the panning and stereo reverb will be provided by the engine, so there is little need for stereo sound. How does this affect the resource usage? Look at the following comparison:

Format	1 Second of Audio	Frame Size
24 bits, 96 kHz, Stereo	576kb	12
16 bits, 44 kHz, Mono	88kb	2
16 bits, IMA-ADPCM	22kb	1/2

As seen, for being no audible difference, the 16 bits, 44kHz, mono conversion takes *6 times less memory* than the 24 bits, 96kHz, Stereo version. The IMA-ADPCM version (using computationally-light audio compression) takes *24 times less memory* than what was exported from the DAW.

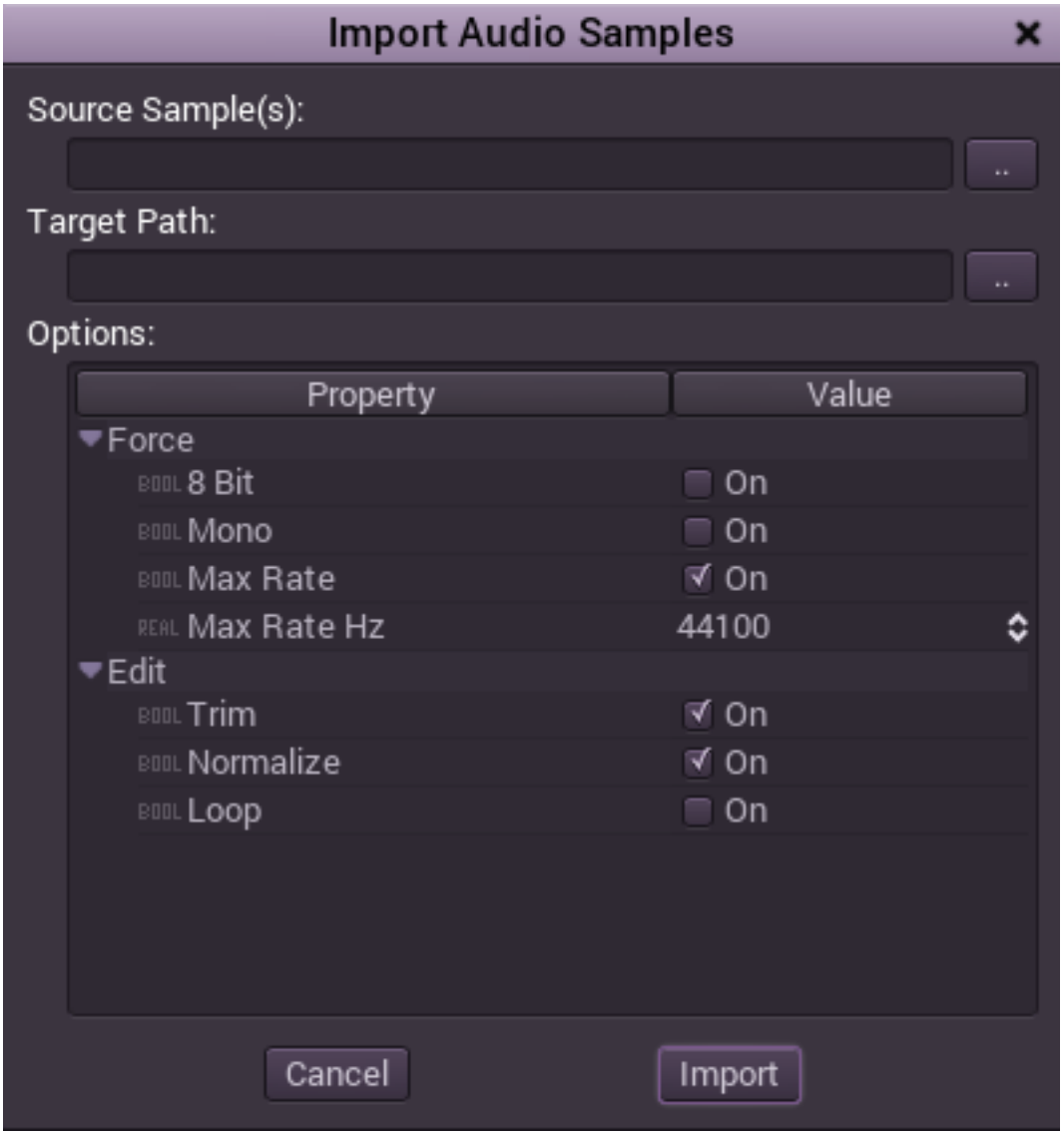
Trimming

One last issue that happens often is that the waveform files received have silences at the beginning and at the end. These are inserted by DAWs when saving to a waveform, increase their size unnecessarily and add latency to the moment they are played back. Trimming them solves this, but it takes effort for the SFX artist, as they have to do it in a separate application. In the worst case, they may not even know the silences are being added.



Importing audio samples

Godot has a simple screen for importing audio samples to the engine. SFX artists only have to save the wav files to a folder outside the project, and the import dialog will fix the files for inclusion, as well as doing it automatically every time they are modified and re-imported.



In this screen, the quality of the audio can be limited to what is needed, and trimming is done automatically. In addition, several samples can be loaded and batch-converted, just as textures can.

Looping

Godot supports looping in the samples (Tools such as Sound Forge or Audition can add loop points to wav files). This is useful for sound effects such as engines, machine guns, etc. Ping-pong looping is also supported.

As an alternative, the import screen has a "loop" option that enables looping for the entire sample when importing.

8.2.5 Importing translations

Games and internationalization

The world is full of different markets and cultures and, to maximize profits™, nowadays games are released in several languages. To solve this, internationalized text must be supported in any modern game engine.

In regular desktop or mobile applications, internationalized text is usually located in resource files (or .po files for GNU stuff). Games, however, can use several orders of magnitude more text than applications, so they must support efficient methods for dealing with loads of multilingual text.

There are two approaches to generate multilingual language games and applications. Both are based on a key:value system. The first is to use one of the languages as the key (usually English), the second is to use a specific identifier. The first approach is probably easier for development if a game is released first in English, later in other languages, but a complete nightmare if working with many languages at the same time.

In general, games use the second approach and a unique ID is used for each string. This allows to revise the text while it's being translated to others. The unique ID can be a number, a string, or a string with a number (it's just a unique string anyway).

Translators also, most of the time prefer to work with spreadsheets (either as a Microsoft Excel file or a shared Google Spreadsheet).

Translation format

To complete the picture and allow efficient support for translations, Godot has a special importer that can read csv files. Both Microsoft Excel and Google Spreadsheet can export to this format, so the only requirement is that the files have a special arrangement. The csv files must be saved in utf-8 encoding and be formatted as follows:

	<lang1>	<lang2>	<langN>
KEY1	string	string	string
KEY2	string	string	string
KEYN	string	string	string

The "lang" tags must represent a language, which must be one of the *valid locales* supported by the engine. The "KEY" tags must be unique and represent a string universally (they are usually in uppercase, to differentiate from other strings). Here's an example:

id	en	es	ja
GREET	Hello, friend!	Hola, Amigo!	
ASK	How are you?	Cómo está?	
BYE	Good Bye	Adiós	

Import dialog

The import dialog takes a csv file in the previously described format and generates several compressed translation resource files inside the project.

Selecting a csv file autodetects the languages from the first row and determines which column represents which language. It is possible to change this manually, by selecting the language for each column.

The import dialog also can add the translation to the list of translations to load when the game runs, specified in engine.cfg (or the project properties). Godot allows loading and removing translations at runtime as well.

8.3 Export

8.3.1 Exporting projects

Why exporting?

Originally, Godot did not have any means to export projects. The developers would compile the proper binaries and build the packages for each platform manually.

When more developers (and even non-programmers) started using it, and when our company started taking more projects at the same time, it became evident that this was a bottleneck.

On PC

Distributing a game project on PC with Godot is rather easy. Just drop the godot.exe (or godot) binary together in the same place as the engine.cfg file, zip it and you are done. This can be taken advantage of to make custom installers.

It sounds simple, but there are probably a few reasons why the developer may not want to do this. The first one is that it may not be desirable to distribute loads of files. Some developers may not like curious users peeking at how the game was made, others may just find it inelegant, etc.

Another reason is that, for distribution, the developer might use a specially compiled binary, which is smaller in size, more optimized and does not include tools inside (like the editor, debugger, etc.).

Finally, Godot has a simple but efficient system for creating DLCs as extra package files.

On mobile

The same scenario in mobile is a little worse. To distribute a project in those devices, a binary for each of those platforms is built, then added to a native project together with the game data.

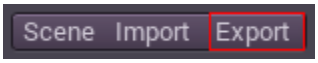
This can be troublesome because it means that the developer must be familiarized with the SDK of each platform before even being able to export. In other words, while learning each SDK is always encouraged, it can be frustrating to be forced to do it at an undesired time.

There is also another problem with this approach, which is the fact that different devices prefer some data in different formats to run. The main example of this is texture compression. All PC hardware uses S3TC (BC) compression and that has been standardized for more than a decade, but mobile devices use different formats for texture compression, such as PVRCT (iOS) or ETC (Android).

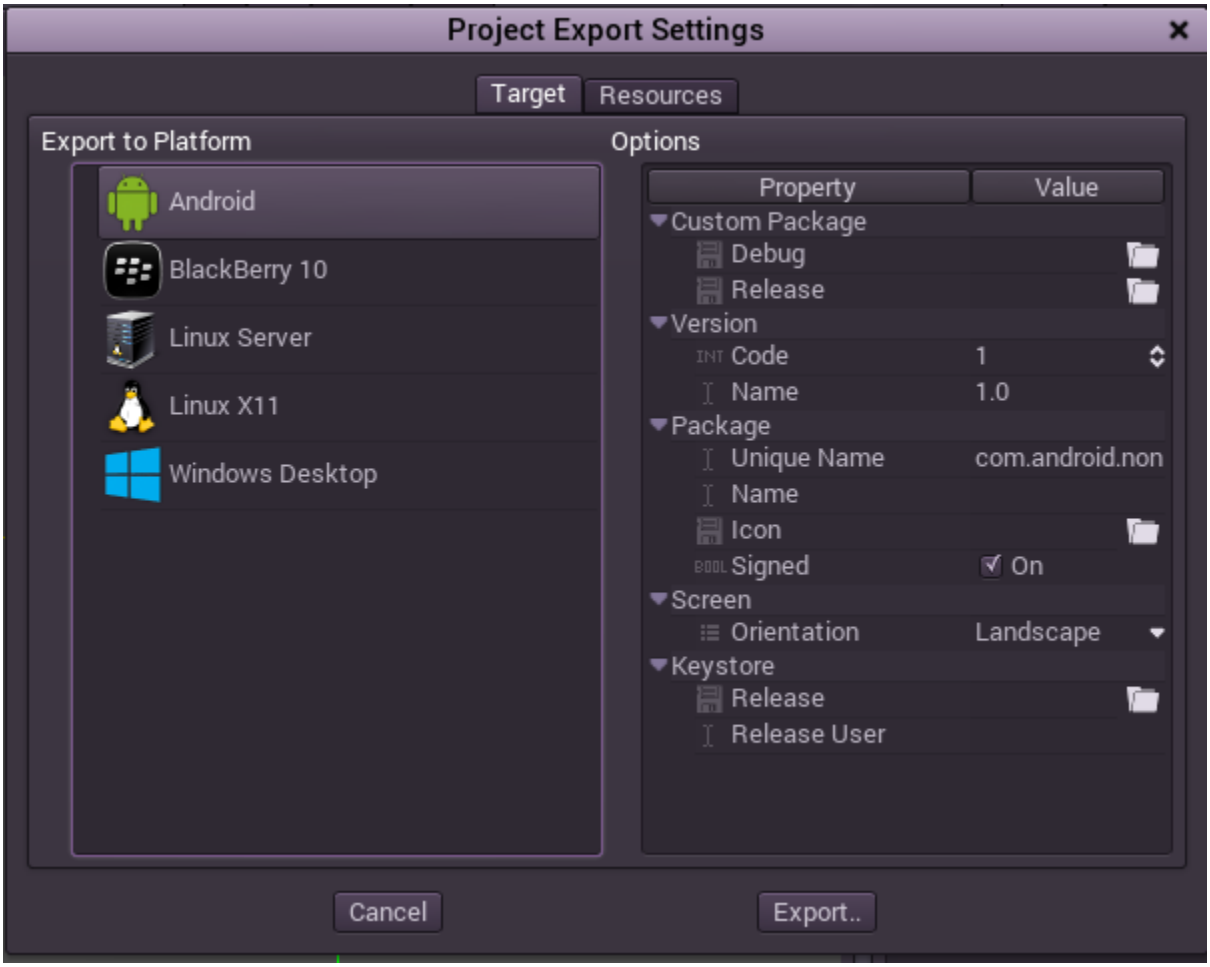
Export dialog

After many attempts at different export workflows, the current one has proven to work the best. At the time of this writing, not all platforms are supported yet, but the supported platforms continue to grow.

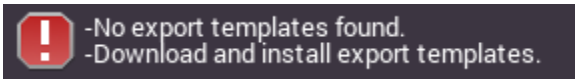
To open the export dialog, just click the "Export" button:



The dialog will open, showing all the supported export platforms:



The default options are often enough to export, so tweaking them is not necessary, but provide extra control. However, many platforms require additional tools (SDKs) to be installed to be able to export. Additionally, Godot needs exports templates installed to create packages. The export dialog will complain when something is missing and will not allow the user to export for that platform until they resolve it:

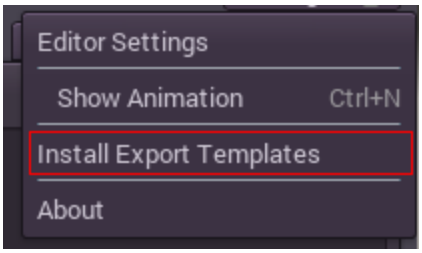


At that time, the user is expected to come back to the documentation and follow instructions on how to properly set up that platform.

Export templates

Apart from setting up the platform, the export templates must be installed to be able to export projects. They can be obtained as a .tpz (a renamed .zip) file from the download page of the website.

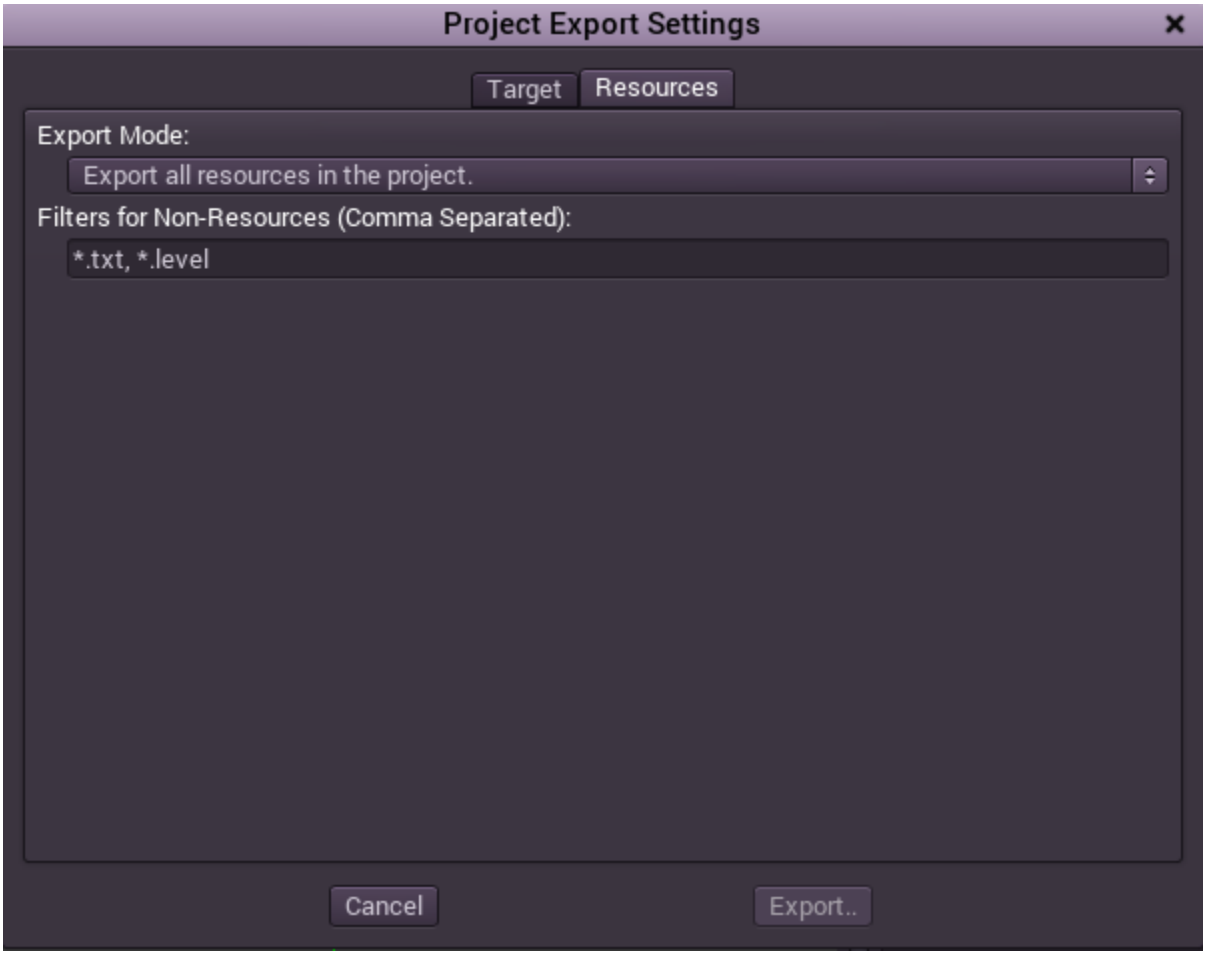
Once downloaded, they can be installed using the "Install Export Templates" option in the editor:



Export mode

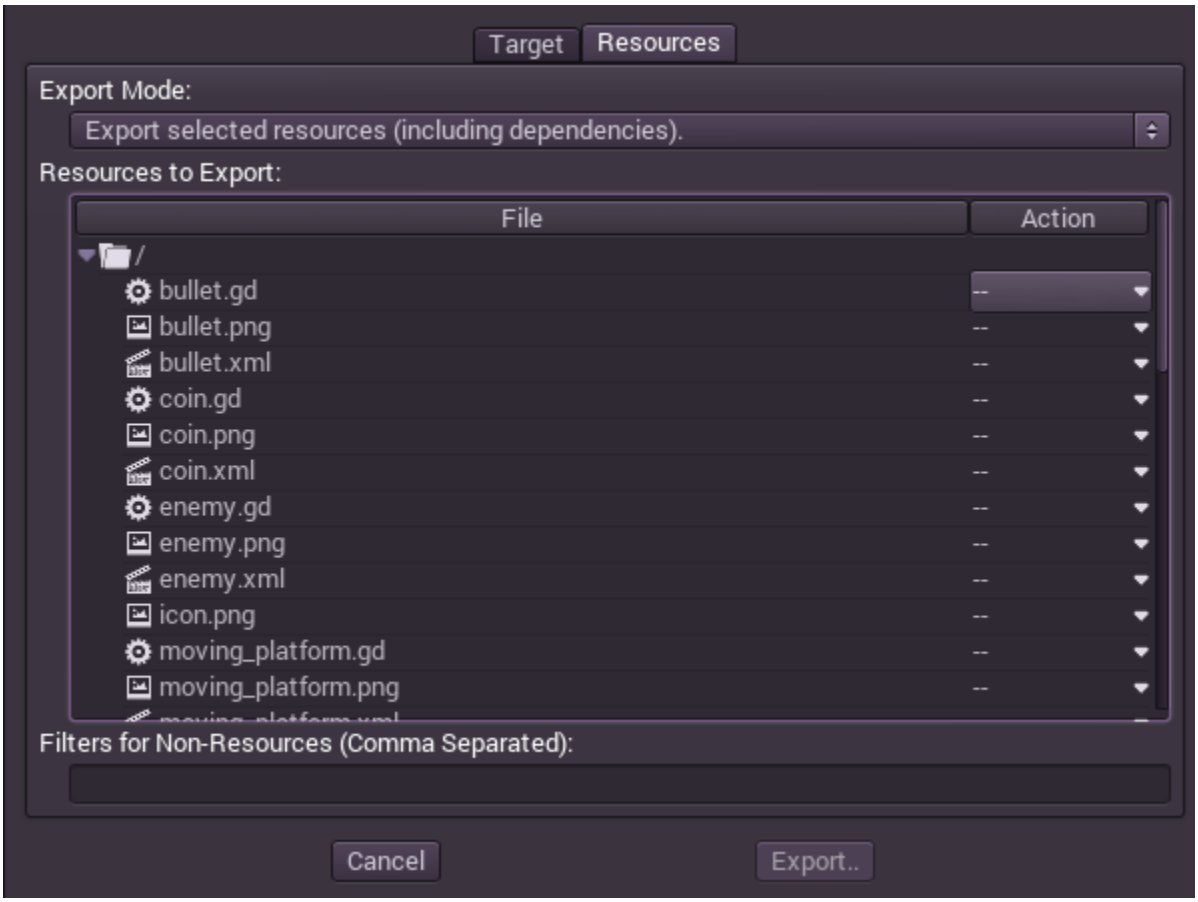
When exporting, Godot makes a list of all the files to export and then creates the package. There are 3 different modes for exporting:

- Export every single file in the project
- Export only resources (+custom filter), this is default.
- Export only selected resources (+custom filter)



- **Export every single file** - This mode exports every single file in the project. This is good to test if something is being forgotten, but developers often have a lot of unrelated stuff around in the dev directory, which makes it a bad idea.

- **Export only resources** - Only resources are exported. For most projects, this is enough. However many developers like to use custom datafiles in their games. To compensate for this, filters can be added for extra extensions (like, *.txt,*.csv, etc.).
- **Export only selected resources** - Only select resources from a list are exported. This is probably overkill for most projects, but in some cases it is justified (usually huge projects). This mode offers total control of what is exported. Individual resources can be selected and dependency detection is performed to ensure that everything needed is added. As a plus, this mode allows to "Bundle" scenes and dependencies into a single file, which is *really* useful for games distributed on optical media.



8.3.2 One-click deploy

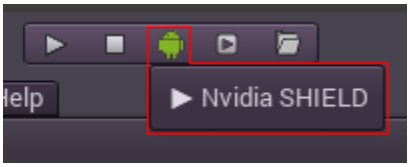
Sounds good, what is it?

This feature will pop up automatically once a platform is properly configured and a supported device is connected to the computer. Since things can go wrong at many levels (platform may not be configured correctly, SDK may incorrectly installed, device may be improperly configured, kitty ate the USB cable, etc.), it's good to let the user know that it exists.

Some platforms (at the time of this writing, only Android and Blackberry 10) can detect when a USB device is connected to the computer, and offer the user to automatically export, install and run the project (in debug mode) on the device. This feature is called, in industry buzz-words, "One Click Deploy" (though, it's technically two clicks...).

Steps for one-click deploy

1. Configure target platform.
2. Configure device (make sure it's in developer mode, likes the computer, usb is recognized, usb cable is plugged, etc.).
3. Connect the device..
4. And voila!



Click once.. and deploy!

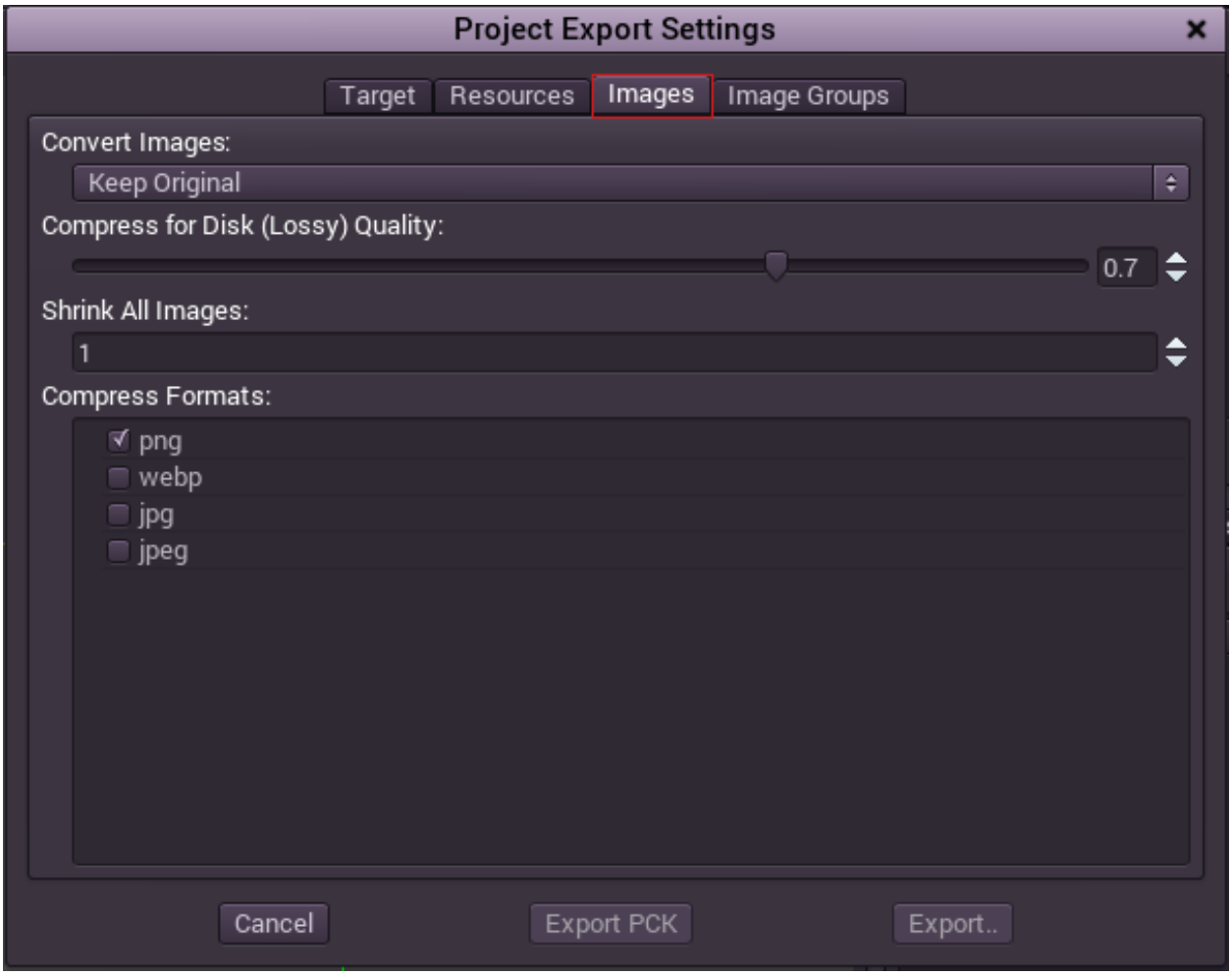
8.3.3 Exporting images

It is often desired to do an operation to all or a group of images upon export. Godot provides some tools for this. Examples of such operations are:

- Converting all images from a lossless format to a lossy one (ie: png -> WebP) for greater compression.
- Shrinking all images to half the size, to create a low resolution build for smaller screens.
- Create an atlas for a group of images and crop them, for higher performance and less memory usage.

Image export options

In the "Project Export Settings" dialog, go to the Images tab:



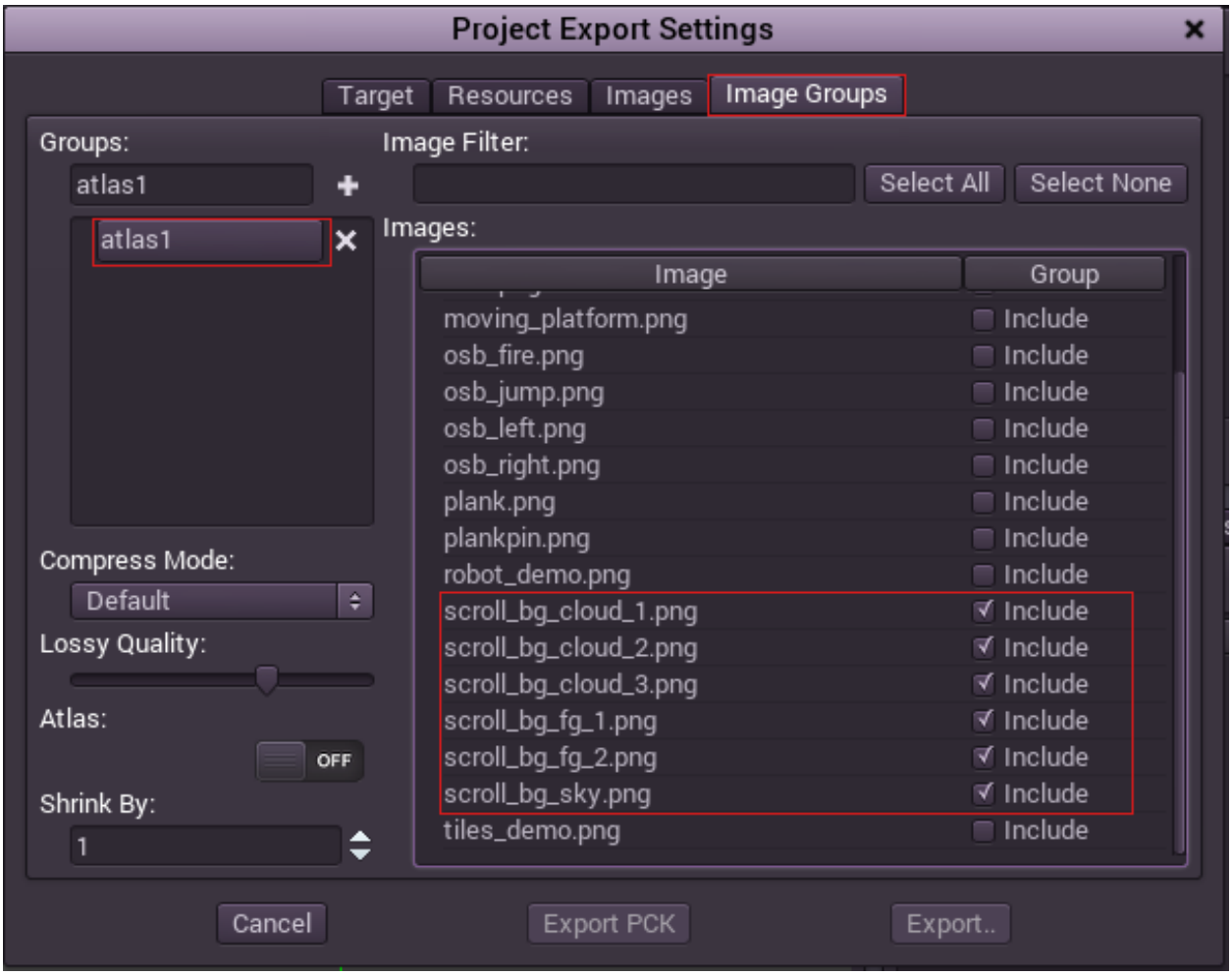
In this dialog the image extensions for conversion can be selected, and operations can be performed that apply to all images (except those in groups, see the next section for those):

- **Convert Image Format**: Probably the most useful operation is to convert to Lossy (WebP) to save disk space. For lossy, a Quality bar can set the quality/vs size ratio.
- **Shrink**: This allows to shrink all images by a given amount. It's useful to export a game to half or less resolution for special devices.
- **Compress Formats**: Allows to select which image exensions to convert.

On export, Godot will perform the desired operation. The first export might be really slow, but subsequent exports will be fast, as the converted images will be cached.

Image group export options

This section is similar to the previous one, except it can operate on a selected group of images. When a image is in a group, the settings from the global export options are overridden by the ones from the group. An image can only be in one group at the same time. So if the image is in another group different to the current one being edited, it will not be selectable.



Atlas

Grouping images allows a texture atlas to be created. When this mode is active, a button to preview the resulting atlas becomes available. Make sure that atlases don't become too big, as some hardware will not support textures bigger than 2048x2048 pixels. If this happens, just create another atlas.

The atlas can be useful to speed up drawing of some scenes, as state changes are minimized when drawing from it (through unlike other engines, Godot is designed so state changes do not affect it as much). Textures added to an atlas get cropped (empty spaces around the image are removed), so this is another reason to use them (save space). If unsure, though, just leave that option disabled.

8.3.4 Exporting for PC

The simplest way to distribute a game for PC is to copy the executables (godot.exe on windows, godot on the rest), zip the folder and send it to someone else. However, this is often not desired.

Godot offers a more elegant approach for PC distribution when using the export system. When exporting for PC (Linux, Windows, Mac), the exporter takes all the project files and creates a "data.pck" file. This file is bundled with a specially optimized binary that is smaller, faster and lacks tools and debugger.

Optionally, the files can be bundled inside the executable, though this does not always works properly.

8.3.5 Exporting for Android

Exporting for Android has fewer requirements than compiling Godot for it. The following steps detail what is needed to setup the SDK and the engine.

Download the Android SDK

Download and install the Android SDK from http://developer.android.com/sdk/index.html

Install OpenJDK or Oracle JDK

Download and install OpenJDK or Oracle JDK. Version 6 and 8 are known to work, some users have reported issues with the jarsigner (used to sign the APKs) in JDK 7.

Create a debug.keystore

Android needs a debug keystore file to install to devices and distribute non-release APKs. If you have used the SDK before and have built projects, ant or eclipse probably generated one for you (In Linux and OSX, you can find it in the ~/.android folder).

If you can't find it or need to generate one, the keytool command from the JDK can be used for this purpose:

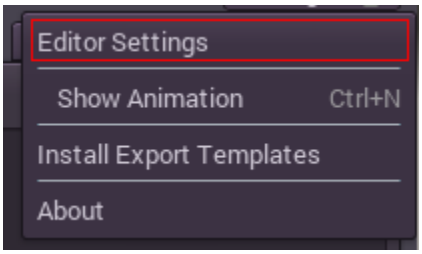
```
keytool -keyalg RSA -genkeypair -alias androiddebugkey -keypass android -keystore␣
↪debug.keystore -storepass android -dname "CN=Android Debug,O=Android,C=US" -
↪validity 9999
```

Make sure you have adb

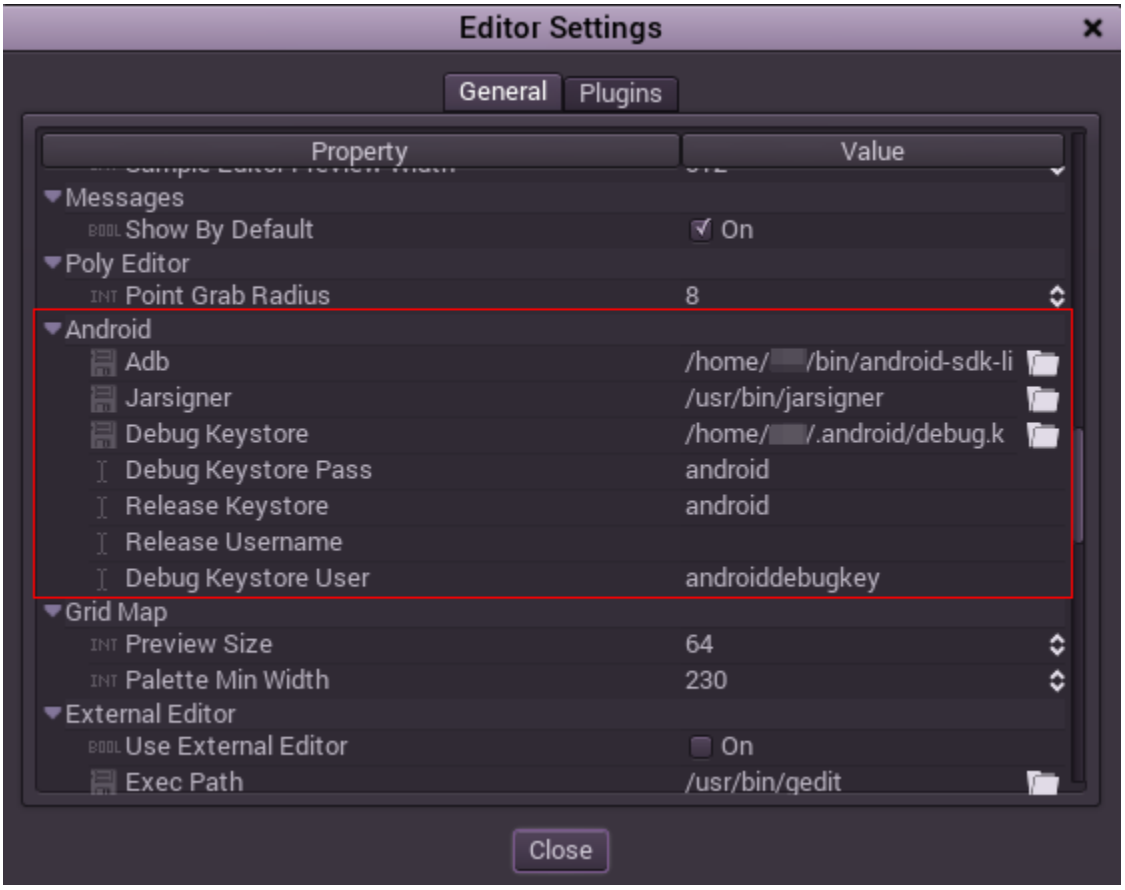
Android Debug Bridge (adb) is the command line tool used to communicate with Android devices. It's installed with the SDK, but you may need to install one (any) of the Android API levels for it to be installed in the SDK directory.

Setting it up in Godot

Enter the Editor Settings screen. This screens contains the editor settings for the user account in the computer (It's independent from the project).



Scroll down to the section where the Android settings are located:



In that screen, the path to 3 files needs to be set:

- The *adb* executable (adb.exe on Windows)
- The *jarsigner* executable (from JDK 6 or 8)
- The debug *keystore*

Once that is configured, everything is ready to export to Android!

8.3.6 Exporting for iOS

Exporting for iOS is done manually at the moment. These are the steps to load your game in an XCode project, where you can deploy to a device, publish, etc.

Requirements

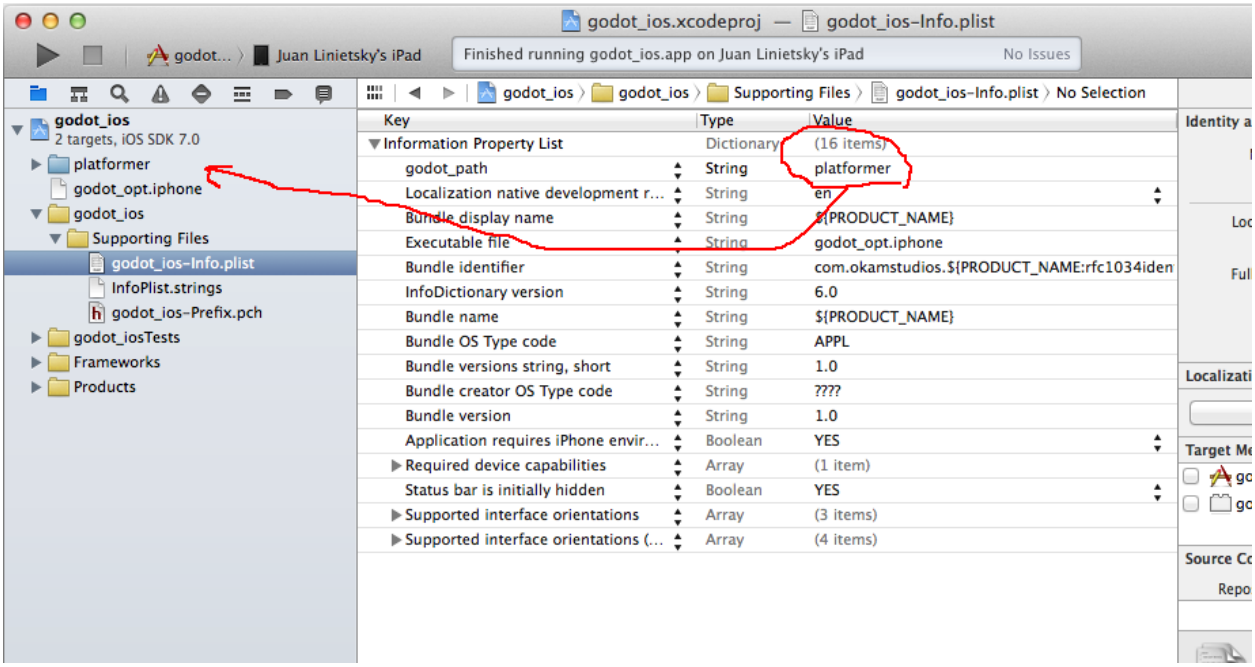
- Download XCode for iOS
- Download the export templates: https://godotengine.org/download
- Since there is no automatic deployer yet, unzip export_templates.tpz manually and extract GodotiOSXCode.zip from it.

The zip contains an XCode project, godot_ios.xcodeproj, an empty data.pck file and the engine executable. Open the project, and modify the game name, icon, organization, provisioning signing certificate identities (??), etc.

Add your project data

Using the Godot editor, *Exporting for PC*, to obtain the data.pck file. Replace the empty data.pck in the XCode project with the new one, and run/archive.

If you want to test your scenes on the iOS device as you edit them, you can add your game directory to the project (instead of data.pck), and add a property "godot_path" to Info.plist, with the name of your directory as its value.



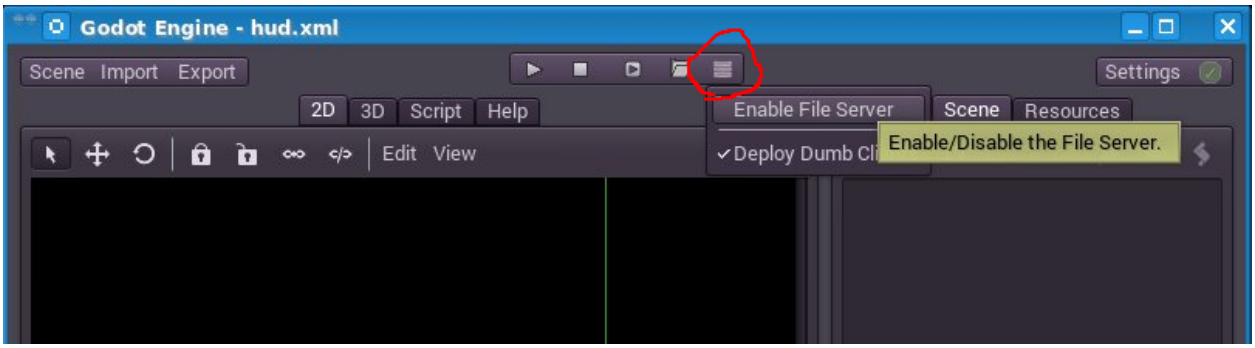
Alternatively you can add all the files from your game directly, with "engine.cfg" at the root.

Loading files from a host

Sometimes your game becomes too big and deploying to the device takes too long every time you run. In that case you can deploy only the engine executable, and serve the game files from your computer.

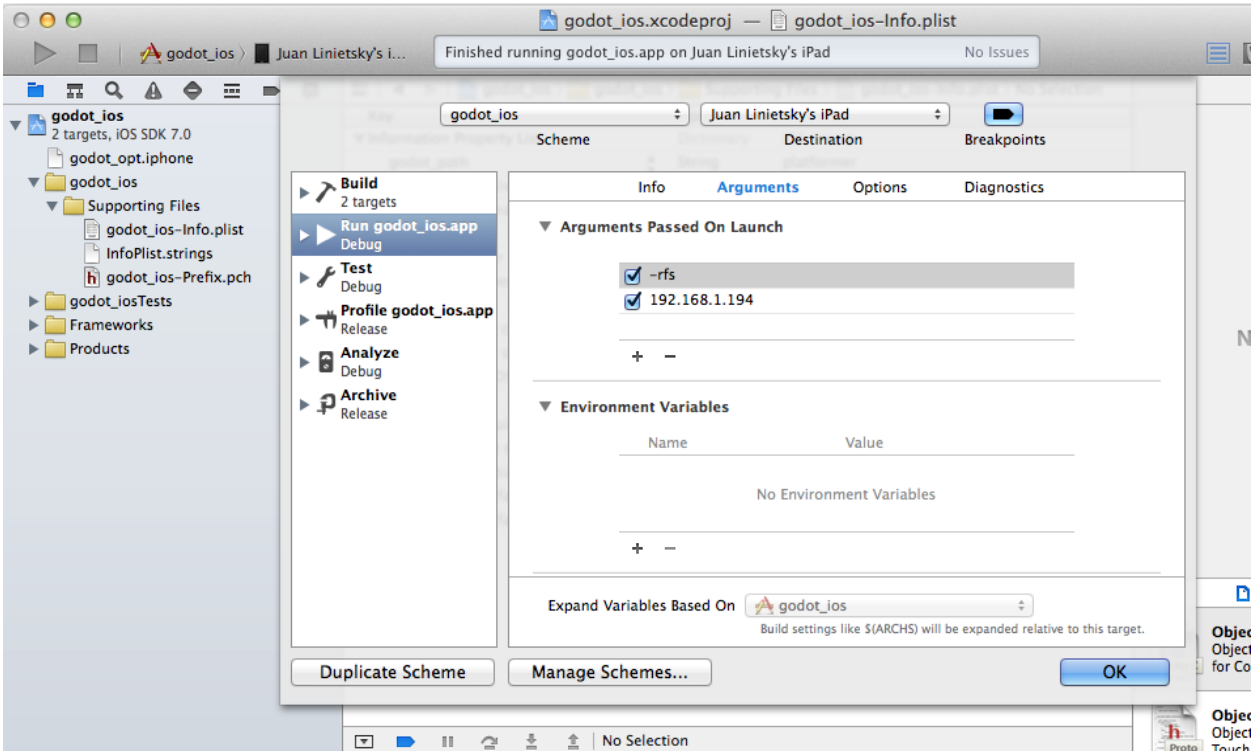
Setting up the file host

On your PC, open the editor, and click the righ-most icon on the top-center group of icons, and select "Enable File Server". The icon turns red. Your PC will open a port and accept connections to serve files from your project's directory (so enable your local firewall accordingly).



Setting up the game

On XCode, click on your app name (top left, next to the "Stop" button), and select "Edit Scheme". Go to the "Arguments" tab, and add 2 arguments, "-rfs" and the IP of your PC.



When you run, your device will connect to the host and open the files remotely. Note that the directory with the game data ("platformer") is no longer added to the project, only the engine executable.

Services for iOS

Special iOS services can be used in Godot. Check out the *Services for iOS* page.

CAPÍTULO 9

Class reference

9.1 @GDScript

Category: Core

9.1.1 Brief Description

Built-in GDScript functions.

9.1.2 Member Functions

float	*sin* (*float* s)
float	*cos* (*float* s)
float	*tan* (*float* s)
float	*sinh* (*float* s)
float	*cosh* (*float* s)
float	*tanh* (*float* s)
float	*asin* (*float* s)
float	*acos* (*float* s)
float	*atan* (*float* s)
float	*atan2* (*float* x, *float* y)
float	*sqrt* (*float* s)
float	*fmod* (*float* x, *float* y)
float	*fposmod* (*float* x, *float* y)
float	*floor* (*float* s)
float	*ceil* (*float* s)
float	*round* (*float* s)
float	*abs* (*float* s)

Continúa en la página siguiente

Tabla 9.1 – proviene de la página anterior

float	*sign* (*float* s)
float	*pow* (*float* x, *float* y)
float	*log* (*float* s)
float	*exp* (*float* s)
float	*is_nan* (*float* s)
float	*is_inf* (*float* s)
float	*ease* (*float* s, *float* curve)
float	*decimals* (*float* step)
float	*stepify* (*float* s, *float* step)
float	*lerp* (*float* from, *float* to, *float* weight)
float	*dectime* (*float* value, *float* amount, *float* step)
Nil	*randomize* ( )
int	*randi* ( )
float	*randf* ( )
float	*rand_range* (*float* from, *float* to)
Nil	*seed* (*float* seed)
Array	*rand_seed* (*float* seed)
float	*deg2rad* (*float* deg)
float	*rad2deg* (*float* rad)
float	*linear2db* (*float* nrg)
float	*db2linear* (*float* db)
float	*max* (*float* a, *float* b)
float	*min* (*float* a, *float* b)
float	*clamp* (*float* val, *float* min, *float* max)
int	*nearest_po2* (*int* val)
WeakRef	*weakref* (*Object* obj)
FuncRef	*funcref* (*Object* instance, *String* funcname)
Object	*convert* (Variant what, *int* type)
int	*typeof* (Variant what)
String	*str* (Variant what, Variant ...)
Nil	*print* (Variant what, Variant ...)
Nil	*printt* (Variant what, Variant ...)
Nil	*prints* (Variant what, Variant ...)
Nil	*printerr* (Variant what, Variant ...)
Nil	*printraw* (Variant what, Variant ...)
String	*var2str* (Variant var)
Variant	*str2var* (*String* string)
RawArray	*var2bytes* (Variant var)
Variant	*bytes2var* (*RawArray* bytes)
Array	*range* (Variant ...)
Resource	*load* (*String* path)
Dictionary	*inst2dict* (*Object* inst)
Object	*dict2inst* (*Dictionary* dict)
int	*hash* (Variant var:Variant)
Color	*Color8* (*int* r8, *int* g8, *int* b8, *int* a8)
Nil	*print_stack* ( )
Object	*instance_from_id* (*int* instance_id)
Resource	*preload* (*String* path)
Nil	*yield* (*Object* object, *String* signal)
Nil	*assert* (*bool* condition)

9.1.3 Numeric Constants

- **PI = 3.141593** — Constant that represents how many times the diameter of a circumference fits around it's perimeter.

9.1.4 Description

This contains the list of built-in gdscript functions. Mostly math functions and other utilities. Everything else is expanded by objects.

9.1.5 Member Function Description

- *float* **sin** (*float* s)

Standard sine function.

- *float* **cos** (*float* s)

Standard cosine function.

- *float* **tan** (*float* s)

Standard tangent function.

- *float* **sinh** (*float* s)

Hyperbolic sine.

- *float* **cosh** (*float* s)

Hyperbolic cosine.

- *float* **tanh** (*float* s)

Hyperbolic tangent.

- *float* **asin** (*float* s)

Arc-sine.

- *float* **acos** (*float* s)

Arc-cosine.

- *float* **atan** (*float* s)

Arc-tangent.

- *float* **atan2** (*float* x, *float* y)

Arc-tangent that takes a 2D vector as argument, returns the full -pi to +pi range.

- *float* **sqrt** (*float* s)

Square root.

- *float* **fmod** (*float* x, *float* y)

Module (remainder of x/y).

- *float* **fposmod** (*float* x, *float* y)

Module (remainder of x/y) that wraps equally in positive and negative.

- *float* **floor** (*float* s)

Floor (rounds down to nearest integer).

- *float* **ceil** (*float* s)

Ceiling (rounds up to nearest integer).

- *float* **round** (*float* s)

Round to nearest integer.

- *float* **abs** (*float* s)

Remove sign (works for integer and float).

- *float* **sign** (*float* s)

Return sign (-1 or +1).

- *float* **pow** (*float* x, *float* y)

Power function, x elevate to y.

- *float* **log** (*float* s)

Natural logarithm.

- *float* **exp** (*float* s)

Exponential logarithm.

- *float* **is_nan** (*float* s)

Return true if the float is not a number.

- *float* **is_inf** (*float* s)

Return true if the float is infinite.

- *float* **ease** (*float* s, *float* curve)

Easing function, based on exponent. 0 is constant, 1 is linear, 0 to 1 is ease-in, 1+ is ease out. Negative values are in-out/out in.

- *float* **decimals** (*float* step)

Return the amount of decimals in the floating point value.

- *float* **stepify** (*float* s, *float* step)

Snap float value to a given step.

- *float* **lerp** (*float* from, *float* to, *float* weight)

Linear interpolates between two values by a normalized value.

- *float* **dectime** (*float* value, *float* amount, *float* step)

Decreases time by a specified amount.

- *Nil* **randomize** ( )

Reset the seed of the random number generator with a new, different one.

- *int* **randi** ( )

Random 32 bits value (integer). To obtain a value from 0 to N, you can use remainder, like (for random from 0 to 19): randi() % 20.

- *float* **randf** ( )

Random value (0 to 1 float).

- *float* **rand_range** (*float* from, *float* to)

Random range, any floating point value between 'from' and 'to'

- *Nil* **seed** (*float* seed)

Set seed for the random number generator.

- *Array* **rand_seed** (*float* seed)

Random from seed, pass a seed and an array with both number and new seed is returned.

- *float* **deg2rad** (*float* deg)

Convert from degrees to radians.

- *float* **rad2deg** (*float* rad)

Convert from radians to degrees.

- *float* **linear2db** (*float* nrg)

Convert from linear energy to decibels (audio).

- *float* **db2linear** (*float* db)

Convert from decibels to linear energy (audio).

- *float* **max** (*float* a, *float* b)

Return the maximum of two values.

- *float* **min** (*float* a, *float* b)

Return the minimum of two values.

- *float* **clamp** (*float* val, *float* min, *float* max)

Clamp both values to a range.

- *int* **nearest_po2** (*int* val)

Return the nearest larger power of 2 for an integer.

- *WeakRef* **weakref** (*Object* obj)

Return a weak reference to an object.

- *FuncRef* **funcref** (*Object* instance, *String* funcname)

Return a reference to the specified function.

- *Object* **convert** (Variant what, *int* type)

Convert from a type to another in the best way possible. The "type" parameter uses the enum TYPE_* in *@Global Scope*.

- *int* **typeof** (Variant what)

Return the internal type of the given Variant object, using the TYPE_* enum in *@Global Scope*.

- *String* **str** (Variant what, Variant ...)

Convert one or more arguments to strings in the best way possible.

- *Nil* **print** (Variant what, Variant ...)

Print one or more arguments to strings in the best way possible to a console line.

- *Nil* **printt** (Variant what, Variant ...)

Print one or more arguments to the console with a tab between each argument.

- *Nil* **prints** (Variant what, Variant ...)

Print one or more arguments to the console with a space between each argument.

- *Nil* **printerr** (Variant what, Variant ...)

Print one or more arguments to strings in the best way possible to standard error line.

- *Nil* **printraw** (Variant what, Variant ...)

Print one or more arguments to strings in the best way possible to console. No newline is added at the end.

- *String* **var2str** (Variant var)

Convert a value to a formatted string that can later be parsed using *str2var*.

- Variant **str2var** (*String* string)

Convert a formatted string that was returned by *var2str* to the original value.

- *RawArray* **var2bytes** (Variant var)

Encode a variable value to a byte array.

- Variant **bytes2var** (*RawArray* bytes)

Decode a byte array back to a value.

- *Array* **range** (Variant ...)

Return an array with the given range. Range can be 1 argument N (0 to N-1), two arguments (initial, final-1) or three arguments (initial, final-1, increment).

- *Resource* **load** (*String* path)

Load a resource from the filesystem, pass a valid path as argument.

- *Dictionary* **inst2dict** (*Object* inst)

Convert a script class instance to a dictionary (useful for serializing).

- *Object* **dict2inst** (*Dictionary* dict)

Convert a previously converted instances to dictionary back into an instance. Useful for deserializing.

- *int* **hash** (Variant var:Variant)

Hash the variable passed and return an integer.

- *Color* **Color8** (*int* r8, *int* g8, *int* b8, *int* a8)

Make a color from red, green, blue and alpha. Arguments can range from 0 to 255.

- *Nil* **print_stack** ( )

Print a stack track at code location, only works when running with debugger turned on.

- *Object* **instance_from_id** (*int* instance_id)

Get an object by its ID.

- *Resource* **preload** (*String* path)

Preload a resource from the filesystem. The resource is loaded during script parsing.

- *Nil* **yield** (*Object* object, *String* signal)

Stop the function execution and return the current state. Call resume on the state to resume execution. This makes the state invalid.

Returns anything that was passed to the resume function call.

If passed an object and a signal, the execution is resumed when the object's signal is emmited.

- *Nil* **assert (** *bool* condition **)**

Assert that the condition is true. If the condition is false, generates an error.

9.2 @Global Scope

Category: Core

9.2.1 Brief Description

Global scope constants and variables.

9.2.2 Member Variables

- *Performance* **Performance** - [Performance] singleton
- *Globals* **Globals** - [Globals] singleton
- *IP* **IP** - [IP] singleton
- *Geometry* **Geometry** - [Geometry] singleton
- *ResourceLoader* **ResourceLoader** - [ResourceLoader] singleton
- *ResourceSaver* **ResourceSaver** - [ResourceSaver] singleton
- *PathRemap* **PathRemap** - [PathRemap] singleton
- *OS* **OS** - [OS] singleton
- *Reference* **Marshalls** - [Marshalls] singleton
- *TranslationServer* **TranslationServer** - [TranslationServer] singleton
- *TranslationServer* **TS** - [TranslationServer] singleton
- *Input* **Input** - [Input] singleton
- *InputMap* **InputMap** - [InputMap] singleton
- *VisualServer* **VisualServer** - [VisualServer] singleton
- *VisualServer* **VS** - [VisualServer] singleton
- *AudioServer* **AudioServer** - [AudioServer] singleton
- *AudioServer* **AS** - [AudioServer] singleton
- *PhysicsServer* **PhysicsServer** - [PhysicsServer] singleton
- *PhysicsServer* **PS** - [PhysicsServer] singleton
- *Physics2DServer* **Physics2DServer** - [Physics2DServer] singleton
- *Physics2DServer* **PS2D** - [Physics2DServer] singleton

- *SpatialSoundServer* **SpatialSoundServer** - [SpatialSoundServer] singleton
- *SpatialSoundServer* **SS** - [SpatialSoundServer] singleton
- *SpatialSound2DServer* **SpatialSound2DServer** - [SpatialSound2DServer] singleton
- *SpatialSound2DServer* **SS2D** - [SpatialSound2DServer] singleton

9.2.3 Numeric Constants

- **MARGIN_LEFT = 0** — Left margin, used usually for *Control* or *StyleBox* derived classes.
- **MARGIN_TOP = 1** — Top margin, used usually for *Control* or *StyleBox* derived classes.
- **MARGIN_RIGHT = 2** — Right margin, used usually for *Control* or *StyleBox* derived classes.
- **MARGIN_BOTTOM = 3** — Bottom margin, used usually for *Control* or *StyleBox* derived classes.
- **VERTICAL = 1** — General vertical alignment, used usually for *Separator*, *ScrollBar*, *Slider*, etc.
- **HORIZONTAL = 0** — General horizontal alignment, used usually for *Separator*, *ScrollBar*, *Slider*, etc.
- **HALIGN_LEFT = 0** — Horizontal left alignment, usually for text-derived classes.
- **HALIGN_CENTER = 1** — Horizontal center alignment, usually for text-derived classes.
- **HALIGN_RIGHT = 2** — Horizontal right alignment, usually for text-derived classes.
- **VALIGN_TOP = 0** — Vertical top alignment, usually for text-derived classes.
- **VALIGN_CENTER = 1** — Vertical center alignment, usually for text-derived classes.
- **VALIGN_BOTTOM = 2** — Vertical bottom alignment, usually for text-derived classes.
- **SPKEY = 16777216** — Scancodes with this bit applied are non printable.
- **KEY_ESCAPE = 16777217** — Escape Key
- **KEY_TAB = 16777218** — Tab Key
- **KEY_BACKTAB = 16777219** — Shift-Tab Key
- **KEY_BACKSPACE = 16777220** — Backspace Key
- **KEY_RETURN = 16777221** — Return Key
- **KEY_ENTER = 16777222** — Enter Key
- **KEY_INSERT = 16777223** — Insert Key
- **KEY_DELETE = 16777224** — Delete Key
- **KEY_PAUSE = 16777225** — Pause Key
- **KEY_PRINT = 16777226** — Printscreen Key
- **KEY_SYSREQ = 16777227**
- **KEY_CLEAR = 16777228**
- **KEY_HOME = 16777229** — Home Key
- **KEY_END = 16777230** — End Key
- **KEY_LEFT = 16777231** — Left Arrow Key
- **KEY_UP = 16777232** — Up Arrow Key
- **KEY_RIGHT = 16777233** — Right Arrow Key

- **KEY_DOWN = 16777234** — Down Arrow Key
- **KEY_PAGEUP = 16777235** — Pageup Key
- **KEY_PAGEDOWN = 16777236** — Pagedown Key
- **KEY_SHIFT = 16777237** — Shift Key
- **KEY_CONTROL = 16777238** — Control Key
- **KEY_META = 16777239**
- **KEY_ALT = 16777240** — Alt Key
- **KEY_CAPSLOCK = 16777241** — Capslock Key
- **KEY_NUMLOCK = 16777242** — Numlock Key
- **KEY_SCROLLLOCK = 16777243** — Scrolllock Key
- **KEY_F1 = 16777244** — F1 Key
- **KEY_F2 = 16777245** — F2 Key
- **KEY_F3 = 16777246** — F3 Key
- **KEY_F4 = 16777247** — F4 Key
- **KEY_F5 = 16777248** — F5 Key
- **KEY_F6 = 16777249** — F6 Key
- **KEY_F7 = 16777250** — F7 Key
- **KEY_F8 = 16777251** — F8 Key
- **KEY_F9 = 16777252** — F9 Key
- **KEY_F10 = 16777253** — F10 Key
- **KEY_F11 = 16777254** — F11 Key
- **KEY_F12 = 16777255** — F12 Key
- **KEY_F13 = 16777256** — F13 Key
- **KEY_F14 = 16777257** — F14 Key
- **KEY_F15 = 16777258** — F15 Key
- **KEY_F16 = 16777259** — F16 Key
- **KEY_KP_ENTER = 16777344** — Enter Key on Numpad
- **KEY_KP_MULTIPLY = 16777345** — Multiply Key on Numpad
- **KEY_KP_DIVIDE = 16777346** — Divide Key on Numpad
- **KEY_KP_SUBTRACT = 16777347** — Subtract Key on Numpad
- **KEY_KP_PERIOD = 16777348** — Period Key on Numpad
- **KEY_KP_ADD = 16777349** — Add Key on Numpad
- **KEY_KP_0 = 16777350** — Number 0 on Numpad
- **KEY_KP_1 = 16777351** — Number 1 on Numpad
- **KEY_KP_2 = 16777352** — Number 2 on Numpad
- **KEY_KP_3 = 16777353** — Number 3 on Numpad

- **KEY_KP_4 = 16777354** — Number 4 on Numpad
- **KEY_KP_5 = 16777355** — Number 5 on Numpad
- **KEY_KP_6 = 16777356** — Number 6 on Numpad
- **KEY_KP_7 = 16777357** — Number 7 on Numpad
- **KEY_KP_8 = 16777358** — Number 8 on Numpad
- **KEY_KP_9 = 16777359** — Number 9 on Numpad
- **KEY_SUPER_L = 16777260** — Super Left key (windows key)
- **KEY_SUPER_R = 16777261** — Super Left key (windows key)
- **KEY_MENU = 16777262** — Context menu key
- **KEY_HYPER_L = 16777263**
- **KEY_HYPER_R = 16777264**
- **KEY_HELP = 16777265** — Help key
- **KEY_DIRECTION_L = 16777266**
- **KEY_DIRECTION_R = 16777267**
- **KEY_BACK = 16777280** — Back key
- **KEY_FORWARD = 16777281** — Forward key
- **KEY_STOP = 16777282** — Stop key
- **KEY_REFRESH = 16777283** — Refresh key
- **KEY_VOLUMEDOWN = 16777284** — Volume down key
- **KEY_VOLUMEMUTE = 16777285** — Mute volume key
- **KEY_VOLUMEUP = 16777286** — Volume up key
- **KEY_BASSBOOST = 16777287**
- **KEY_BASSUP = 16777288**
- **KEY_BASSDOWN = 16777289**
- **KEY_TREBLEUP = 16777290**
- **KEY_TREBLEDOWN = 16777291**
- **KEY_MEDIAPLAY = 16777292** — Media play key
- **KEY_MEDIASTOP = 16777293** — Media stop key
- **KEY_MEDIAPREVIOUS = 16777294** — Previous song key
- **KEY_MEDIANEXT = 16777295** — Next song key
- **KEY_MEDIARECORD = 16777296** — Media record key
- **KEY_HOMEPAGE = 16777297** — Home page key
- **KEY_FAVORITES = 16777298** — Favorites key
- **KEY_SEARCH = 16777299** — Search key
- **KEY_STANDBY = 16777300**
- **KEY_OPENURL = 16777301**

- **KEY_LAUNCHMAIL = 16777302**
- **KEY_LAUNCHMEDIA = 16777303**
- **KEY_LAUNCH0 = 16777304**
- **KEY_LAUNCH1 = 16777305**
- **KEY_LAUNCH2 = 16777306**
- **KEY_LAUNCH3 = 16777307**
- **KEY_LAUNCH4 = 16777308**
- **KEY_LAUNCH5 = 16777309**
- **KEY_LAUNCH6 = 16777310**
- **KEY_LAUNCH7 = 16777311**
- **KEY_LAUNCH8 = 16777312**
- **KEY_LAUNCH9 = 16777313**
- **KEY_LAUNCHA = 16777314**
- **KEY_LAUNCHB = 16777315**
- **KEY_LAUNCHC = 16777316**
- **KEY_LAUNCHD = 16777317**
- **KEY_LAUNCHE = 16777318**
- **KEY_LAUNCHF = 16777319**
- **KEY_UNKNOWN = 33554431**
- **KEY_SPACE = 32** — Space Key
- **KEY_EXCLAM = 33** — ! key
- **KEY_QUOTEDBL = 34** — ” key
- **KEY_NUMBERSIGN = 35** — # key
- **KEY_DOLLAR = 36** — $ key
- **KEY_PERCENT = 37** — % key
- **KEY_AMPERSAND = 38** — & key
- **KEY_APOSTROPHE = 39** — ‘ key
- **KEY_PARENLEFT = 40** — (key
- **KEY_PARENRIGHT = 41** —) key
- **KEY_ASTERISK = 42** — * key
- **KEY_PLUS = 43** — + key
- **KEY_COMMA = 44** — , key
- **KEY_MINUS = 45** — - key
- **KEY_PERIOD = 46** — . key
- **KEY_SLASH = 47** — / key
- **KEY_0 = 48** — Number 0

- **KEY_1 = 49** — Number 1
- **KEY_2 = 50** — Number 2
- **KEY_3 = 51** — Number 3
- **KEY_4 = 52** — Number 4
- **KEY_5 = 53** — Number 5
- **KEY_6 = 54** — Number 6
- **KEY_7 = 55** — Number 7
- **KEY_8 = 56** — Number 8
- **KEY_9 = 57** — Number 9
- **KEY_COLON = 58** — : key
- **KEY_SEMICOLON = 59** — ; key
- **KEY_LESS = 60** — Lower than key
- **KEY_EQUAL = 61** — = key
- **KEY_GREATER = 62** — Greater than key
- **KEY_QUESTION = 63** — ? key
- **KEY_AT = 64** — @ key
- **KEY_A = 65** — A Key
- **KEY_B = 66** — B Key
- **KEY_C = 67** — C Key
- **KEY_D = 68** — D Key
- **KEY_E = 69** — E Key
- **KEY_F = 70** — F Key
- **KEY_G = 71** — G Key
- **KEY_H = 72** — H Key
- **KEY_I = 73** — I Key
- **KEY_J = 74** — J Key
- **KEY_K = 75** — K Key
- **KEY_L = 76** — L Key
- **KEY_M = 77** — M Key
- **KEY_N = 78** — N Key
- **KEY_O = 79** — O Key
- **KEY_P = 80** — P Key
- **KEY_Q = 81** — Q Key
- **KEY_R = 82** — R Key
- **KEY_S = 83** — S Key
- **KEY_T = 84** — T Key

- **KEY_U = 85** — U Key
- **KEY_V = 86** — V Key
- **KEY_W = 87** — W Key
- **KEY_X = 88** — X Key
- **KEY_Y = 89** — Y Key
- **KEY_Z = 90** — Z Key
- **KEY_BRACKETLEFT = 91** — [key
- **KEY_BACKSLASH = 92** — key
- **KEY_BRACKETRIGHT = 93** —] key
- **KEY_ASCIICIRCUM = 94** — ^ key
- **KEY_UNDERSCORE = 95** — _ key
- **KEY_QUOTELEFT = 96**
- **KEY_BRACELEFT = 123** — { key
- **KEY_BAR = 124** — | key
- **KEY_BRACERIGHT = 125** — } key
- **KEY_ASCIITILDE = 126** — ~ key
- **KEY_NOBREAKSPACE = 160**
- **KEY_EXCLAMDOWN = 161**
- **KEY_CENT = 162** — ¢ key
- **KEY_STERLING = 163**
- **KEY_CURRENCY = 164**
- **KEY_YEN = 165**
- **KEY_BROKENBAR = 166** — ¦ key
- **KEY_SECTION = 167** — § key
- **KEY_DIAERESIS = 168** — ¨ key
- **KEY_COPYRIGHT = 169** — © key
- **KEY_ORDFEMININE = 170**
- **KEY_GUILLEMOTLEFT = 171** — « key
- **KEY_NOTSIGN = 172** — » key
- **KEY_HYPHEN = 173** — key
- **KEY_REGISTERED = 174** — ® key
- **KEY_MACRON = 175**
- **KEY_DEGREE = 176** — ° key
- **KEY_PLUSMINUS = 177** — ± key
- **KEY_TWOSUPERIOR = 178** — ^{2} key
- **KEY_THREESUPERIOR = 179** — ^{3} key

- **KEY_ACUTE = 180** — ´ key
- **KEY_MU = 181** — µ key
- **KEY_PARAGRAPH = 182**
- **KEY_PERIODCENTERED = 183** — · key
- **KEY_CEDILLA = 184** — ¬ key
- **KEY_ONESUPERIOR = 185**
- **KEY_MASCULINE = 186**
- **KEY_GUILLEMOTRIGHT = 187**
- **KEY_ONEQUARTER = 188**
- **KEY_ONEHALF = 189** — ½ key
- **KEY_THREEQUARTERS = 190**
- **KEY_QUESTIONDOWN = 191**
- **KEY_AGRAVE = 192**
- **KEY_AACUTE = 193**
- **KEY_ACIRCUMFLEX = 194**
- **KEY_ATILDE = 195**
- **KEY_ADIAERESIS = 196**
- **KEY_ARING = 197**
- **KEY_AE = 198**
- **KEY_CCEDILLA = 199**
- **KEY_EGRAVE = 200**
- **KEY_EACUTE = 201**
- **KEY_ECIRCUMFLEX = 202**
- **KEY_EDIAERESIS = 203**
- **KEY_IGRAVE = 204**
- **KEY_IACUTE = 205**
- **KEY_ICIRCUMFLEX = 206**
- **KEY_IDIAERESIS = 207**
- **KEY_ETH = 208**
- **KEY_NTILDE = 209**
- **KEY_OGRAVE = 210**
- **KEY_OACUTE = 211**
- **KEY_OCIRCUMFLEX = 212**
- **KEY_OTILDE = 213**
- **KEY_ODIAERESIS = 214**
- **KEY_MULTIPLY = 215**

- **KEY_OOBLIQUE = 216**
- **KEY_UGRAVE = 217**
- **KEY_UACUTE = 218**
- **KEY_UCIRCUMFLEX = 219**
- **KEY_UDIAERESIS = 220**
- **KEY_YACUTE = 221**
- **KEY_THORN = 222**
- **KEY_SSHARP = 223**
- **KEY_DIVISION = 247**
- **KEY_YDIAERESIS = 255**
- **KEY_CODE_MASK = 33554431**
- **KEY_MODIFIER_MASK = -16777216**
- **KEY_MASK_SHIFT = 33554432**
- **KEY_MASK_ALT = 67108864**
- **KEY_MASK_META = 134217728**
- **KEY_MASK_CTRL = 268435456**
- **KEY_MASK_CMD = 268435456**
- **KEY_MASK_KPAD = 536870912**
- **KEY_MASK_GROUP_SWITCH = 1073741824**
- **BUTTON_LEFT = 1** — Left Mouse Button
- **BUTTON_RIGHT = 2** — Right Mouse Button
- **BUTTON_MIDDLE = 3** — Middle Mouse Button
- **BUTTON_WHEEL_UP = 4** — Mouse wheel up
- **BUTTON_WHEEL_DOWN = 5** — Mouse wheel down
- **BUTTON_WHEEL_LEFT = 6** — Mouse wheel left button
- **BUTTON_WHEEL_RIGHT = 7** — Mouse wheel right button
- **BUTTON_MASK_LEFT = 1**
- **BUTTON_MASK_RIGHT = 2**
- **BUTTON_MASK_MIDDLE = 4**
- **JOY_BUTTON_0 = 0** — Joystick Button 0
- **JOY_BUTTON_1 = 1** — Joystick Button 1
- **JOY_BUTTON_2 = 2** — Joystick Button 2
- **JOY_BUTTON_3 = 3** — Joystick Button 3
- **JOY_BUTTON_4 = 4** — Joystick Button 4
- **JOY_BUTTON_5 = 5** — Joystick Button 5
- **JOY_BUTTON_6 = 6** — Joystick Button 6

- **JOY_BUTTON_7 = 7** — Joystick Button 7
- **JOY_BUTTON_8 = 8** — Joystick Button 8
- **JOY_BUTTON_9 = 9** — Joystick Button 9
- **JOY_BUTTON_10 = 10** — Joystick Button 10
- **JOY_BUTTON_11 = 11** — Joystick Button 11
- **JOY_BUTTON_12 = 12** — Joystick Button 12
- **JOY_BUTTON_13 = 13** — Joystick Button 13
- **JOY_BUTTON_14 = 14** — Joystick Button 14
- **JOY_BUTTON_15 = 15** — Joystick Button 15
- **JOY_BUTTON_MAX = 16** — Joystick Button 16
- **JOY_SNES_A = 1** — Super Nintendo Entertaiment System controller A button
- **JOY_SNES_B = 0** — Super Nintendo Entertaiment System controller B button
- **JOY_SNES_X = 3** — Super Nintendo Entertaiment System controller X button
- **JOY_SNES_Y = 2** — Super Nintendo Entertaiment System controller Y button
- **JOY_SONY_CIRCLE = 1** — DUALSHOCK circle button
- **JOY_SONY_X = 0** — DUALSHOCK X button
- **JOY_SONY_SQUARE = 2** — DUALSHOCK square button
- **JOY_SONY_TRIANGLE = 3** — DUALSHOCK triangle button
- **JOY_SEGA_B = 1** — SEGA controller B button
- **JOY_SEGA_A = 0** — SEGA controller A button
- **JOY_SEGA_X = 2** — SEGA controller X button
- **JOY_SEGA_Y = 3** — SEGA controller Y button
- **JOY_XBOX_B = 1** — XBOX controller B button
- **JOY_XBOX_A = 0** — XBOX controller A button
- **JOY_XBOX_X = 2** — XBOX controller X button
- **JOY_XBOX_Y = 3** — XBOX controller Y button
- **JOY_DS_A = 1**
- **JOY_DS_B = 0**
- **JOY_DS_X = 3**
- **JOY_DS_Y = 2**
- **JOY_SELECT = 10** — Joystick Button Select
- **JOY_START = 11** — Joystick Button Start
- **JOY_DPAD_UP = 12** — Joystick DPad Up
- **JOY_DPAD_DOWN = 13** — Joystick DPad Down
- **JOY_DPAD_LEFT = 14** — Joystick DPad Left
- **JOY_DPAD_RIGHT = 15** — Joystick DPad Right

- **JOY_L = 4** — Joystick Left Shoulder Button
- **JOY_L2 = 6** — Joystick Left Trigger
- **JOY_L3 = 8** — Joystick Left Stick Click
- **JOY_R = 5** — Joystick Right Shoulder Button
- **JOY_R2 = 7** — Joystick Right Trigger
- **JOY_R3 = 9** — Joystick Right Stick Click
- **JOY_AXIS_0 = 0** — Joystick Left Stick Horizontal Axis
- **JOY_AXIS_1 = 1** — Joystick Left Stick Vertical Axis
- **JOY_AXIS_2 = 2** — Joystick Right Stick Horizontal Axis
- **JOY_AXIS_3 = 3** — Joystick Right Stick Vertical Axis
- **JOY_AXIS_4 = 4**
- **JOY_AXIS_5 = 5**
- **JOY_AXIS_6 = 6** — Joystick Left Trigger Analog Axis
- **JOY_AXIS_7 = 7** — Joystick Right Trigger Analog Axis
- **JOY_AXIS_MAX = 8**
- **JOY_ANALOG_0_X = 0** — Joystick Left Stick Horizontal Axis
- **JOY_ANALOG_0_Y = 1** — Joystick Left Stick Vertical Axis
- **JOY_ANALOG_1_X = 2** — Joystick Right Stick Horizontal Axis
- **JOY_ANALOG_1_Y = 3** — Joystick Right Stick Vertical Axis
- **JOY_ANALOG_2_X = 4**
- **JOY_ANALOG_2_Y = 5**
- **JOY_ANALOG_L2 = 6**
- **JOY_ANALOG_R2 = 7**
- **OK = 0** — Functions that return Error return OK when everything went ok. Most functions don't return error anyway and/or just print errors to stdout.
- **FAILED = 1** — Generic fail return error.
- **ERR_UNAVAILABLE = 2**
- **ERR_UNCONFIGURED = 3**
- **ERR_UNAUTHORIZED = 4**
- **ERR_PARAMETER_RANGE_ERROR = 5**
- **ERR_OUT_OF_MEMORY = 6**
- **ERR_FILE_NOT_FOUND = 7**
- **ERR_FILE_BAD_DRIVE = 8**
- **ERR_FILE_BAD_PATH = 9**
- **ERR_FILE_NO_PERMISSION = 10**
- **ERR_FILE_ALREADY_IN_USE = 11**

- **ERR_FILE_CANT_OPEN = 12**
- **ERR_FILE_CANT_WRITE = 13**
- **ERR_FILE_CANT_READ = 14**
- **ERR_FILE_UNRECOGNIZED = 15**
- **ERR_FILE_CORRUPT = 16**
- **ERR_FILE_MISSING_DEPENDENCIES = 17**
- **ERR_FILE_EOF = 18**
- **ERR_CANT_OPEN = 19**
- **ERR_CANT_CREATE = 20**
- **ERROR_QUERY_FAILED = 21**
- **ERR_ALREADY_IN_USE = 22**
- **ERR_LOCKED = 23**
- **ERR_TIMEOUT = 24**
- **ERR_CANT_AQUIRE_RESOURCE = 28**
- **ERR_INVALID_DATA = 30**
- **ERR_INVALID_PARAMETER = 31**
- **ERR_ALREADY_EXISTS = 32**
- **ERR_DOES_NOT_EXIST = 33**
- **ERR_DATABASE_CANT_READ = 34**
- **ERR_DATABASE_CANT_WRITE = 35**
- **ERR_COMPILATION_FAILED = 36**
- **ERR_METHOD_NOT_FOUND = 37**
- **ERR_LINK_FAILED = 38**
- **ERR_SCRIPT_FAILED = 39**
- **ERR_CYCLIC_LINK = 40**
- **ERR_BUSY = 44**
- **ERR_HELP = 46**
- **ERR_BUG = 47**
- **ERR_WTF = 49**
- **PROPERTY_HINT_NONE = 0** — No hint for edited property.
- **PROPERTY_HINT_RANGE = 1** — Hints that the string is a range, defined as “min,max” or “min,max,step”. This is valid for integers and floats.
- **PROPERTY_HINT_EXP_RANGE = 2** — Hints that the string is an exponential range, defined as “min,max” or “min,max,step”. This is valid for integers and floats.
- **PROPERTY_HINT_ENUM = 3** — Property hint for an enumerated value, like “Hello,Something,Else”. This is valid for integer, float and string properties.
- **PROPERTY_HINT_EXP_EASING = 4**

- **PROPERTY_HINT_LENGTH = 5**
- **PROPERTY_HINT_KEY_ACCEL = 7**
- **PROPERTY_HINT_FLAGS = 8** — Property hint for a bitmask description, for bits 0,1,2,3 and 5 the hint would be like "Bit0,Bit1,Bit2,Bit3,,Bit5". Valid only for integers.
- **PROPERTY_HINT_ALL_FLAGS = 9** — Property hint for a bitmask description that covers all 32 bits. Valid only for integers.
- **PROPERTY_HINT_FILE = 10** — String property is a file (so pop up a file dialog when edited). Hint string can be a set of wildcards like "*.doc".
- **PROPERTY_HINT_DIR = 11** — String property is a directory (so pop up a file dialog when edited).
- **PROPERTY_HINT_GLOBAL_FILE = 12**
- **PROPERTY_HINT_GLOBAL_DIR = 13**
- **PROPERTY_HINT_RESOURCE_TYPE = 14** — String property is a resource, so open the resource popup menu when edited.
- **PROPERTY_HINT_MULTILINE_TEXT = 15**
- **PROPERTY_HINT_COLOR_NO_ALPHA = 16**
- **PROPERTY_HINT_IMAGE_COMPRESS_LOSSY = 17**
- **PROPERTY_HINT_IMAGE_COMPRESS_LOSSLESS = 18**
- **PROPERTY_USAGE_STORAGE = 1** — Property will be used as storage (default).
- **PROPERTY_USAGE_EDITOR = 2** — Property will be visible in editor (default).
- **PROPERTY_USAGE_NETWORK = 4**
- **PROPERTY_USAGE_DEFAULT = 7** — Default usage (storage and editor).
- **METHOD_FLAG_NORMAL = 1**
- **METHOD_FLAG_EDITOR = 2**
- **METHOD_FLAG_NOSCRIPT = 4**
- **METHOD_FLAG_CONST = 8**
- **METHOD_FLAG_REVERSE = 16**
- **METHOD_FLAG_VIRTUAL = 32**
- **METHOD_FLAG_FROM_SCRIPT = 64**
- **METHOD_FLAGS_DEFAULT = 1**
- **TYPE_NIL = 0** — Variable is of type nil (only applied for null).
- **TYPE_BOOL = 1** — Variable is of type *bool*.
- **TYPE_INT = 2** — Variable is of type *int*.
- **TYPE_REAL = 3** — Variable is of type *float*/real.
- **TYPE_STRING = 4** — Variable is of type *String*.
- **TYPE_VECTOR2 = 5** — Variable is of type *Vector2*.
- **TYPE_RECT2 = 6** — Variable is of type *Rect2*.
- **TYPE_VECTOR3 = 7** — Variable is of type *Vector3*.

- **TYPE_MATRIX32 = 8** — Variable is of type *Matrix32*.
- **TYPE_PLANE = 9** — Variable is of type *Plane*.
- **TYPE_QUAT = 10** — Variable is of type *Quat*.
- **TYPE_AABB = 11** — Variable is of type *AABB*.
- **TYPE_MATRIX3 = 12** — Variable is of type *Matrix3*.
- **TYPE_TRANSFORM = 13** — Variable is of type *Transform*.
- **TYPE_COLOR = 14** — Variable is of type *Color*.
- **TYPE_IMAGE = 15** — Variable is of type *Image*.
- **TYPE_NODE_PATH = 16** — Variable is of type *NodePath*.
- **TYPE_RID = 17** — Variable is of type *RID*.
- **TYPE_OBJECT = 18** — Variable is of type *Object*.
- **TYPE_INPUT_EVENT = 19** — Variable is of type *InputEvent*.
- **TYPE_DICTIONARY = 20** — Variable is of type *Dictionary*.
- **TYPE_ARRAY = 21** — Variable is of type *Array*.
- **TYPE_RAW_ARRAY = 22**
- **TYPE_INT_ARRAY = 23**
- **TYPE_REAL_ARRAY = 24**
- **TYPE_STRING_ARRAY = 25**
- **TYPE_VECTOR2_ARRAY = 26**
- **TYPE_VECTOR3_ARRAY = 27**
- **TYPE_COLOR_ARRAY = 28**
- **TYPE_MAX = 29**

9.2.4 Description

Global scope constants and variables. This is all that resides in the globals, constants regarding error codes, scancodes, property hints, etc. It's not much.

Singletons are also documented here, since they can be accessed from anywhere.

9.3 AABB

Category: Built-In Types

9.3.1 Brief Description

Axis-Aligned Bounding Box.

9.3.2 Member Functions

bool	*encloses* (*AABB* with)
AABB	*expand* (*Vector3* to_point)
float	*get_area* ( )
Vector3	*get_endpoint* (*int* idx)
Vector3	*get_longest_axis* ( )
int	*get_longest_axis_index* ( )
float	*get_longest_axis_size* ( )
Vector3	*get_shortest_axis* ( )
int	*get_shortest_axis_index* ( )
float	*get_shortest_axis_size* ( )
Vector3	*get_support* (*Vector3* dir)
AABB	*grow* (*float* by)
bool	*has_no_area* ( )
bool	*has_no_surface* ( )
bool	*has_point* (*Vector3* point)
AABB	*intersection* (*AABB* with)
bool	*intersects* (*AABB* with)
bool	*intersects_plane* (*Plane* plane)
bool	*intersects_segment* (*Vector3* from, *Vector3* to)
AABB	*merge* (*AABB* with)
AABB	*AABB* (*Vector3* pos, *Vector3* size)

9.3.3 Member Variables

- *Vector3* **pos** - Position (starting corner).
- *Vector3* **size** - Size from position to end.
- *Vector3* **end** - Ending corner.

9.3.4 Description

AABB provides an 3D Axis-Aligned Bounding Box. It consists of a position, a size, and several utility functions. It is typically used for simple (fast) overlap tests.

9.3.5 Member Function Description

- *bool* **encloses** (*AABB* with)

Return true if this *AABB* completely encloses another one.

- *AABB* **expand** (*Vector3* to_point)

Return this *AABB* expanded to include a given point.

- *float* **get_area** ( )

Get the area of the *AABB*.

- *Vector3* **get_endpoint** (*int* idx)

Get the position of the 8 endpoints of the *AABB* in space.

- *Vector3* **get_longest_axis ( )**

Return the normalized longest axis of the *AABB*.

- *int* **get_longest_axis_index ( )**

Return the index of the longest axis of the *AABB* (according to *Vector3*::AXIS* enum).

- *float* **get_longest_axis_size ( )**

Return the scalar length of the longest axis of the *AABB*.

- *Vector3* **get_shortest_axis ( )**

Return the normalized shortest axis of the *AABB*.

- *int* **get_shortest_axis_index ( )**

Return the index of the shortest axis of the *AABB* (according to *Vector3*::AXIS* enum).

- *float* **get_shortest_axis_size ( )**

Return the scalar length of the shortest axis of the *AABB*.

- *Vector3* **get_support (***Vector3*** dir)**

Return the support point in a given direction. This is useful for collision detection algorithms.

- *AABB* **grow (***float*** by)**

Return a copy of the *AABB* grown a given amount of units towards all the sides.

- *bool* **has_no_area ( )**

Return true if the *AABB* is flat or empty.

- *bool* **has_no_surface ( )**

Return true if the *AABB* is empty.

- *bool* **has_point (***Vector3*** point)**

Return true if the *AABB* contains a point.

- *AABB* **intersection (***AABB*** with)**

Return the intersection between two *AABB*. An empty AABB (size 0,0,0) is returned on failure.

- *bool* **intersects (***AABB*** with)**

Return true if the *AABB* overlaps with another.

- *bool* **intersects_plane (***Plane*** plane)**

Return true if the *AABB* is at both sides of a plane.

- *bool* **intersects_segment (***Vector3*** from, ***Vector3*** to)**

Return true if the *AABB* intersects the line segment between from and to

- *AABB* **merge (***AABB*** with)**

Combine this *AABB* with another, a larger one is returned that contains both.

- *AABB* **AABB (***Vector3*** pos, ***Vector3*** size)**

Optional constructor, accepts position and size.

9.4 AcceptDialog

Inherits: *WindowDialog* < *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *ConfirmationDialog*

Category: Core

9.4.1 Brief Description

Base dialog for user notification.

9.4.2 Member Functions

Object	*get_ok* ( )
Object	*get_label* ( )
void	*set_hide_on_ok* (*bool* enabled)
bool	*get_hide_on_ok* ( ) const
Button	*add_button* (*String* text, *bool* right=false, *String* action="")
Button	*add_cancel* (*String* name)
LineEdit	*register_text_enter* (*Object* line_edit)
void	*set_text* (*String* text)
String	*get_text* ( ) const

9.4.3 Signals

- **confirmed** ( )
- **custom_action** (*String* action)

9.4.4 Description

This dialog is useful for small notifications to the user about an event. It can only be accepted or closed, with the same result.

9.4.5 Member Function Description

- *Object* **get_ok** ( )

Return the OK Button.

- *Object* **get_label** ( )

Return the label used for built-in text.

- void **set_hide_on_ok** (*bool* enabled)

Set whether the dialog is hidden when accepted (default true).

- *bool* **get_hide_on_ok** ( ) const

Return true if the dialog will be hidden when accepted (default true).

- *Button* **add_button** (*String* text, *bool* right=false, *String* action="")

Add custom button to the dialog and return the created button.

The button titled with *text* and the *action* will be passed to custom_action signal when it is pressed.

- *Button* **add_cancel** (*String* name)

Add custom cancel button to the dialog and return the created button.

- *LineEdit* **register_text_enter** (*Object* line_edit)

Register a *LineEdit* in the dialog. When the enter key is pressed, the dialog will be accepted.

- void **set_text** (*String* text)

Set the built-in label text.

- *String* **get_text** ( ) const

Return the built-in label text.

9.5 AnimatedSprite

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.5.1 Brief Description

Sprite node that can use multiple textures for animation.

9.5.2 Member Functions

void	*set_sprite_frames* (*SpriteFrames* sprite_frames)
SpriteFrames	*get_sprite_frames* ( ) const
void	*set_centered* (*bool* centered)
bool	*is_centered* ( ) const
void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const
void	*set_flip_h* (*bool* flip_h)
bool	*is_flipped_h* ( ) const
void	*set_flip_v* (*bool* flip_v)
bool	*is_flipped_v* ( ) const
void	*set_frame* (*int* frame)
int	*get_frame* ( ) const
void	*set_modulate* (*Color* modulate)
Color	*get_modulate* ( ) const

9.5.3 Signals

- **frame_changed** ( )

9.5.4 Description

Sprite node that can use multiple textures for animation.

9.5.5 Member Function Description

- void **set_sprite_frames** (*SpriteFrames* sprite_frames)

Set the *SpriteFrames* resource, which contains all frames.

- *SpriteFrames* **get_sprite_frames** ( ) const

Get the *SpriteFrames* resource, which contains all frames.

- void **set_centered** (*bool* centered)

When turned on, offset at (0,0) is the center of the sprite, when off, the top-left corner is.

- *bool* **is_centered** ( ) const

Return true when centered. See *set_centered*.

- void **set_offset** (*Vector2* offset)

Set the offset of the sprite in the node origin. Position varies depending on whether it is centered or not.

- *Vector2* **get_offset** ( ) const

Return the offset of the sprite in the node origin.

- void **set_flip_h** (*bool* flip_h)

If true, sprite is flipped horizontally.

- *bool* **is_flipped_h** ( ) const

Return true if sprite is flipped horizontally.

- void **set_flip_v** (*bool* flip_v)

If true, sprite is flipped vertically.

- *bool* **is_flipped_v** ( ) const

Return true if sprite is flipped vertically.

- void **set_frame** (*int* frame)

Set the visible sprite frame index (from the list of frames inside the *SpriteFrames* resource).

- *int* **get_frame** ( ) const

Return the visible frame index.

- void **set_modulate** (*Color* modulate)

Change the color modulation (multiplication) for this sprite.

- *Color* **get_modulate** ( ) const

Return the color modulation for this sprite.

9.6 AnimatedSprite3D

Inherits: *SpriteBase3D* < *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.6.1 Brief Description

9.6.2 Member Functions

void	*set_sprite_frames* (*SpriteFrames* sprite_frames)
Texture	*get_sprite_frames* ( ) const
void	*set_frame* (*int* frame)
int	*get_frame* ( ) const

9.6.3 Signals

- **frame_changed** ( )

9.6.4 Member Function Description

- void **set_sprite_frames** (*SpriteFrames* sprite_frames)
- *Texture* **get_sprite_frames** ( ) const
- void **set_frame** (*int* frame)
- *int* **get_frame** ( ) const

9.7 Animation

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.7.1 Brief Description

Contains data used to animate everything in the engine.

9.7.2 Member Functions

int	*add_track* (*int* type, *int* at_pos=-1)
void	*remove_track* (*int* idx)
int	*get_track_count* ( ) const
int	*track_get_type* (*int* idx) const
NodePath	*track_get_path* (*int* idx) const
void	*track_set_path* (*int* idx, *NodePath* path)

Continúa en la página siguiente

Tabla 9.2 – proviene de la página anterior

int	*find_track* (*NodePath* path) const
void	*track_move_up* (*int* idx)
void	*track_move_down* (*int* idx)
int	*transform_track_insert_key* (*int* idx, *float* time, *Vector3* loc, *Quat* rot, *Vector3* scale)
void	*track_insert_key* (*int* idx, *float* time, var key, *float* transition=1)
void	*track_remove_key* (*int* idx, *int* key_idx)
void	*track_remove_key_at_pos* (*int* idx, *float* pos)
void	*track_set_key_value* (*int* idx, *int* key, var value)
void	*track_set_key_transition* (*int* idx, *int* key_idx, *float* transition)
float	*track_get_key_transition* (*int* idx, *int* key_idx) const
int	*track_get_key_count* (*int* idx) const
void	*track_get_key_value* (*int* idx, *int* key_idx) const
float	*track_get_key_time* (*int* idx, *int* key_idx) const
int	*track_find_key* (*int* idx, *float* time, *bool* exact=false) const
void	*track_set_interpolation_type* (*int* idx, *int* interpolation)
int	*track_get_interpolation_type* (*int* idx) const
Array	*transform_track_interpolate* (*int* idx, *float* time_sec) const
void	*value_track_set_continuous* (*int* idx, *bool* continuous)
bool	*value_track_is_continuous* (*int* idx) const
IntArray	*value_track_get_key_indices* (*int* idx, *float* time_sec, *float* delta) const
IntArray	*method_track_get_key_indices* (*int* idx, *float* time_sec, *float* delta) const
String	*method_track_get_name* (*int* idx, *int* key_idx) const
Array	*method_track_get_params* (*int* idx, *int* key_idx) const
void	*set_length* (*float* time_sec)
float	*get_length* ( ) const
void	*set_loop* (*bool* enabled)
bool	*has_loop* ( ) const
void	*set_step* (*float* size_sec)
float	*get_step* ( ) const
void	*clear* ( )

9.7.3 Numeric Constants

- **TYPE_VALUE = 0** — Value tracks set values in node properties, but only those which can be Interpolated.
- **TYPE_TRANSFORM = 1** — Transform tracks are used to change node local transforms or skeleton pose bones. Transitions are Interpolated.
- **TYPE_METHOD = 2** — Method tracks call functions with given arguments per key.
- **INTERPOLATION_NEAREST = 0** — No interpolation (nearest value).
- **INTERPOLATION_LINEAR = 1** — Linear interpolation.
- **INTERPOLATION_CUBIC = 2** — Cubic interpolation.

9.7.4 Description

An Animation resource contains data used to animate everything in the engine. Animations are divided into tracks, and each track must be linked to a node. The state of that node can be changed through time, by adding timed keys (events) to the track.

Animations are just data containers, and must be added to odes such as an *AnimationPlayer* or *AnimationTreePlayer* to be played back.

9.7.5 Member Function Description

- *int* **add_track** (*int* type, *int* at_pos=-1)

Add a track to the Animation. The track type must be specified as any of the values in the TYPE_* enumeration.

- void **remove_track** (*int* idx)

Remove a track by specifying the track index.

- *int* **get_track_count** ( ) const

Return the amount of tracks in the animation.

- *int* **track_get_type** (*int* idx) const

Get the type of a track.

- *NodePath* **track_get_path** (*int* idx) const

Get the path of a track. for more information on the path format, see *track_set_path*

- void **track_set_path** (*int* idx, *NodePath* path)

Set the path of a track. Paths must be valid scene-tree paths to a node, and must be specified starting from the parent node of the node that will reproduce the animation. Tracks that control properties or bones must append their name after the path, separated by ":". Example: "character/skeleton:ankle" or "character/mesh:transform/local"

- *int* **find_track** (*NodePath* path) const
- void **track_move_up** (*int* idx)

Move a track up.

- void **track_move_down** (*int* idx)

Move a track down.

- *int* **transform_track_insert_key** (*int* idx, *float* time, *Vector3* loc, *Quat* rot, *Vector3* scale)

Insert a transform key for a transform track.

- void **track_insert_key** (*int* idx, *float* time, var key, *float* transition=1)

Insert a generic key in a given track.

- void **track_remove_key** (*int* idx, *int* key_idx)

Remove a key by index in a given track.

- void **track_remove_key_at_pos** (*int* idx, *float* pos)

Remove a key by position (seconds) in a given track.

- void **track_set_key_value** (*int* idx, *int* key, var value)

Set the value of an existing key.

- void **track_set_key_transition** (*int* idx, *int* key_idx, *float* transition)

Set the transition curve (easing) for a specific key (see built-in math function "ease").

- *float* **track_get_key_transition** (*int* idx, *int* key_idx) const

Return the transition curve (easing) for a specific key (see built-in math function "ease").

- *int* **track_get_key_count** (*int* idx) const

Return the amount of keys in a given track.

- void **track_get_key_value** (*int* idx, *int* key_idx) const

Return the value of a given key in a given track.

- *float* **track_get_key_time** (*int* idx, *int* key_idx) const

Return the time at which the key is located.

- *int* **track_find_key** (*int* idx, *float* time, *bool* exact=false) const

Find the key index by time in a given track. Optionally, only find it if the exact time is given.

- void **track_set_interpolation_type** (*int* idx, *int* interpolation)

Set the interpolation type of a given track, from the INTERPOLATION_* enum.

- *int* **track_get_interpolation_type** (*int* idx) const

Return the interpolation type of a given track, from the INTERPOLATION_* enum.

- *Array* **transform_track_interpolate** (*int* idx, *float* time_sec) const

Return the interpolated value of a transform track at a given time (in seconds). An array consisting of 3 elements: position (*Vector3*), rotation (*Quat*) and scale (*Vector3*).

- void **value_track_set_continuous** (*int* idx, *bool* continuous)

Enable or disable interpolation for a whole track. By default tracks are interpolated.

- *bool* **value_track_is_continuous** (*int* idx) const

Return whether interpolation is enabled or disabled for a whole track. By default tracks are interpolated.

- *IntArray* **value_track_get_key_indices** (*int* idx, *float* time_sec, *float* delta) const

Return all the key indices of a value track, given a position and delta time.

- *IntArray* **method_track_get_key_indices** (*int* idx, *float* time_sec, *float* delta) const

Return all the key indices of a method track, given a position and delta time.

- *String* **method_track_get_name** (*int* idx, *int* key_idx) const

Return the method name of a method track.

- *Array* **method_track_get_params** (*int* idx, *int* key_idx) const

Return the arguments values to be called on a method track for a given key in a given track.

- void **set_length** (*float* time_sec)

Set the total length of the animation (in seconds). Note that length is not delimited by the last key, as this one may be before or after the end to ensure correct interpolation and looping.

- *float* **get_length** ( ) const

Return the total length of the animation (in seconds).

- void **set_loop** (*bool* enabled)

Set a flag indicating that the animation must loop. This is uses for correct interpolation of animation cycles, and for hinting the player that it must restart the animation.

- *bool* **has_loop** ( ) const

Return whether the animation has the loop flag set.

- void **set_step** (*float* size_sec)
- *float* **get_step** ( ) const
- void **clear** ( )

Clear the animation (clear all tracks and reset all).

9.8 AnimationPlayer

Inherits: *Node* < *Object*

Category: Core

9.8.1 Brief Description

Container and player of *Animation* resources.

9.8.2 Member Functions

int	*add_animation* (*String* name, *Animation* animation)
void	*remove_animation* (*String* name)
void	*rename_animation* (*String* name, *String* newname)
bool	*has_animation* (*String* name) const
Animation	*get_animation* (*String* name) const
StringArray	*get_animation_list* ( ) const
void	*animation_set_next* (*String* anim_from, *String* anim_to)
String	*animation_get_next* (*String* anim_from) const
void	*set_blend_time* (*String* anim_from, *String* anim_to, *float* sec)
float	*get_blend_time* (*String* anim_from, *String* anim_to) const
void	*set_default_blend_time* (*float* sec)
float	*get_default_blend_time* ( ) const
void	*play* (*String* name="", *float* custom_blend=-1, *float* custom_speed=1, *bool* from_end=false)
void	*play_backwards* (*String* name="", *float* custom_blend=-1)
void	*stop* (*bool* reset=true)
void	*stop_all* ( )
bool	*is_playing* ( ) const
void	*set_current_animation* (*String* anim)
String	*get_current_animation* ( ) const
void	*queue* (*String* name)
void	*clear_queue* ( )
void	*set_active* (*bool* active)
bool	*is_active* ( ) const
void	*set_speed* (*float* speed)
float	*get_speed* ( ) const
void	*set_autoplay* (*String* name)
String	*get_autoplay* ( ) const
void	*set_root* (*NodePath* path)
NodePath	*get_root* ( ) const
void	*seek* (*float* pos_sec, *bool* update=false)

Continúa en la página siguiente

Tabla 9.3 – proviene de la página anterior

float	*get_pos* ( ) const
String	*find_animation* (*Animation* animation) const
void	*clear_caches* ( )
void	*set_animation_process_mode* (*int* mode)
int	*get_animation_process_mode* ( ) const
float	*get_current_animation_pos* ( ) const
float	*get_current_animation_length* ( ) const
void	*advance* (*float* delta)

9.8.3 Signals

- **animation_changed** (*String* old_name, *String* new_name)
- **finished** ( )

9.8.4 Numeric Constants

- **ANIMATION_PROCESS_FIXED = 0** — Process animation on fixed process. This is specially useful when animating kinematic bodies.
- **ANIMATION_PROCESS_IDLE = 1** — Process animation on idle process.

9.8.5 Description

An animation player is used for general purpose playback of *Animation* resources. It contains a dictionary of animations (referenced by name) and custom blend times between their transitions. Additionally, animations can be played and blended in different channels.

9.8.6 Member Function Description

- *int* **add_animation** (*String* name, *Animation* animation)

Add an animation resource to the player, which will be later referenced by the "name" argument.

- void **remove_animation** (*String* name)

Remove an animation from the player (by supplying the same name used to add it).

- void **rename_animation** (*String* name, *String* newname)

Rename an existing animation.

- *bool* **has_animation** (*String* name) const

Request whether an *Animation* name exist within the player.

- *Animation* **get_animation** (*String* name) const

Get an *Animation* resource by requesting a name.

- *StringArray* **get_animation_list** ( ) const

Get the list of names of the animations stored in the player.

- void **animation_set_next** (*String* anim_from, *String* anim_to)
- *String* **animation_get_next** (*String* anim_from) const

- void **set_blend_time** (*String* anim_from, *String* anim_to, *float* sec)

Specify a blend time (in seconds) between two animations, referenced by their names.

- *float* **get_blend_time** (*String* anim_from, *String* anim_to) const

Get the blend time between two animations, referenced by their names.

- void **set_default_blend_time** (*float* sec)

Set the default blend time between animations.

- *float* **get_default_blend_time** ( ) const

Return the default blend time between animations.

- void **play** (*String* name="", *float* custom_blend=-1, *float* custom_speed=1, *bool* from_end=false)

Play a given animation by the animation name. Custom speed and blend times can be set. If custom speed is negative (-1), 'from_end' being true can play the animation backwards.

- void **play_backwards** (*String* name="", *float* custom_blend=-1)

Play a given animation by the animation name in reverse.

- void **stop** (*bool* reset=true)

Stop the currently playing animation.

- void **stop_all** ( )

Stop playback of animations (deprecated).

- *bool* **is_playing** ( ) const

Return whether an animation is playing.

- void **set_current_animation** (*String* anim)

Set the current animation (even if no playback occurs). Using set_current_animation() and set_active() are similar to calling play().

- *String* **get_current_animation** ( ) const

Return the name of the animation being played.

- void **queue** (*String* name)

Queue an animation for playback once the current one is done.

- void **clear_queue** ( )

If animations are queued to play, clear them.

- void **set_active** (*bool* active)

Set the player as active (playing). If false, it

will do nothing.

- *bool* **is_active** ( ) const

Return true if the player is active.

- void **set_speed** (*float* speed)

Set a speed scaling ratio in a given animation channel (or channel 0 if none is provided). Default ratio is *1* (no scaling).

- *float* **get_speed** ( ) const

Get the speed scaling ratio in a given animation channel (or channel 0 if none is provided). Default ratio is *1* (no scaling).

- void **set_autoplay** (*String* name)

Set the name of the animation that will be automatically played when the scene is loaded.

- *String* **get_autoplay** ( ) const

Return the name of the animation that will be automatically played when the scene is loaded.

- void **set_root** (*NodePath* path)

AnimationPlayer resolves animation track paths from this node (which is relative to itself), by default root is "..", but it can be changed.

- *NodePath* **get_root** ( ) const

Return path to root node (see *set_root*).

- void **seek** (*float* pos_sec, *bool* update=false)

Seek the animation to a given position in time (in seconds). If 'update' is true, the animation will be updated too, otherwise it will be updated at process time.

- *float* **get_pos** ( ) const

Return the playback position (in seconds) in an animation channel (or channel 0 if none is provided).

- *String* **find_animation** (*Animation* animation) const

Find an animation name by resource.

- void **clear_caches** ( )

The animation player creates caches for faster access to the nodes it will animate. However, if a specific node is removed, it may not notice it, so clear_caches will force the player to search for the nodes again.

- void **set_animation_process_mode** (*int* mode)

Set the mode in which the animation player processes. By default, it processes on idle time (framerate dependent), but using fixed time works well for animating static collision bodies in 2D and 3D. See enum ANIMATION_PROCESS_*.

- *int* **get_animation_process_mode** ( ) const

Return the mode in which the animation player processes. See *set_animation_process_mode*.

- *float* **get_current_animation_pos** ( ) const

Get the position (in seconds) of the currently being played animation.

- *float* **get_current_animation_length** ( ) const

Get the length (in seconds) of the currently being played animation.

- void **advance** (*float* delta)

Used to skip ahead or skip back in an animation. Delta is the time in seconds to skip.

9.9 AnimationTreePlayer

Inherits: *Node* < *Object*

Category: Core

9.9.1 Brief Description

Animation Player that uses a node graph for the blending.

9.9.2 Member Functions

void	*add_node* (*int* type, *String* id)
bool	*node_exists* (*String* node) const
int	*node_rename* (*String* node, *String* new_name)
int	*node_get_type* (*String* id) const
int	*node_get_input_count* (*String* id) const
String	*node_get_input_source* (*String* id, *int* idx) const
void	*animation_node_set_animation* (*String* id, *Animation* animation)
Animation	*animation_node_get_animation* (*String* id) const
void	*animation_node_set_master_animation* (*String* id, *String* source)
String	*animation_node_get_master_animation* (*String* id) const
void	*oneshot_node_set_fadein_time* (*String* id, *float* time_sec)
float	*oneshot_node_get_fadein_time* (*String* id) const
void	*oneshot_node_set_fadeout_time* (*String* id, *float* time_sec)
float	*oneshot_node_get_fadeout_time* (*String* id) const
void	*oneshot_node_set_autorestart* (*String* id, *bool* enable)
void	*oneshot_node_set_autorestart_delay* (*String* id, *float* delay_sec)
void	*oneshot_node_set_autorestart_random_delay* (*String* id, *float* rand_sec)
bool	*oneshot_node_has_autorestart* (*String* id) const
float	*oneshot_node_get_autorestart_delay* (*String* id) const
float	*oneshot_node_get_autorestart_random_delay* (*String* id) const
void	*oneshot_node_start* (*String* id)
void	*oneshot_node_stop* (*String* id)
bool	*oneshot_node_is_active* (*String* id) const
void	*oneshot_node_set_filter_path* (*String* id, *NodePath* path, *bool* enable)
void	*mix_node_set_amount* (*String* id, *float* ratio)
float	*mix_node_get_amount* (*String* id) const
void	*blend2_node_set_amount* (*String* id, *float* blend)
float	*blend2_node_get_amount* (*String* id) const
void	*blend2_node_set_filter_path* (*String* id, *NodePath* path, *bool* enable)
void	*blend3_node_set_amount* (*String* id, *float* blend)
float	*blend3_node_get_amount* (*String* id) const
void	*blend4_node_set_amount* (*String* id, *Vector2* blend)
Vector2	*blend4_node_get_amount* (*String* id) const
void	*timescale_node_set_scale* (*String* id, *float* scale)
float	*timescale_node_get_scale* (*String* id) const
void	*timeseek_node_seek* (*String* id, *float* pos_sec)
void	*transition_node_set_input_count* (*String* id, *int* count)
int	*transition_node_get_input_count* (*String* id) const
void	*transition_node_delete_input* (*String* id, *int* input_idx)
void	*transition_node_set_input_auto_advance* (*String* id, *int* input_idx, *bool* enable)
bool	*transition_node_has_input_auto_advance* (*String* id, *int* input_idx) const
void	*transition_node_set_xfade_time* (*String* id, *float* time_sec)
float	*transition_node_get_xfade_time* (*String* id) const
void	*transition_node_set_current* (*String* id, *int* input_idx)

Continúa en la página siguiente

Tabla 9.4 – proviene de la página anterior

int	*transition_node_get_current* (*String* id) const
void	*node_set_pos* (*String* id, *Vector2* screen_pos)
Vector2	*node_get_pos* (*String* id) const
void	*remove_node* (*String* id)
int	*connect* (*String* id, *String* dst_id, *int* dst_input_idx)
bool	*is_connected* (*String* id, *String* dst_id, *int* dst_input_idx) const
void	*disconnect* (*String* id, *int* dst_input_idx)
void	*set_active* (*bool* enabled)
bool	*is_active* ( ) const
void	*set_base_path* (*NodePath* path)
NodePath	*get_base_path* ( ) const
void	*set_master_player* (*NodePath* nodepath)
NodePath	*get_master_player* ( ) const
StringArray	*get_node_list* ( )
void	*set_animation_process_mode* (*int* mode)
int	*get_animation_process_mode* ( ) const
void	*advance* (*float* delta)
void	*reset* ( )
void	*recompute_caches* ( )

9.9.3 Numeric Constants

- **NODE_OUTPUT = 0**
- **NODE_ANIMATION = 1**
- **NODE_ONESHOT = 2**
- **NODE_MIX = 3**
- **NODE_BLEND2 = 4**
- **NODE_BLEND3 = 5**
- **NODE_BLEND4 = 6**
- **NODE_TIMESCALE = 7**
- **NODE_TIMESEEK = 8**
- **NODE_TRANSITION = 9**

9.9.4 Description

Animation Player that uses a node graph for the blending. This kind of player is very useful when animating character or other skeleton based rigs, because it can combine several animations to form a desired pose.

9.9.5 Member Function Description

- void **add_node** (*int* type, *String* id)

Add a node of a given type in the graph with given id.

- *bool* **node_exists** (*String* node) const

Check if a node exists (by name).

- *int* **node_rename** (*String* node, *String* new_name)

Rename a node in the graph.

- *int* **node_get_type** (*String* id) const

Get the node type, will return from NODE_* enum.

- *int* **node_get_input_count** (*String* id) const

Return the input count for a given node. Different types of nodes have different amount of inputs.

- *String* **node_get_input_source** (*String* id, *int* idx) const

Return the input source for a given node input.

- void **animation_node_set_animation** (*String* id, *Animation* animation)

Set the animation for an animation node.

- *Animation* **animation_node_get_animation** (*String* id) const
- void **animation_node_set_master_animation** (*String* id, *String* source)
- *String* **animation_node_get_master_animation** (*String* id) const
- void **oneshot_node_set_fadein_time** (*String* id, *float* time_sec)
- *float* **oneshot_node_get_fadein_time** (*String* id) const
- void **oneshot_node_set_fadeout_time** (*String* id, *float* time_sec)
- *float* **oneshot_node_get_fadeout_time** (*String* id) const
- void **oneshot_node_set_autorestart** (*String* id, *bool* enable)
- void **oneshot_node_set_autorestart_delay** (*String* id, *float* delay_sec)
- void **oneshot_node_set_autorestart_random_delay** (*String* id, *float* rand_sec)
- *bool* **oneshot_node_has_autorestart** (*String* id) const
- *float* **oneshot_node_get_autorestart_delay** (*String* id) const
- *float* **oneshot_node_get_autorestart_random_delay** (*String* id) const
- void **oneshot_node_start** (*String* id)
- void **oneshot_node_stop** (*String* id)
- *bool* **oneshot_node_is_active** (*String* id) const
- void **oneshot_node_set_filter_path** (*String* id, *NodePath* path, *bool* enable)
- void **mix_node_set_amount** (*String* id, *float* ratio)
- *float* **mix_node_get_amount** (*String* id) const
- void **blend2_node_set_amount** (*String* id, *float* blend)
- *float* **blend2_node_get_amount** (*String* id) const
- void **blend2_node_set_filter_path** (*String* id, *NodePath* path, *bool* enable)
- void **blend3_node_set_amount** (*String* id, *float* blend)
- *float* **blend3_node_get_amount** (*String* id) const
- void **blend4_node_set_amount** (*String* id, *Vector2* blend)
- *Vector2* **blend4_node_get_amount** (*String* id) const

- void **timescale_node_set_scale** (*String* id, *float* scale)
- *float* **timescale_node_get_scale** (*String* id) const
- void **timeseek_node_seek** (*String* id, *float* pos_sec)
- void **transition_node_set_input_count** (*String* id, *int* count)
- *int* **transition_node_get_input_count** (*String* id) const
- void **transition_node_delete_input** (*String* id, *int* input_idx)
- void **transition_node_set_input_auto_advance** (*String* id, *int* input_idx, *bool* enable)
- *bool* **transition_node_has_input_auto_advance** (*String* id, *int* input_idx) const
- void **transition_node_set_xfade_time** (*String* id, *float* time_sec)
- *float* **transition_node_get_xfade_time** (*String* id) const
- void **transition_node_set_current** (*String* id, *int* input_idx)
- *int* **transition_node_get_current** (*String* id) const
- void **node_set_pos** (*String* id, *Vector2* screen_pos)
- *Vector2* **node_get_pos** (*String* id) const
- void **remove_node** (*String* id)
- *int* **connect** (*String* id, *String* dst_id, *int* dst_input_idx)
- *bool* **is_connected** (*String* id, *String* dst_id, *int* dst_input_idx) const
- void **disconnect** (*String* id, *int* dst_input_idx)
- void **set_active** (*bool* enabled)
- *bool* **is_active** ( ) const
- void **set_base_path** (*NodePath* path)
- *NodePath* **get_base_path** ( ) const
- void **set_master_player** (*NodePath* nodepath)
- *NodePath* **get_master_player** ( ) const
- *StringArray* **get_node_list** ( )
- void **set_animation_process_mode** (*int* mode)
- *int* **get_animation_process_mode** ( ) const
- void **advance** (*float* delta)
- void **reset** ( )
- void **recompute_caches** ( )

9.10 Area

Inherits: *CollisionObject* **<** *Spatial* **<** *Node* **<** *Object*

Category: Core

9.10.1 Brief Description

General purpose area detection and influence for 3D physics.

9.10.2 Member Functions

void	*set_space_override_mode* (*int* enable)
int	*get_space_override_mode* ( ) const
void	*set_gravity_is_point* (*bool* enable)
bool	*is_gravity_a_point* ( ) const
void	*set_gravity_distance_scale* (*float* distance_scale)
float	*get_gravity_distance_scale* ( ) const
void	*set_gravity_vector* (*Vector3* vector)
Vector3	*get_gravity_vector* ( ) const
void	*set_gravity* (*float* gravity)
float	*get_gravity* ( ) const
void	*set_angular_damp* (*float* angular_damp)
float	*get_angular_damp* ( ) const
void	*set_linear_damp* (*float* linear_damp)
float	*get_linear_damp* ( ) const
void	*set_priority* (*float* priority)
float	*get_priority* ( ) const
void	*set_monitorable* (*bool* enable)
bool	*is_monitorable* ( ) const
void	*set_enable_monitoring* (*bool* enable)
bool	*is_monitoring_enabled* ( ) const
Array	*get_overlapping_bodies* ( ) const
Array	*get_overlapping_areas* ( ) const
bool	*overlaps_body* (*Object* body) const
bool	*overlaps_area* (*Object* area) const

9.10.3 Signals

- **body_enter** (*Object* body)
- **body_enter_shape** (*int* body_id, *Object* body, *int* body_shape, *int* area_shape)
- **area_enter** (*Object* area)
- **area_enter_shape** (*int* area_id, *Object* area, *int* area_shape, *int* area_shape)
- **body_exit** (*Object* body)
- **body_exit_shape** (*int* body_id, *Object* body, *int* body_shape, *int* area_shape)
- **area_exit** (*Object* area)
- **area_exit_shape** (*int* area_id, *Object* area, *int* area_shape, *int* area_shape)

9.10.4 Description

General purpose area detection for 3D physics. Areas can be used for detection of objects that enter/exit them, as well as overriding space parameters (changing gravity, damping, etc). For this, use any space override different from AREA_SPACE_OVERRIDE_DISABLE and point gravity at the center of mass.

9.10.5 Member Function Description

- void **set_space_override_mode** (*int* enable)

Set the space override mode. This mode controls how an area affects gravity and damp.

AREA_SPACE_OVERRIDE_DISABLED: This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.

AREA_SPACE_OVERRIDE_COMBINE: This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.

AREA_SPACE_OVERRIDE_COMBINE_REPLACE: This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.

AREA_SPACE_OVERRIDE_REPLACE: This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.

AREA_SPACE_OVERRIDE_REPLACE_COMBINE: This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.

- *int* **get_space_override_mode** ( ) const

Return the space override mode.

- void **set_gravity_is_point** (*bool* enable)

When overriding space parameters, this method sets whether this area has a center of gravity. To set/get the location of the center of gravity, use *set_gravity_vector*/*get_gravity_vector*.

- *bool* **is_gravity_a_point** ( ) const

Return whether gravity is a point. A point gravity will attract objects towards it, as opposed to a gravity vector, which moves them in a given direction.

- void **set_gravity_distance_scale** (*float* distance_scale)

Set the falloff factor for point gravity. The greater this value is, the faster the strength of gravity decreases with the square of distance.

- *float* **get_gravity_distance_scale** ( ) const

Return the falloff factor for point gravity.

- void **set_gravity_vector** (*Vector3* vector)

Set the gravity vector. This vector does not have to be normalized.

If gravity is a point (see *is_gravity_a_point*), this will be the attraction center.

- *Vector3* **get_gravity_vector** ( ) const

Return the gravity vector. If gravity is a point (see *is_gravity_a_point*), this will be the attraction center.

- void **set_gravity** (*float* gravity)

Set the gravity intensity. This is useful to alter the force of gravity without altering its direction.

This value multiplies the gravity vector, whether it is the given vector (*set_gravity_vector*), or a calculated one (when using a center of gravity).

- *float* **get_gravity** ( ) const

Return the gravity intensity.

- void **set_angular_damp** (*float* angular_damp)

Set the rate at which objects stop spinning in this area, if there are not any other forces making it spin. The value is a fraction of its current speed, lost per second. Thus, a value of 1.0 should mean stopping immediately, and 0.0 means the object never stops.

In practice, as the fraction of speed lost gets smaller with each frame, a value of 1.0 does not mean the object will stop in exactly one second. Only when the physics calculations are done at 1 frame per second, it does stop in a second.

- *float* **get_angular_damp** **(** **)** const

Return the angular damp rate.

- void **set_linear_damp** **(** *float* linear_damp **)**

Set the rate at which objects stop moving in this area, if there are not any other forces moving it. The value is a fraction of its current speed, lost per second. Thus, a value of 1.0 should mean stopping immediately, and 0.0 means the object never stops.

In practice, as the fraction of speed lost gets smaller with each frame, a value of 1.0 does not mean the object will stop in exactly one second. Only when the physics calculations are done at 1 frame per second, it does stop in a second.

- *float* **get_linear_damp** **(** **)** const

Return the linear damp rate.

- void **set_priority** **(** *float* priority **)**

Set the order in which the area is processed. Greater values mean the area gets processed first. This is useful for areas which have an space override different from AREA_SPACE_OVERRIDE_DISABLED or AREA_SPACE_OVERRIDE_COMBINE, as they replace values, and are thus order-dependent.

Areas with the same priority value get evaluated in an unpredictable order, and should be differentiated if evaluation order is to be important.

- *float* **get_priority** **(** **)** const

Return the processing order of this area.

- void **set_monitorable** **(** *bool* enable **)**

Set whether this area can be detected by other, monitoring, areas. Only areas need to be marked as monitorable. Bodies are always so.

- *bool* **is_monitorable** **(** **)** const

Return whether this area can be detected by other, monitoring, areas.

- void **set_enable_monitoring** **(** *bool* enable **)**

Set whether this area can detect bodies/areas entering/exiting it.

- *bool* **is_monitoring_enabled** **(** **)** const

Return whether this area detects bodies/areas entering/exiting it.

- *Array* **get_overlapping_bodies** **(** **)** const

Return a list of the bodies (*PhysicsBody*) that are totally or partially inside this area.

- *Array* **get_overlapping_areas** **(** **)** const

Return a list of the areas that are totally or partially inside this area.

- *bool* **overlaps_body** **(** *Object* body **)** const

Return whether the body passed is totally or partially inside this area.

- *bool* **overlaps_area** **(** *Object* area **)** const

Return whether the area passed is totally or partially inside this area.

9.11 Area2D

Inherits: *CollisionObject2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.11.1 Brief Description

General purpose area detection and influence for 2D physics.

9.11.2 Member Functions

void	*set_space_override_mode* (*int* enable)
int	*get_space_override_mode* ( ) const
void	*set_gravity_is_point* (*bool* enable)
bool	*is_gravity_a_point* ( ) const
void	*set_gravity_distance_scale* (*float* distance_scale)
float	*get_gravity_distance_scale* ( ) const
void	*set_gravity_vector* (*Vector2* vector)
Vector2	*get_gravity_vector* ( ) const
void	*set_gravity* (*float* gravity)
float	*get_gravity* ( ) const
void	*set_linear_damp* (*float* linear_damp)
float	*get_linear_damp* ( ) const
void	*set_angular_damp* (*float* angular_damp)
float	*get_angular_damp* ( ) const
void	*set_priority* (*float* priority)
float	*get_priority* ( ) const
void	*set_collision_mask* (*int* collision_mask)
int	*get_collision_mask* ( ) const
void	*set_layer_mask* (*int* layer_mask)
int	*get_layer_mask* ( ) const
void	*set_collision_mask_bit* (*int* bit, *bool* value)
bool	*get_collision_mask_bit* (*int* bit) const
void	*set_layer_mask_bit* (*int* bit, *bool* value)
bool	*get_layer_mask_bit* (*int* bit) const
void	*set_enable_monitoring* (*bool* enable)
bool	*is_monitoring_enabled* ( ) const
void	*set_monitorable* (*bool* enable)
bool	*is_monitorable* ( ) const
Array	*get_overlapping_bodies* ( ) const
Array	*get_overlapping_areas* ( ) const
bool	*overlaps_body* (*Object* body) const
bool	*overlaps_area* (*Object* area) const

9.11.3 Signals

- **body_enter** (*Object* body)
- **body_enter_shape** (*int* body_id, *Object* body, *int* body_shape, *int* area_shape)
- **area_enter** (*Object* area)
- **area_enter_shape** (*int* area_id, *Object* area, *int* area_shape, *int* area_shape)
- **body_exit** (*Object* body)
- **body_exit_shape** (*int* body_id, *Object* body, *int* body_shape, *int* area_shape)
- **area_exit** (*Object* area)
- **area_exit_shape** (*int* area_id, *Object* area, *int* area_shape, *int* area_shape)

9.11.4 Description

General purpose area detection for 2D physics. Areas can be used for detection of objects that enter/exit them, as well as overriding space parameters (changing gravity, damping, etc). For this, use any space override different from AREA_SPACE_OVERRIDE_DISABLE and point gravity at the center of mass.

9.11.5 Member Function Description

- void **set_space_override_mode** (*int* enable)

Set the space override mode. This mode controls how an area affects gravity and damp.

AREA_SPACE_OVERRIDE_DISABLED: This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.

AREA_SPACE_OVERRIDE_COMBINE: This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.

AREA_SPACE_OVERRIDE_COMBINE_REPLACE: This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.

AREA_SPACE_OVERRIDE_REPLACE: This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.

AREA_SPACE_OVERRIDE_REPLACE_COMBINE: This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.

- *int* **get_space_override_mode** ( ) const

Return the space override mode.

- void **set_gravity_is_point** (*bool* enable)

When overriding space parameters, this method sets whether this area has a center of gravity. To set/get the location of the center of gravity, use *set_gravity_vector*/*get_gravity_vector*.

- *bool* **is_gravity_a_point** ( ) const

Return whether gravity is a point. A point gravity will attract objects towards it, as opposed to a gravity vector, which moves them in a given direction.

- void **set_gravity_distance_scale** (*float* distance_scale)

Set the falloff factor for point gravity. The greater this value is, the faster the strength of gravity decreases with the square of distance.

- *float* **get_gravity_distance_scale ( )** const

Return the falloff factor for point gravity.

- void **set_gravity_vector (** *Vector2* vector **)**

Set the gravity vector. This vector does not have to be normalized.

If gravity is a point (see *is_gravity_a_point*), this will be the attraction center.

- *Vector2* **get_gravity_vector ( )** const

Return the gravity vector. If gravity is a point (see *is_gravity_a_point*), this will be the attraction center.

- void **set_gravity (** *float* gravity **)**

Set the gravity intensity. This is useful to alter the force of gravity without altering its direction.

This value multiplies the gravity vector, whether it is the given vector (*set_gravity_vector*), or a calculated one (when using a center of gravity).

- *float* **get_gravity ( )** const

Return the gravity intensity.

- void **set_linear_damp (** *float* linear_damp **)**

Set the rate at which objects stop moving in this area, if there are not any other forces moving it. The value is a fraction of its current speed, lost per second. Thus, a value of 1.0 should mean stopping immediately, and 0.0 means the object never stops.

In practice, as the fraction of speed lost gets smaller with each frame, a value of 1.0 does not mean the object will stop in exactly one second. Only when the physics calculations are done at 1 frame per second, it does stop in a second.

- *float* **get_linear_damp ( )** const

Return the linear damp rate.

- void **set_angular_damp (** *float* angular_damp **)**

Set the rate at which objects stop spinning in this area, if there are not any other forces making it spin. The value is a fraction of its current speed, lost per second. Thus, a value of 1.0 should mean stopping immediately, and 0.0 means the object never stops.

In practice, as the fraction of speed lost gets smaller with each frame, a value of 1.0 does not mean the object will stop in exactly one second. Only when the physics calculations are done at 1 frame per second, it does stop in a second.

- *float* **get_angular_damp ( )** const

Return the angular damp rate.

- void **set_priority (** *float* priority **)**

Set the order in which the area is processed. Greater values mean the area gets processed first. This is useful for areas which have an space override different from AREA_SPACE_OVERRIDE_DISABLED or AREA_SPACE_OVERRIDE_COMBINE, as they replace values, and are thus order-dependent.

Areas with the same priority value get evaluated in an unpredictable order, and should be differentiated if evaluation order is to be important.

- *float* **get_priority ( )** const

Return the processing order of this area.

- void **set_collision_mask (** *int* collision_mask **)**

Set the physics layers this area can scan for collisions.

- *int* **get_collision_mask** **(** **)** const

Return the physics layers this area can scan for collisions.

- void **set_layer_mask** **(** *int* layer_mask **)**

Set the physics layers this area is in.

Collidable objects can exist in any of 32 different layers. These layers are not visual, but more of a tagging system instead. A collidable can use these layers/tags to select with which objects it can collide, using *set_collision_mask*.

A contact is detected if object A is in any of the layers that object B scans, or object B is in any layer scanned by object A.

- *int* **get_layer_mask** **(** **)** const

Return the physics layer this area is in.

- void **set_collision_mask_bit** **(** *int* bit, *bool* value **)**

Set/clear individual bits on the collision mask. This makes selecting the areas scanned easier.

- *bool* **get_collision_mask_bit** **(** *int* bit **)** const

Return an individual bit on the collision mask.

- void **set_layer_mask_bit** **(** *int* bit, *bool* value **)**

Set/clear individual bits on the layer mask. This makes getting an area in/out of only one layer easier.

- *bool* **get_layer_mask_bit** **(** *int* bit **)** const

Return an individual bit on the layer mask.

- void **set_enable_monitoring** **(** *bool* enable **)**

Set whether this area can detect bodies/areas entering/exiting it.

- *bool* **is_monitoring_enabled** **(** **)** const

Return whether this area detects bodies/areas entering/exiting it.

- void **set_monitorable** **(** *bool* enable **)**

Set whether this area can be detected by other, monitoring, areas. Only areas need to be marked as monitorable. Bodies are always so.

- *bool* **is_monitorable** **(** **)** const

Return whether this area can be detected by other, monitoring, areas.

- *Array* **get_overlapping_bodies** **(** **)** const

Return a list of the bodies (*PhysicsBody2D*) that are totally or partially inside this area.

- *Array* **get_overlapping_areas** **(** **)** const

Return a list of the areas that are totally or partially inside this area.

- *bool* **overlaps_body** **(** *Object* body **)** const

Return whether the body passed is totally or partially inside this area.

- *bool* **overlaps_area** **(** *Object* area **)** const

Return whether the area passed is totally or partially inside this area.

9.12 Array

Category: Built-In Types

9.12.1 Brief Description

Generic array datatype.

9.12.2 Member Functions

void	*append* (var value)
void	*clear* ( )
bool	*empty* ( )
void	*erase* (var value)
int	*find* (var value)
int	*hash* ( )
void	*insert* (*int* pos, var value)
void	*invert* ( )
bool	*is_shared* ( )
void	*pop_back* ( )
void	*pop_front* ( )
void	*push_back* (var value)
void	*push_front* (var value)
void	*remove* (*int* pos)
void	*resize* (*int* pos)
int	*size* ( )
void	*sort* ( )
void	*sort_custom* (*Object* obj, *String* func)
Array	Array (*RawArray* from)
Array	Array (*IntArray* from)
Array	Array (*RealArray* from)
Array	Array (*StringArray* from)
Array	Array (*Vector2Array* from)
Array	Array (*Vector3Array* from)
Array	Array (*ColorArray* from)

9.12.3 Description

Generic array, contains several elements of any type, accessible by numerical index starting at 0. Arrays are always passed by reference.

9.12.4 Member Function Description

- void **append** (var value)

Append an element at the end of the array (alias of *push_back*).

- void **clear** ( )

Clear the array (resize to 0).

- *bool* **empty ( )**

Return true if the array is empty (size==0).

- void **erase** (var value)

Remove the first occurrence of a value from the array.

- *int* **find** (var value)

Searches the array for a value and returns its index or -1 if not found.

- *int* **hash ( )**

Return a hashed integer value representing the array contents.

- void **insert** (*int* pos, var value)

Insert a new element at a given position in the array. The position must be valid, or at the end of the array (pos==size()).

- void **invert ( )**

Reverse the order of the elements in the array (so first element will now be the last).

- *bool* **is_shared ( )**

Get whether this is a shared array instance.

- void **pop_back ( )**

Remove the last element of the array.

- void **pop_front ( )**

Remove the first element of the array.

- void **push_back** (var value)

Append an element at the end of the array.

- void **push_front** (var value)

Add an element at the beginning of the array.

- void **remove** (*int* pos)

Remove an element from the array by index.

- void **resize** (*int* pos)

Resize the array to contain a different number of elements. If the array size is smaller, elements are cleared, if bigger, new elements are Null.

- *int* **size ( )**

Return the amount of elements in the array.

- void **sort ( )**

Sort the array using natural order.

- void **sort_custom** (*Object* obj, *String* func)

Sort the array using a custom method. The arguments are an object that holds the method and the name of such method. The custom method receives two arguments (a pair of elements from the array) and must return true if the first argument is less than the second, and return false otherwise.

- *Array* **Array** (*RawArray* from)

Construct an array from a *RawArray*.

- *Array* **Array** (*IntArray* from)

Construct an array from a *RawArray*.

- *Array* **Array** (*RealArray* from)

Construct an array from a *RawArray*.

- *Array* **Array** (*StringArray* from)

Construct an array from a *RawArray*.

- *Array* **Array** (*Vector2Array* from)

Construct an array from a *RawArray*.

- *Array* **Array** (*Vector3Array* from)

Construct an array from a *RawArray*.

- *Array* **Array** (*ColorArray* from)

Construct an array from a *RawArray*.

9.13 AtlasTexture

Inherits: *Texture* < *Resource* < *Reference* < *Object*

Category: Core

9.13.1 Brief Description

9.13.2 Member Functions

void	*set_atlas* (*Texture* atlas)
Texture	*get_atlas* ( ) const
void	*set_region* (*Rect2* region)
Rect2	*get_region* ( ) const
void	*set_margin* (*Rect2* margin)
Rect2	*get_margin* ( ) const

9.13.3 Member Function Description

- void **set_atlas** (*Texture* atlas)
- *Texture* **get_atlas** ( ) const
- void **set_region** (*Rect2* region)
- *Rect2* **get_region** ( ) const
- void **set_margin** (*Rect2* margin)
- *Rect2* **get_margin** ( ) const

9.14 AudioServer

Inherits: *Object*

Inherited By: *AudioServerSW*

Category: Core

9.14.1 Brief Description

Server interface for low level audio access.

9.14.2 Member Functions

RID	*sample_create* (*int* format, *bool* stereo, *int* length)
void	*sample_set_description* (*RID* sample, *String* description)
String	*sample_get_description* (*RID* sample) const
int	*sample_get_format* (*RID* sample) const
bool	*sample_is_stereo* (*RID* sample) const
int	*sample_get_length* (*RID* sample) const
void	*sample_set_signed_data* (*RID* sample, *RealArray* data)
void	*sample_set_data* (*RID* sample, *RawArray* data)
RawArray	*sample_get_data* (*RID* sample) const
void	*sample_set_mix_rate* (*RID* sample, *int* mix_rate)
int	*sample_get_mix_rate* (*RID* sample) const
void	*sample_set_loop_format* (*RID* sample, *int* loop_format)
int	*sample_get_loop_format* (*RID* sample) const
void	*sample_set_loop_begin* (*RID* sample, *int* pos)
int	*sample_get_loop_begin* (*RID* sample) const
void	*sample_set_loop_end* (*RID* sample, *int* pos)
int	*sample_get_loop_end* (*RID* sample) const
RID	*voice_create* ( )
void	*voice_play* (*RID* voice, *RID* sample)
void	*voice_set_volume* (*RID* voice, *float* volume)
void	*voice_set_pan* (*RID* voice, *float* pan, *float* depth=0, *float* height=0)
void	*voice_set_filter* (*RID* voice, *int* type, *float* cutoff, *float* resonance, *float* gain=0)
void	*voice_set_chorus* (*RID* voice, *float* chorus)
void	*voice_set_reverb* (*RID* voice, *int* room, *float* reverb)
void	*voice_set_mix_rate* (*RID* voice, *int* rate)
void	*voice_set_positional* (*RID* voice, *bool* enabled)
float	*voice_get_volume* (*RID* voice) const
float	*voice_get_pan* (*RID* voice) const
float	*voice_get_pan_height* (*RID* voice) const
float	*voice_get_pan_depth* (*RID* voice) const
int	*voice_get_filter_type* (*RID* voice) const
float	*voice_get_filter_cutoff* (*RID* voice) const
float	*voice_get_filter_resonance* (*RID* voice) const
float	*voice_get_chorus* (*RID* voice) const
int	*voice_get_reverb_type* (*RID* voice) const
float	*voice_get_reverb* (*RID* voice) const

Continúa en la página siguiente

Tabla 9.6 – proviene de la página anterior

int	*voice_get_mix_rate* (*RID* voice) const
bool	*voice_is_positional* (*RID* voice) const
void	*voice_stop* (*RID* voice)
void	*free_rid* (*RID* rid)
void	*set_stream_global_volume_scale* (*float* scale)
float	*get_stream_global_volume_scale* ( ) const
void	*set_fx_global_volume_scale* (*float* scale)
float	*get_fx_global_volume_scale* ( ) const
void	*set_event_voice_global_volume_scale* (*float* scale)
float	*get_event_voice_global_volume_scale* ( ) const

9.14.3 Numeric Constants

- **SAMPLE_FORMAT_PCM8 = 0** — Sample format is 8 bits, signed.
- **SAMPLE_FORMAT_PCM16 = 1** — Sample format is 16 bits, little-endian, signed.
- **SAMPLE_FORMAT_IMA_ADPCM = 2** — Sample format is IMA-ADPCM compressed.
- **SAMPLE_LOOP_NONE = 0** — Sample does not loop.
- **SAMPLE_LOOP_FORWARD = 1** — Sample loops in forward mode.
- **SAMPLE_LOOP_PING_PONG = 2** — Sample loops in a bidirectional way.
- **FILTER_NONE = 0** — Filter is disabled.
- **FILTER_LOWPASS = 1** — Filter is a resonant lowpass.
- **FILTER_BANDPASS = 2** — Filter is a resonant bandpass.
- **FILTER_HIPASS = 3** — Filter is a resonant highpass.
- **FILTER_NOTCH = 4** — Filter is a notch (band reject).
- **FILTER_BANDLIMIT = 6** — Filter is a bandlimit (resonance used as highpass).
- **REVERB_SMALL = 0** — Small reverb room (closet, bathroom, etc).
- **REVERB_MEDIUM = 1** — Medium reverb room (living room)
- **REVERB_LARGE = 2** — Large reverb room (warehouse).
- **REVERB_HALL = 3** — Large reverb room with long decay.

9.14.4 Description

AudioServer is a low level server interface for audio access. It is in charge of creating sample data (playable audio) as well as its playback via a voice interface.

9.14.5 Member Function Description

- *RID* **sample_create** (*int* format, *bool* stereo, *int* length)

Create an audio sample, return a *RID* referencing it. The sample will be created with a given format (from the SAMPLE_FORMAT_* enum), a total length (in samples, not bytes), in either stereo or mono.

Even if a stereo sample consists of a left sample and a right sample, it still counts as one sample for length purposes.

- void **sample_set_description** (*RID* sample, *String* description)

Set the description of an audio sample. Mainly used for organization.

- *String* **sample_get_description** (*RID* sample) const

Return the description of an audio sample. Mainly used for organization.

- *int* **sample_get_format** (*RID* sample) const

Return the format of the audio sample, in the form of the SAMPLE_FORMAT_* enum.

- *bool* **sample_is_stereo** (*RID* sample) const

Return whether the sample is stereo (2 channels).

- *int* **sample_get_length** (*RID* sample) const

Return the length in samples (not bytes) of the audio sample. Even if a stereo sample consists of a left sample and a right sample, it still counts as one sample for length purposes.

- void **sample_set_signed_data** (*RID* sample, *RealArray* data)

Set the sample data for a given sample as an array of floats. The length must be equal to the sample length or an error will be produced.

For this method, a stereo sample is made from two samples. Thus, in case of a stereo sample, the array length must be twice the length returned by *sample_get_length*.

Trying to alter a SAMPLE_FORMAT_IMA_ADPCM sample is not supported. It will throw an error to the console, but will not alter the sample data.

- void **sample_set_data** (*RID* sample, *RawArray* data)

Set the sample data for a given sample as an array of bytes. The length must be equal to the sample length expected in bytes or an error will be produced. The byte length can be calculated as follows:

Get the sample length (*sample_get_length*).

If the sample format is SAMPLE_FORMAT_PCM16, multiply it by 2.

If the sample format is SAMPLE_FORMAT_IMA_ADPCM, divide it by 2 (rounding any fraction up), then add 4.

If the sample is stereo (*sample_is_stereo*), multiply it by 2.

- *RawArray* **sample_get_data** (*RID* sample) const

Return the sample data as an array of bytes. The length will be the expected length in bytes.

- void **sample_set_mix_rate** (*RID* sample, *int* mix_rate)

Change the default mix rate of a given sample.

- *int* **sample_get_mix_rate** (*RID* sample) const

Return the mix rate of the given sample.

- void **sample_set_loop_format** (*RID* sample, *int* loop_format)

Set the loop format for a sample from the SAMPLE_LOOP_* enum. As a warning, Ping Pong loops may not be available on some hardware-mixing platforms.

- *int* **sample_get_loop_format** (*RID* sample) const

Return the loop format for a sample, as a value from the SAMPLE_LOOP_* enum.

- void **sample_set_loop_begin** (*RID* sample, *int* pos)

Set the initial loop point of a sample. Only has effect if sample loop is enabled. See *sample_set_loop_format*.

- *int* **sample_get_loop_begin** (*RID* sample) const

Return the initial loop point of a sample. Only has effect if sample loop is enabled. See *sample_set_loop_format*.

- void **sample_set_loop_end** (*RID* sample, *int* pos)

Set the final loop point of a sample. Only has effect if sample loop is enabled. See *sample_set_loop_format*.

- *int* **sample_get_loop_end** (*RID* sample) const

Return the final loop point of a sample. Only has effect if sample loop is enabled. See *sample_set_loop_format*.

- *RID* **voice_create** ( )

Allocate a voice for playback. Voices are persistent. A voice can play a single sample at the same time. See *sample_create*.

- void **voice_play** (*RID* voice, *RID* sample)

Start playback of a given voice using a given sample. If the voice was already playing it will be restarted.

- void **voice_set_volume** (*RID* voice, *float* volume)

Change the volume of a currently playing voice. Volume is expressed as linear gain where 0.0 is mute and 1.0 is default.

- void **voice_set_pan** (*RID* voice, *float* pan, *float* depth=0, *float* height=0)

Change the pan of a currently playing voice and, optionally, the depth and height for a positional/3D sound. Panning values are expressed within the -1 to +1 range.

- void **voice_set_filter** (*RID* voice, *int* type, *float* cutoff, *float* resonance, *float* gain=0)

Set a resonant filter post processing for the voice. Filter type is a value from the FILTER_* enum.

- void **voice_set_chorus** (*RID* voice, *float* chorus)

Set chorus send post processing for the voice (from 0 to 1).

- void **voice_set_reverb** (*RID* voice, *int* room, *float* reverb)

Set the reverb send post processing for the voice (from 0 to 1) and the reverb type, from the REVERB_* enum.

- void **voice_set_mix_rate** (*RID* voice, *int* rate)

Set a different playback mix rate for the given voice.

- void **voice_set_positional** (*RID* voice, *bool* enabled)

Set whether a given voice is positional. This is only interpreted as a hint and used for backends that may support binaural encoding.

- *float* **voice_get_volume** (*RID* voice) const

Return the current volume for a given voice.

- *float* **voice_get_pan** (*RID* voice) const

Return the current pan for a given voice (-1 to +1 range).

- *float* **voice_get_pan_height** (*RID* voice) const

Return the current pan height for a given voice (-1 to +1 range).

- *float* **voice_get_pan_depth** (*RID* voice) const

Return the current pan depth for a given voice (-1 to +1 range).

- *int* **voice_get_filter_type** (*RID* voice) const

Return the current selected filter type for a given voice, from the FILTER_* enum.

- *float* **voice_get_filter_cutoff** (*RID* voice) const

Return the current filter cutoff (in hz) for a given voice.

- *float* **voice_get_filter_resonance** (*RID* voice) const

Return the current filter resonance for a given voice.

- *float* **voice_get_chorus** (*RID* voice) const

Return the current chorus send for a given voice (0 to 1).

- *int* **voice_get_reverb_type** (*RID* voice) const

Return the current reverb type for a given voice from the REVERB_* enum.

- *float* **voice_get_reverb** (*RID* voice) const

Return the current reverb send for a given voice (0 to 1).

- *int* **voice_get_mix_rate** (*RID* voice) const

Return the current mix rate for a given voice.

- *bool* **voice_is_positional** (*RID* voice) const

Return whether the current voice is positional. See *voice_set_positional*.

- void **voice_stop** (*RID* voice)

Stop a given voice.

- void **free_rid** (*RID* rid)

Free a *RID* resource.

- void **set_stream_global_volume_scale** (*float* scale)

Set global scale for stream playback. Default is 1.0.

- *float* **get_stream_global_volume_scale** ( ) const

Return the global scale for stream playback.

- void **set_fx_global_volume_scale** (*float* scale)

Set global scale for all voices (not including streams). Default is 1.0.

- *float* **get_fx_global_volume_scale** ( ) const

Return the global scale for all voices.

- void **set_event_voice_global_volume_scale** (*float* scale)

Set global scale for event-based stream (*EventStream*) playback. Default is 1.0.

- *float* **get_event_voice_global_volume_scale** ( ) const

Return the global scale for event-based stream playback.

9.15 AudioServerSW

Inherits: *AudioServer* < *Object*

Category: Core

9.15.1 Brief Description

Software implementation of *AudioServer*.

9.15.2 Description

This is a software audio server. It does not use any kind of hardware acceleration.

This class does not expose any new method.

9.16 AudioStream

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *AudioStreamSpeex*, *AudioStreamMPC*, *AudioStreamOGGVorbis*, *AudioStreamOpus*

Category: Core

9.16.1 Brief Description

Base class for audio streams.

9.16.2 Description

Base class for audio streams. Audio streams are used for music playback, or other types of streamed sounds that don't fit or require more flexibility than a *Sample*.

9.17 AudioStreamMPC

Inherits: *AudioStream* < *Resource* < *Reference* < *Object*

Category: Core

9.17.1 Brief Description

MusePack audio stream driver.

9.17.2 Description

MusePack audio stream driver.

9.18 AudioStreamOGGVorbis

Inherits: *AudioStream* < *Resource* < *Reference* < *Object*

Category: Core

9.18.1 Brief Description

OGG Vorbis audio stream driver.

9.18.2 Description

OGG Vorbis audio stream driver.

9.19 AudioStreamOpus

Inherits: *AudioStream* < *Resource* < *Reference* < *Object*

Category: Core

9.19.1 Brief Description

Opus Codec audio stream driver.

9.19.2 Description

Opus Codec audio stream driver.

9.20 AudioStreamPlayback

Inherits: *Reference* < *Object*

Category: Core

9.20.1 Brief Description

9.20.2 Member Functions

void	*play* (*float* from_pos_sec=0)
void	*stop* ( )
bool	*is_playing* ( ) const
void	*set_loop* (*bool* enabled)
bool	*has_loop* ( ) const
int	*get_loop_count* ( ) const
void	*seek_pos* (*float* pos)
float	*get_pos* ( ) const
float	*get_length* ( ) const
int	*get_channels* ( ) const
int	*get_mix_rate* ( ) const
int	*get_minimum_buffer_size* ( ) const

9.20.3 Member Function Description

- void **play** (*float* from_pos_sec=0)
- void **stop** ( )
- *bool* **is_playing** ( ) const
- void **set_loop** (*bool* enabled)
- *bool* **has_loop** ( ) const
- *int* **get_loop_count** ( ) const
- void **seek_pos** (*float* pos)
- *float* **get_pos** ( ) const
- *float* **get_length** ( ) const
- *int* **get_channels** ( ) const
- *int* **get_mix_rate** ( ) const
- *int* **get_minimum_buffer_size** ( ) const

9.21 AudioStreamSpeex

Inherits: *AudioStream* < *Resource* < *Reference* < *Object*

Category: Core

9.21.1 Brief Description

Speex audio stream driver.

9.21.2 Description

Speex audio stream driver. Speex is very useful for compressed speech. It allows loading a very large amount of speech in memory at little IO/latency cost.

9.22 BackBufferCopy

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.22.1 Brief Description

Copies a region of the screen (or the whole screen) to a buffer so it can be accessed with the texscreen() shader instruction.

9.22.2 Member Functions

void	*set_rect* (*Rect2* rect)
Rect2	*get_rect* ( ) const
void	*set_copy_mode* (*int* copy_mode)
int	*get_copy_mode* ( ) const

9.22.3 Numeric Constants

- **COPY_MODE_DISABLED = 0** — Disables the buffering mode. This means the BackBufferCopy node will directly use the portion of screen it covers.
- **COPY_MODE_RECT = 1** — Sets the copy mode to a region.
- **COPY_MODE_VIEWPORT = 2** — Sets the copy mode to the entire screen.

9.22.4 Description

Node for back-buffering the currently displayed screen. The region defined in the BackBufferCopy node is bufferized with the content of the screen it covers, or the entire screen according to the copy mode set. Accessing this buffer is done with the texscreen() shader instruction.

9.22.5 Member Function Description

- void **set_rect** (*Rect2* rect)

Defines the area covered by the BackBufferCopy.

- *Rect2* **get_rect** ( ) const

Return the area covered by the BackBufferCopy.

- void **set_copy_mode** (*int* copy_mode)

Set the copy mode of the BackBufferCopy (refer to constants section).

- *int* **get_copy_mode** ( ) const

Return the copy mode currently applied to the BackBufferCopy (refer to constants section).

9.23 BakedLight

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.23.1 Brief Description

9.23.2 Member Functions

void	*set_mode* (*int* mode)

Continúa en la página siguiente

Tabla 9.7 – proviene de la página anterior

int	*get_mode* ( ) const
void	*set_octree* (*RawArray* octree)
RawArray	*get_octree* ( ) const
void	*set_light* (*RawArray* light)
RawArray	*get_light* ( ) const
void	*set_sampler_octree* (*IntArray* sampler_octree)
IntArray	*get_sampler_octree* ( ) const
void	*add_lightmap* (*Texture* texture, *Vector2* gen_size)
void	*erase_lightmap* (*int* id)
void	*clear_lightmaps* ( )
void	*set_cell_subdivision* (*int* cell_subdivision)
int	*get_cell_subdivision* ( ) const
void	*set_initial_lattice_subdiv* (*int* cell_subdivision)
int	*get_initial_lattice_subdiv* ( ) const
void	*set_plot_size* (*float* plot_size)
float	*get_plot_size* ( ) const
void	*set_bounces* (*int* bounces)
int	*get_bounces* ( ) const
void	*set_cell_extra_margin* (*float* cell_extra_margin)
float	*get_cell_extra_margin* ( ) const
void	*set_edge_damp* (*float* edge_damp)
float	*get_edge_damp* ( ) const
void	*set_normal_damp* (*float* normal_damp)
float	*get_normal_damp* ( ) const
void	*set_tint* (*float* tint)
float	*get_tint* ( ) const
void	*set_saturation* (*float* saturation)
float	*get_saturation* ( ) const
void	*set_ao_radius* (*float* ao_radius)
float	*get_ao_radius* ( ) const
void	*set_ao_strength* (*float* ao_strength)
float	*get_ao_strength* ( ) const
void	*set_format* (*int* format)
int	*get_format* ( ) const
void	*set_transfer_lightmaps_only_to_uv2* (*bool* enable)
bool	*get_transfer_lightmaps_only_to_uv2* ( ) const
void	*set_energy_multiplier* (*float* energy_multiplier)
float	*get_energy_multiplier* ( ) const
void	*set_gamma_adjust* (*float* gamma_adjust)
float	*get_gamma_adjust* ( ) const
void	*set_bake_flag* (*int* flag, *bool* enabled)
bool	*get_bake_flag* (*int* flag) const

9.23.3 Numeric Constants

- **MODE_OCTREE = 0**
- **MODE_LIGHTMAPS = 1**
- **BAKE_DIFFUSE = 0**
- **BAKE_SPECULAR = 1**

- **BAKE_TRANSLUCENT = 2**
- **BAKE_CONSERVE_ENERGY = 3**
- **BAKE_MAX = 5**

9.23.4 Member Function Description

- void **set_mode** (*int* mode)
- *int* **get_mode** ( ) const
- void **set_octree** (*RawArray* octree)
- *RawArray* **get_octree** ( ) const
- void **set_light** (*RawArray* light)
- *RawArray* **get_light** ( ) const
- void **set_sampler_octree** (*IntArray* sampler_octree)
- *IntArray* **get_sampler_octree** ( ) const
- void **add_lightmap** (*Texture* texture, *Vector2* gen_size)
- void **erase_lightmap** (*int* id)
- void **clear_lightmaps** ( )
- void **set_cell_subdivision** (*int* cell_subdivision)
- *int* **get_cell_subdivision** ( ) const
- void **set_initial_lattice_subdiv** (*int* cell_subdivision)
- *int* **get_initial_lattice_subdiv** ( ) const
- void **set_plot_size** (*float* plot_size)
- *float* **get_plot_size** ( ) const
- void **set_bounces** (*int* bounces)
- *int* **get_bounces** ( ) const
- void **set_cell_extra_margin** (*float* cell_extra_margin)
- *float* **get_cell_extra_margin** ( ) const
- void **set_edge_damp** (*float* edge_damp)
- *float* **get_edge_damp** ( ) const
- void **set_normal_damp** (*float* normal_damp)
- *float* **get_normal_damp** ( ) const
- void **set_tint** (*float* tint)
- *float* **get_tint** ( ) const
- void **set_saturation** (*float* saturation)
- *float* **get_saturation** ( ) const
- void **set_ao_radius** (*float* ao_radius)
- *float* **get_ao_radius** ( ) const

- void **set_ao_strength** (*float* ao_strength)
- *float* **get_ao_strength** ( ) const
- void **set_format** (*int* format)
- *int* **get_format** ( ) const
- void **set_transfer_lightmaps_only_to_uv2** (*bool* enable)
- *bool* **get_transfer_lightmaps_only_to_uv2** ( ) const
- void **set_energy_multiplier** (*float* energy_multiplier)
- *float* **get_energy_multiplier** ( ) const
- void **set_gamma_adjust** (*float* gamma_adjust)
- *float* **get_gamma_adjust** ( ) const
- void **set_bake_flag** (*int* flag, *bool* enabled)
- *bool* **get_bake_flag** (*int* flag) const

9.24 BakedLightInstance

Inherits: *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.24.1 Brief Description

9.24.2 Member Functions

void	*set_baked_light* (*Object* baked_light)
Object	*get_baked_light* ( ) const
RID	*get_baked_light_instance* ( ) const

9.24.3 Signals

- **baked_light_changed** ( )

9.24.4 Member Function Description

- void **set_baked_light** (*Object* baked_light)
- *Object* **get_baked_light** ( ) const
- *RID* **get_baked_light_instance** ( ) const

9.25 BakedLightSampler

Inherits: *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.25.1 Brief Description

9.25.2 Member Functions

void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const
void	*set_resolution* (*int* resolution)
int	*get_resolution* ( ) const

9.25.3 Numeric Constants

- **PARAM_RADIUS = 0**
- **PARAM_STRENGTH = 1**
- **PARAM_ATTENUATION = 2**
- **PARAM_DETAIL_RATIO = 3**
- **PARAM_MAX = 4**

9.25.4 Member Function Description

- void **set_param** (*int* param, *float* value)
- *float* **get_param** (*int* param) const
- void **set_resolution** (*int* resolution)
- *int* **get_resolution** ( ) const

9.26 BaseButton

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *TextureButton*, *Button*

Category: Core

9.26.1 Brief Description

Provides a base class for different kinds of buttons.

9.26.2 Member Functions

void	*_pressed* ( ) virtual
void	*_toggled* (*bool* pressed) virtual
void	*set_pressed* (*bool* pressed)
bool	*is_pressed* ( ) const
bool	*is_hovered* ( ) const
void	*set_toggle_mode* (*bool* enabled)
bool	*is_toggle_mode* ( ) const
void	*set_disabled* (*bool* disabled)
bool	*is_disabled* ( ) const
void	*set_click_on_press* (*bool* enable)
bool	*get_click_on_press* ( ) const
int	*get_draw_mode* ( ) const

9.26.3 Signals

- **released** ( )
- **toggled** (*bool* pressed)
- **pressed** ( )

9.26.4 Numeric Constants

- **DRAW_NORMAL = 0** — The normal state (i.e. not pressed, not hovered, not toggled and enabled) of buttons.
- **DRAW_PRESSED = 1** — The state of buttons are pressed.
- **DRAW_HOVER = 2** — The state of buttons are hovered.
- **DRAW_DISABLED = 3** — The state of buttons are disabled.

9.26.5 Description

BaseButton is the abstract base class for buttons, so it shouldn't be used directly (It doesn't display anything). Other types of buttons inherit from it.

9.26.6 Member Function Description

- void **_pressed** ( ) virtual

Called when button is pressed.

- void **_toggled** (*bool* pressed) virtual

Called when button is toggled (only if toggle_mode is active).

- void **set_pressed** (*bool* pressed)

Set the button to pressed state (only if toggle_mode is active).

- *bool* **is_pressed** ( ) const

If toggle_mode is active, return whether the button is toggled. If toggle_mode is not active, return whether the button is pressed down.

- *bool* **is_hovered** ( ) const

Return true if mouse entered the button before it exit.

- void **set_toggle_mode** (*bool* enabled)

Set the button toggle_mode property. Toggle mode makes the button flip state between pressed and unpressed each time its area is clicked.

- *bool* **is_toggle_mode** ( ) const

Return the toggle_mode property (see *set_toggle_mode*).

- void **set_disabled** (*bool* disabled)

Set the button into disabled state. When a button is disabled, it can't be clicked or toggled.

- *bool* **is_disabled** ( ) const

Return whether the button is in disabled state (see *set_disabled*).

- void **set_click_on_press** (*bool* enable)

Set the button click_on_press mode. This mode generates click events when a mouse button or key is just pressed (by default events are generated when the button/keys are released and both press and release occur in the visual area of the Button).

- *bool* **get_click_on_press** ( ) const

Return the state of the click_on_press property (see *set_click_on_press*).

- *int* **get_draw_mode** ( ) const

Return the visual state used to draw the button. This is useful mainly when implementing your own draw code by either overriding _draw() or connecting to "draw" signal. The visual state of the button is defined by the DRAW_* enum.

9.27 BitMap

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.27.1 Brief Description

9.27.2 Member Functions

void	*create* (*Vector2* size)
void	*create_from_image_alpha* (*Image* image)
void	*set_bit* (*Vector2* pos, *bool* bit)
bool	*get_bit* (*Vector2* pos) const
void	*set_bit_rect* (*Rect2* p_rect, *bool* bit)
int	*get_true_bit_count* ( ) const
Vector2	*get_size* ( ) const

9.27.3 Member Function Description

- void **create** (*Vector2* size)
- void **create_from_image_alpha** (*Image* image)
- void **set_bit** (*Vector2* pos, *bool* bit)
- *bool* **get_bit** (*Vector2* pos) const
- void **set_bit_rect** (*Rect2* p_rect, *bool* bit)
- *int* **get_true_bit_count** ( ) const
- *Vector2* **get_size** ( ) const

9.28 BoneAttachment

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.28.1 Brief Description

9.29 bool

Category: Built-In Types

9.29.1 Brief Description

Boolean built-in type

9.29.2 Member Functions

bool	bool (*int* from)
bool	bool (*float* from)
bool	bool (*String* from)

9.29.3 Description

Boolean built-in type.

9.29.4 Member Function Description

- *bool* **bool** (*int* from)

Cast an *int* value to a boolean value, this method will return true if called with an integer value different to 0 and false in other case.

- *bool* **bool** (*float* from)

Cast a *float* value to a boolean value, this method will return true if called with a floating point value different to 0 and false in other case.

- *bool* **bool (***String* from **)**

Cast a *String* value to a boolean value, this method will return true if called with a non empty string and false in other case. Examples: `bool('False')` returns true, `bool('')`. returns false

9.30 BoxContainer

Inherits: *Container* **<** *Control* **<** *CanvasItem* **<** *Node* **<** *Object*

Inherited By: *VBoxContainer*, *ButtonGroup*, *HBoxContainer*, *ColorPicker*

Category: Core

9.30.1 Brief Description

Base class for Box containers.

9.30.2 Member Functions

void	*add_spacer* **(** *bool* begin **)**
int	*get_alignment* **( )** const
void	*set_alignment* **(** *int* alignment **)**

9.30.3 Numeric Constants

- **ALIGN_BEGIN = 0** — Align children with beginning of the container.
- **ALIGN_CENTER = 1** — Align children with center of the container.
- **ALIGN_END = 2** — Align children with end of the container.

9.30.4 Description

Base class for Box containers. It arranges children controls vertically or horizontally, and rearranges them automatically when their minimum size changes.

9.30.5 Member Function Description

- void **add_spacer (***bool* begin **)**

Add a control to the box as a spacer.

If *begin* is true the spacer control will be inserted in front of other children.

- *int* **get_alignment ( )** const

Return the alignment of children in the container.

- void **set_alignment (***int* alignment **)**

Set the alignment of children in the container(Must be one of ALIGN_BEGIN, ALIGN_CENTER or ALIGN_END).

9.31 BoxShape

Inherits: *Shape* < *Resource* < *Reference* < *Object*

Category: Core

9.31.1 Brief Description

Box shape resource.

9.31.2 Member Functions

void	*set_extents* (*Vector3* extents)
Vector3	*get_extents* ( ) const

9.31.3 Description

Box shape resource, which can be set into a *PhysicsBody* or area.

9.31.4 Member Function Description

- void **set_extents** (*Vector3* extents)

Set the half extents for the shape.

- *Vector3* **get_extents** ( ) const

Return the half extents of the shape.

9.32 Button

Inherits: *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *OptionButton*, *ColorPickerButton*, *CheckButton*, *MenuButton*, *ToolButton*, *CheckBox*

Category: Core

9.32.1 Brief Description

Standard themed Button.

9.32.2 Member Functions

void	*set_text* (*String* text)
String	*get_text* ( ) const
void	*set_button_icon* (*Texture* texture)
Texture	*get_button_icon* ( ) const
void	*set_flat* (*bool* enabled)
void	*set_clip_text* (*bool* enabled)
bool	*get_clip_text* ( ) const
void	*set_text_align* (*int* align)
int	*get_text_align* ( ) const
bool	*is_flat* ( ) const

9.32.3 Numeric Constants

- **ALIGN_LEFT = 0** — Align the text to the left.
- **ALIGN_CENTER = 1** — Center the text.
- **ALIGN_RIGHT = 2** — Align the text to the right.

9.32.4 Description

Button is the standard themed button. It can contain text and an icon, and will display them according to the current *Theme*.

9.32.5 Member Function Description

- void **set_text** (*String* text)

Set the button text, which will be displayed inside the button area.

- *String* **get_text** ( ) const

Return the button text.

- void **set_button_icon** (*Texture* texture)

Set the icon that will be displayed next to the text inside the button area.

- *Texture* **get_button_icon** ( ) const

Return the button icon.

- void **set_flat** (*bool* enabled)

Set the *flat* property of a Button. Flat buttons don't display decoration unless hovered or pressed.

- void **set_clip_text** (*bool* enabled)

Set the *clip_text* property of a Button. When this property is enabled, text that is too large to fit the button is clipped, when disabled (default) the Button will always be wide enough to hold the text.

- *bool* **get_clip_text** ( ) const

Return the state of the *clip_text* property (see *set_clip_text*)

- void **set_text_align** (*int* align)

Set the text alignment policy, using one of the ALIGN_* constants.

- *int* **get_text_align** ( ) const

Return the text alignment policy.

- *bool* **is_flat** ( ) const

Return the state of the *flat* property (see *set_flat*).

9.33 ButtonArray

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *HButtonArray*, *VButtonArray*

Category: Core

9.33.1 Brief Description

Array of Buttons.

9.33.2 Member Functions

void	*add_button* (*String* text)
void	*add_icon_button* (*Texture* icon, *String* text="")
void	*set_button_text* (*int* button_idx, *String* text)
void	*set_button_icon* (*int* button_idx, *Texture* icon)
String	*get_button_text* (*int* button_idx) const
Texture	*get_button_icon* (*int* button_idx) const
int	*get_button_count* ( ) const
int	*get_selected* ( ) const
int	*get_hovered* ( ) const
void	*set_selected* (*int* button_idx)
void	*erase_button* (*int* button_idx)
void	*clear* ( )

9.33.3 Signals

- **button_selected** (*int* button_idx)

9.33.4 Numeric Constants

- **ALIGN_BEGIN = 0** — Align buttons at the beginning.
- **ALIGN_CENTER = 1** — Align buttons in the middle.
- **ALIGN_END = 2** — Align buttons at the end.
- **ALIGN_FILL = 3** — Spread the buttons, but keep them small.
- **ALIGN_EXPAND_FILL = 4** — Spread the buttons, but expand them.

9.33.5 Description

Array of Buttons. A ButtonArray is useful to have an array of buttons laid out vertically or horizontally. Only one button can be selected, and is referenced by its index in the array (first button is 0, second button is 1, etc.).

This is useful *e.g.* for joypad-friendly interfaces and option menus.

9.33.6 Member Function Description

- void **add_button** (*String* text)

Append a new button to the array, with the specified text.

- void **add_icon_button** (*Texture* icon, *String* text="")

Append a new button to the array, with the specified icon and text.

- void **set_button_text** (*int* button_idx, *String* text)

Define the text of the specified button.

- void **set_button_icon** (*int* button_idx, *Texture* icon)

Set the icon of the specified button.

- *String* **get_button_text** (*int* button_idx) const

Return the text of the specified button.

- *Texture* **get_button_icon** (*int* button_idx) const

Return the icon of the specified button.

- *int* **get_button_count** ( ) const

Return the amount of buttons in the array.

- *int* **get_selected** ( ) const

Return the index of the currently selected button in the array.

- *int* **get_hovered** ( ) const

Return the index of the currently hovered button in the array.

- void **set_selected** (*int* button_idx)

Select a button in the array based on the given index.

- void **erase_button** (*int* button_idx)

Remove the specified button in the array.

- void **clear** ( )

Remove all buttons from the array.

9.34 ButtonGroup

Inherits: *BoxContainer* < *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.34.1 Brief Description

Group of Buttons.

9.34.2 Member Functions

BaseButton	*get_pressed_button* ( ) const
int	*get_pressed_button_index* ( ) const
BaseButton	*get_focused_button* ( ) const
Array	*get_button_list* ( ) const
void	*set_pressed_button* (*BaseButton* button)

9.34.3 Description

Group of *Button*. All direct and indirect children buttons become radios. Only one allows being pressed.

9.34.4 Member Function Description

- *BaseButton* **get_pressed_button** ( ) const

Return the pressed button.

- *int* **get_pressed_button_index** ( ) const

Return the index of the pressed button (by tree order).

- *BaseButton* **get_focused_button** ( ) const

Return the focused button.

- *Array* **get_button_list** ( ) const

Return the list of all the buttons in the group.

- void **set_pressed_button** (*BaseButton* button)

Set the button to be pressed.

9.35 Camera

Inherits: *Spatial* < *Node* < *Object*

Inherited By: *InterpolatedCamera*

Category: Core

9.35.1 Brief Description

Camera node, displays from a point of view.

9.35.2 Member Functions

Vector3	*project_ray_normal* (*Vector2* screen_point) const
Vector3	*project_local_ray_normal* (*Vector2* screen_point) const
Vector3	*project_ray_origin* (*Vector2* screen_point) const
Vector2	*unproject_position* (*Vector3* world_point) const
bool	*is_position_behind* (*Vector3* world_point) const
Vector3	*project_position* (*Vector2* screen_point) const
void	*set_perspective* (*float* fov, *float* z_near, *float* z_far)
void	*set_orthogonal* (*float* size, *float* z_near, *float* z_far)
void	*make_current* ( )
void	*clear_current* ( )
bool	*is_current* ( ) const
Transform	*get_camera_transform* ( ) const
float	*get_fov* ( ) const
float	*get_size* ( ) const
float	*get_zfar* ( ) const
float	*get_znear* ( ) const
int	*get_projection* ( ) const
void	*set_visible_layers* (*int* mask)
int	*get_visible_layers* ( ) const
void	*set_environment* (*Environment* env)
Environment	*get_environment* ( ) const
void	*set_keep_aspect_mode* (*int* mode)
int	*get_keep_aspect_mode* ( ) const

9.35.3 Numeric Constants

- **PROJECTION_PERSPECTIVE = 0** — Perspective Projection (object's size on the screen becomes smaller when far away).
- **PROJECTION_ORTHOGONAL = 1** — Orthogonal Projection (objects remain the same size on the screen no matter how far away they are).
- **KEEP_WIDTH = 0**
- **KEEP_HEIGHT = 1**

9.35.4 Description

Camera is a special node that displays what is visible from its current location. Cameras register themselves in the nearest *Viewport* node (when ascending the tree). Only one camera can be active per viewport. If no viewport is available ascending the tree, the Camera will register in the global viewport. In other words, a Camera just provides *3D* display capabilities to a *Viewport*, and, without one, a scene registered in that *Viewport* (or higher viewports) can't be displayed.

9.35.5 Member Function Description

- *Vector3* **project_ray_normal** (*Vector2* screen_point) const

Return a normal vector in worldspace, that is the result of projecting a point on the *Viewport* rectangle by the camera projection. This is useful for casting rays in the form of (origin,normal) for object intersection or picking.

- *Vector3* **project_local_ray_normal** (*Vector2* screen_point) const
- *Vector3* **project_ray_origin** (*Vector2* screen_point) const

Return a 3D position in worldspace, that is the result of projecting a point on the *Viewport* rectangle by the camera projection. This is useful for casting rays in the form of (origin,normal) for object intersection or picking.

- *Vector2* **unproject_position** (*Vector3* world_point) const

Return how a 3D point in worldspace maps to a 2D coordinate in the *Viewport* rectangle.

- *bool* **is_position_behind** (*Vector3* world_point) const
- *Vector3* **project_position** (*Vector2* screen_point) const
- void **set_perspective** (*float* fov, *float* z_near, *float* z_far)

Set the camera projection to perspective mode, by specifying a *FOV* Y angle in degrees (FOV means Field of View), and the *near* and *far* clip planes in worldspace units.

- void **set_orthogonal** (*float* size, *float* z_near, *float* z_far)

Set the camera projection to orthogonal mode, by specifying a width and the *near* and *far* clip planes in worldspace units. (As a hint, 2D games often use this projection, with values specified in pixels)

- void **make_current** ( )

Make this camera the current Camera for the *Viewport* (see class description). If the Camera Node is outside the scene tree, it will attempt to become current once it's added.

- void **clear_current** ( )
- *bool* **is_current** ( ) const

Return whether the Camera is the current one in the *Viewport*, or plans to become current (if outside the scene tree).

- *Transform* **get_camera_transform** ( ) const

Get the camera transform. Subclassed cameras (such as CharacterCamera) may provide different transforms than the *Node* transform.

- *float* **get_fov** ( ) const
- *float* **get_size** ( ) const
- *float* **get_zfar** ( ) const
- *float* **get_znear** ( ) const
- *int* **get_projection** ( ) const
- void **set_visible_layers** (*int* mask)
- *int* **get_visible_layers** ( ) const
- void **set_environment** (*Environment* env)
- *Environment* **get_environment** ( ) const
- void **set_keep_aspect_mode** (*int* mode)
- *int* **get_keep_aspect_mode** ( ) const

9.36 Camera2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.36.1 Brief Description

Camera node for 2D scenes.

9.36.2 Member Functions

void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const
void	*set_anchor_mode* (*int* anchor_mode)
int	*get_anchor_mode* ( ) const
void	*set_rotating* (*bool* rotating)
bool	*is_rotating* ( ) const
void	*make_current* ( )
void	*clear_current* ( )
bool	*is_current* ( ) const
void	*set_limit* (*int* margin, *int* limit)
int	*get_limit* (*int* margin) const
void	*set_v_drag_enabled* (*bool* enabled)
bool	*is_v_drag_enabled* ( ) const
void	*set_h_drag_enabled* (*bool* enabled)
bool	*is_h_drag_enabled* ( ) const
void	*set_v_offset* (*float* ofs)
float	*get_v_offset* ( ) const
void	*set_h_offset* (*float* ofs)
float	*get_h_offset* ( ) const
void	*set_drag_margin* (*int* margin, *float* drag_margin)
float	*get_drag_margin* (*int* margin) const
Vector2	*get_camera_pos* ( ) const
Vector2	*get_camera_screen_center* ( ) const
void	*set_zoom* (*Vector2* zoom)
Vector2	*get_zoom* ( ) const
void	*set_follow_smoothing* (*float* follow_smoothing)
float	*get_follow_smoothing* ( ) const
void	*set_enable_follow_smoothing* (*bool* follow_smoothing)
bool	*is_follow_smoothing_enabled* ( ) const
void	*force_update_scroll* ( )

9.36.3 Numeric Constants

- **ANCHOR_MODE_DRAG_CENTER = 1**
- **ANCHOR_MODE_FIXED_TOP_LEFT = 0**

9.36.4 Description

Camera node for 2D scenes. It forces the screen (current layer) to scroll following this node. This makes it easier (and faster) to program scrollable scenes than manually changing the position of *CanvasItem* based nodes.

This node is intended to be a simple helper get get things going quickly and it may happen often that more functionality is desired to change how the camera works. To make your own custom camera node, simply inherit from *Node2D* and change the transform of the canvas by calling get_viewport().set_canvas_transform(m) in *Viewport*.

9.36.5 Member Function Description

- void **set_offset** (*Vector2* offset)

Set the scroll offset. Useful for looking around or camera shake animations.

- *Vector2* **get_offset** ( ) const

Return the scroll offset.

- void **set_anchor_mode** (*int* anchor_mode)
- *int* **get_anchor_mode** ( ) const
- void **set_rotating** (*bool* rotating)
- *bool* **is_rotating** ( ) const
- void **make_current** ( )

Make this the current 2D camera for the scene (viewport and layer), in case there's many cameras in the scene.

- void **clear_current** ( )
- *bool* **is_current** ( ) const

Return true of this is the current camera (see *make_current*).

- void **set_limit** (*int* margin, *int* limit)

Set the scrolling limit in pixels.

- *int* **get_limit** (*int* margin) const

Return the scrolling limit in pixels.

- void **set_v_drag_enabled** (*bool* enabled)
- *bool* **is_v_drag_enabled** ( ) const
- void **set_h_drag_enabled** (*bool* enabled)
- *bool* **is_h_drag_enabled** ( ) const
- void **set_v_offset** (*float* ofs)
- *float* **get_v_offset** ( ) const
- void **set_h_offset** (*float* ofs)
- *float* **get_h_offset** ( ) const
- void **set_drag_margin** (*int* margin, *float* drag_margin)

Set the margins needed to drag the camera (relative to the screen size). Margin uses the MARGIN_* enum. Drag margins of 0,0,0,0 will keep the camera at the center of the screen, while drag margins of 1,1,1,1 will only move when the camera is at the edges.

- *float* **get_drag_margin** (*int* margin) const

Return the margins needed to drag the camera (see *set_drag_margin*).

- *Vector2* **get_camera_pos** ( ) const

Return the camera position.

- *Vector2* **get_camera_screen_center** ( ) const
- void **set_zoom** (*Vector2* zoom)
- *Vector2* **get_zoom** ( ) const
- void **set_follow_smoothing** (*float* follow_smoothing)
- *float* **get_follow_smoothing** ( ) const
- void **set_enable_follow_smoothing** (*bool* follow_smoothing)
- *bool* **is_follow_smoothing_enabled** ( ) const
- void **force_update_scroll** ( )

Force the camera to update scroll immediately.

9.37 CanvasItem

Inherits: *Node* < *Object*

Inherited By: *Node2D*, *Control*

Category: Core

9.37.1 Brief Description

Base class of anything 2D.

9.37.2 Member Functions

void	*_draw* ( ) virtual
void	*edit_set_state* (var state)
void	*edit_get* ( ) const
void	*edit_set_rect* (*Rect2* rect)
void	*edit_rotate* (*float* degrees)
Rect2	*get_item_rect* ( ) const
RID	*get_canvas_item* ( ) const
bool	*is_visible* ( ) const
bool	*is_hidden* ( ) const
void	*show* ( )
void	*hide* ( )
void	*set_hidden* (*bool* hidden)
void	*update* ( )
void	*set_as_toplevel* (*bool* enable)
bool	*is_set_as_toplevel* ( ) const

Continúa en la pá

Tabla 9.8 – proviene de la página anterior

void	*set_blend_mode* (*int* blend_mode)
int	*get_blend_mode* ( ) const
void	*set_light_mask* (*int* light_mask)
int	*get_light_mask* ( ) const
void	*set_opacity* (*float* opacity)
float	*get_opacity* ( ) const
void	*set_self_opacity* (*float* self_opacity)
float	*get_self_opacity* ( ) const
void	*set_draw_behind_parent* (*bool* enable)
bool	*is_draw_behind_parent_enabled* ( ) const
void	*draw_line* (*Vector2* from, *Vector2* to, *Color* color, *float* width=1)
void	*draw_rect* (*Rect2* rect, *Color* color)
void	*draw_circle* (*Vector2* pos, *float* radius, *Color* color)
void	*draw_texture* (*Texture* texture, *Vector2* pos, *Color* modulate=Color(1,1,1,1))
void	*draw_texture_rect* (*Texture* texture, *Rect2* rect, *bool* tile, *Color* modulate=Color(1,1,1,1), *bool* transpose=false
void	*draw_texture_rect_region* (*Texture* texture, *Rect2* rect, *Rect2* src_rect, *Color* modulate=Color(1,1,1,1), *bool* tr
void	*draw_style_box* (*StyleBox* style_box, *Rect2* rect)
void	*draw_primitive* (*Vector2Array* points, *ColorArray* colors, *Vector2Array* uvs, *Texture* texture=NULL, *float* wid
void	*draw_polygon* (*Vector2Array* points, *ColorArray* colors, *Vector2Array* uvs=Vector2Array(), *Texture* texture=N
void	*draw_colored_polygon* (*Vector2Array* points, *Color* color, *Vector2Array* uvs=Vector2Array(), *Texture* texture=
void	*draw_string* (*Font* font, *Vector2* pos, *String* text, *Color* modulate=Color(1,1,1,1), *int* clip_w=-1)
float	*draw_char* (*Font* font, *Vector2* pos, *String* char, *String* next, *Color* modulate=Color(1,1,1,1))
void	*draw_set_transform* (*Vector2* pos, *float* rot, *Vector2* scale)
Matrix32	*get_transform* ( ) const
Matrix32	*get_global_transform* ( ) const
Matrix32	*get_global_transform_with_canvas* ( ) const
Matrix32	*get_viewport_transform* ( ) const
Rect2	*get_viewport_rect* ( ) const
Matrix32	*get_canvas_transform* ( ) const
Vector2	*get_local_mouse_pos* ( ) const
Vector2	*get_global_mouse_pos* ( ) const
RID	*get_canvas* ( ) const
Object	*get_world_2d* ( ) const
void	*set_material* (*CanvasItemMaterial* material)
CanvasItemMaterial	*get_material* ( ) const
void	*set_use_parent_material* (*bool* enable)
bool	*get_use_parent_material* ( ) const
InputEvent	*make_input_local* (*InputEvent* event) const

9.37.3 Signals

- **item_rect_changed** ( )
- **draw** ( )
- **visibility_changed** ( )
- **hide** ( )

9.37.4 Numeric Constants

- **BLEND_MODE_MIX = 0** — Mix blending mode. Colors are assumed to be independent of the alpha (opacity) value.
- **BLEND_MODE_ADD = 1** — Additive blending mode.
- **BLEND_MODE_SUB = 2** — Subtractive blending mode.
- **BLEND_MODE_MUL = 3** — Multiplicative blending mode.
- **BLEND_MODE_PREMULT_ALPHA = 4** — Mix blending mode. Colors are assumed to be premultiplied by the alpha (opacity) value.
- **NOTIFICATION_DRAW = 30** — CanvasItem is requested to draw.
- **NOTIFICATION_VISIBILITY_CHANGED = 31** — Canvas item visibility has changed.
- **NOTIFICATION_ENTER_CANVAS = 32** — Canvas item has entered the canvas.
- **NOTIFICATION_EXIT_CANVAS = 33** — Canvas item has exited the canvas.
- **NOTIFICATION_TRANSFORM_CHANGED = 29** — Canvas item transform has changed. Only received if requested.

9.37.5 Description

Base class of anything 2D. Canvas items are laid out in a tree and children inherit and extend the transform of their parent. CanvasItem is extended by *Control*, for anything GUI related, and by *Node2D* for anything 2D engine related.

Any CanvasItem can draw. For this, the "update" function must be called, then NOTIFICATION_DRAW will be received on idle time to request redraw. Because of this, canvas items don't need to be redraw on every frame, improving the performance significantly. Several functions for drawing on the CanvasItem are provided (see draw_* functions). They can only be used inside the notification, signal or _draw() overrides function, though.

Canvas items are draw in tree order. By default, children are on top of their parents so a root CanvasItem will be drawn behind everything (this can be changed per item though).

Canvas items can also be hidden (hiding also their subtree). They provide many means for changing standard parameters such as opacity (for it and the subtree) and self opacity, blend mode.

Ultimately, a transform notification can be requested, which will notify the node that its global position changed in case the parent tree changed.

9.37.6 Member Function Description

- void **_draw** **(** **)** virtual

Called (if exists) to draw the canvas item.

- void **edit_set_state** **(** var state **)**

Used for editing, returns an opaque value representing the transform state.

- void **edit_get** **(** **)** const
- void **edit_set_rect** **(** *Rect2* rect **)**
- void **edit_rotate** **(** *float* degrees **)**

Used for editing, handle rotation.

- *Rect2* **get_item_rect** **(** **)** const

Return a rect containing the editable contents of the item.

- *RID* **get_canvas_item** ( ) const

Return the canvas item RID used by *VisualServer* for this item.

- *bool* **is_visible** ( ) const

Return true if this CanvasItem is visible. It may be invisible because itself or a parent canvas item is hidden.

- *bool* **is_hidden** ( ) const

Return true if this CanvasItem is hidden. Note that the CanvasItem may not be visible, but as long as it's not hidden (*hide* called) the function will return false.

- void **show** ( )

Show the CanvasItem currently hidden.

- void **hide** ( )

Hide the CanvasItem currently visible.

- void **set_hidden** (*bool* hidden)
- void **update** ( )

Queue the CanvasItem for update. NOTIFICATION_DRAW will be called on idle time to request redraw.

- void **set_as_toplevel** (*bool* enable)

Set as toplevel. This means that it will not inherit transform from parent canvas items.

- *bool* **is_set_as_toplevel** ( ) const

Return if set as toplevel. See *set_as_toplevel*.

- void **set_blend_mode** (*int* blend_mode)

Set the blending mode from enum BLEND_MODE_*.

- *int* **get_blend_mode** ( ) const

Return the current blending mode from enum BLEND_MODE_*.

- void **set_light_mask** (*int* light_mask)
- *int* **get_light_mask** ( ) const
- void **set_opacity** (*float* opacity)

Set canvas item opacity. This will affect the canvas item and all the children.

- *float* **get_opacity** ( ) const

Return the canvas item opacity. This affects the canvas item and all the children.

- void **set_self_opacity** (*float* self_opacity)

Set canvas item self-opacity. This does not affect the opacity of children items.

- *float* **get_self_opacity** ( ) const

Return the canvas item self-opacity.

- void **set_draw_behind_parent** (*bool* enable)

Sets whether the canvas item is drawn behind its parent.

- *bool* **is_draw_behind_parent_enabled** ( ) const

Return whether the item is drawn behind its parent.

- void **draw_line** (*Vector2* from, *Vector2* to, *Color* color, *float* width=1)

Draw a line from a 2D point to another, with a given color and width.

- void **draw_rect** (*Rect2* rect, *Color* color)

Draw a colored rectangle.

- void **draw_circle** (*Vector2* pos, *float* radius, *Color* color)

Draw a colored circle.

- void **draw_texture** (*Texture* texture, *Vector2* pos, *Color* modulate=Color(1,1,1,1))

Draw a texture at a given position.

- void **draw_texture_rect** (*Texture* texture, *Rect2* rect, *bool* tile, *Color* modulate=Color(1,1,1,1), *bool* transpose=false)

Draw a textured rectangle at a given position, optionally modulated by a color. Transpose swaps the x and y coordinates when reading the texture.

- void **draw_texture_rect_region** (*Texture* texture, *Rect2* rect, *Rect2* src_rect, *Color* modulate=Color(1,1,1,1), *bool* transpose=false)

Draw a textured rectangle region at a given position, optionally modulated by a color. Transpose swaps the x and y coordinates when reading the texture.

- void **draw_style_box** (*StyleBox* style_box, *Rect2* rect)

Draw a styled rectangle.

- void **draw_primitive** (*Vector2Array* points, *ColorArray* colors, *Vector2Array* uvs, *Texture* texture=NULL, *float* width=1)

Draw a custom primitive, 1 point for a point, 2 points for a line, 3 points for a triangle and 4 points for a quad.

- void **draw_polygon** (*Vector2Array* points, *ColorArray* colors, *Vector2Array* uvs=Vector2Array(), *Texture* texture=NULL)

Draw a polygon of any amount of points, convex or concave.

- void **draw_colored_polygon** (*Vector2Array* points, *Color* color, *Vector2Array* uvs=Vector2Array(), *Texture* texture=NULL)

Draw a colored polygon of any amount of points, convex or concave.

- void **draw_string** (*Font* font, *Vector2* pos, *String* text, *Color* modulate=Color(1,1,1,1), *int* clip_w=-1)

Draw a string using a custom font.

- *float* **draw_char** (*Font* font, *Vector2* pos, *String* char, *String* next, *Color* modulate=Color(1,1,1,1))

Draw a string character using a custom font. Returns the advance, depending on the char width and kerning with an optional next char.

- void **draw_set_transform** (*Vector2* pos, *float* rot, *Vector2* scale)

Set a custom transform for drawing. Anything drawn afterwards will be transformed by this.

- *Matrix32* **get_transform** ( ) const
- *Matrix32* **get_global_transform** ( ) const
- *Matrix32* **get_global_transform_with_canvas** ( ) const
- *Matrix32* **get_viewport_transform** ( ) const

- *Rect2* **get_viewport_rect** ( ) const
- *Matrix32* **get_canvas_transform** ( ) const
- *Vector2* **get_local_mouse_pos** ( ) const
- *Vector2* **get_global_mouse_pos** ( ) const
- *RID* **get_canvas** ( ) const
- *Object* **get_world_2d** ( ) const
- void **set_material** (*CanvasItemMaterial* material)
- *CanvasItemMaterial* **get_material** ( ) const
- void **set_use_parent_material** (*bool* enable)
- *bool* **get_use_parent_material** ( ) const
- *InputEvent* **make_input_local** (*InputEvent* event) const

9.38 CanvasItemMaterial

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.38.1 Brief Description

9.38.2 Member Functions

void	*set_shader* (*Shader* shader)
Shader	*get_shader* ( ) const
void	*set_shader_param* (*String* param, var value)
void	*get_shader_param* (*String* param) const
void	*set_shading_mode* (*int* mode)
int	*get_shading_mode* ( ) const

9.38.3 Numeric Constants

- **SHADING_NORMAL = 0**
- **SHADING_UNSHADED = 1**
- **SHADING_ONLY_LIGHT = 2**

9.38.4 Member Function Description

- void **set_shader** (*Shader* shader)
- *Shader* **get_shader** ( ) const
- void **set_shader_param** (*String* param, var value)
- void **get_shader_param** (*String* param) const
- void **set_shading_mode** (*int* mode)

- *int* **get_shading_mode** ( ) const

9.39 CanvasItemShader

Inherits: *Shader* < *Resource* < *Reference* < *Object*

Category: Core

9.39.1 Brief Description

9.40 CanvasItemShaderGraph

Inherits: *ShaderGraph* < *Shader* < *Resource* < *Reference* < *Object*

Category: Core

9.40.1 Brief Description

9.41 CanvasLayer

Inherits: *Node* < *Object*

Inherited By: *ParallaxBackground*

Category: Core

9.41.1 Brief Description

Canvas Item layer.

9.41.2 Member Functions

void	*set_layer* (*int* layer)
int	*get_layer* ( ) const
void	*set_transform* (*Matrix32* transform)
Matrix32	*get_transform* ( ) const
void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const
void	*set_rotation* (*float* radians)
float	*get_rotation* ( ) const
void	*set_rotationd* (*float* degrees)
float	*get_rotationd* ( ) const
void	*set_scale* (*Vector2* scale)
Vector2	*get_scale* ( ) const
World2D	*get_world_2d* ( ) const
RID	*get_viewport* ( ) const

9.41.3 Description

Canvas Item layer. *CanvasItem* nodes that are direct or indirect children of a *CanvasLayer* will be drawn in that layer. The layer is a numeric index that defines the draw order. The default 2D scene renders with index 0, so a *CanvasLayer* with index -1 will be drawn below, and one with index 1 will be drawn above. This is very useful for HUDs (in layer 1+ or above), or backgrounds (in layer -1 or below).

9.41.4 Member Function Description

- void **set_layer** (*int* layer)

Set the layer index, determines the draw order, a lower value will be below a higher one.

- *int* **get_layer** ( ) const

Return the layer index, determines the draw order, a lower value will be below a higher one.

- void **set_transform** (*Matrix32* transform)

Set the base transform for this layer.

- *Matrix32* **get_transform** ( ) const

Return the base transform for this layer.

- void **set_offset** (*Vector2* offset)

Set the base offset for this layer (helper).

- *Vector2* **get_offset** ( ) const

Return the base offset for this layer (helper).

- void **set_rotation** (*float* radians)

Set the base rotation for this layer (helper).

- *float* **get_rotation** ( ) const

Return the base rotation for this layer (helper).

- void **set_rotationd** (*float* degrees)

Set rotation of the layer in degree.

- *float* **get_rotationd** ( ) const

Get rotation of the layer in degree.

- void **set_scale** (*Vector2* scale)

Set the base scale for this layer (helper).

- *Vector2* **get_scale** ( ) const

Return the base scale for this layer (helper).

- *World2D* **get_world_2d** ( ) const

Return the *World2D* used by this layer.

- *RID* **get_viewport** ( ) const

Return the viewport RID for this layer.

9.42 CanvasModulate

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.42.1 Brief Description

Tint the entire canvas

9.42.2 Member Functions

void	*set_color* (*Color* color)
Color	*get_color* ( ) const

9.42.3 Description

CanvasModulate tints the canvas elements using its asigned color

9.42.4 Member Function Description

- void **set_color** (*Color* color)

Sets the canvas tint color

- *Color* **get_color** ( ) const

Gets the canvas tint color

9.43 CapsuleShape

Inherits: *Shape* < *Resource* < *Reference* < *Object*

Category: Core

9.43.1 Brief Description

Capsule shape resource.

9.43.2 Member Functions

void	*set_radius* (*float* radius)
float	*get_radius* ( ) const
void	*set_height* (*float* height)
float	*get_height* ( ) const

9.43.3 Description

Capsule shape resource, which can be set into a *PhysicsBody* or area.

9.43.4 Member Function Description

- void **set_radius** (*float* radius)

Set the capsule radius.

- *float* **get_radius** ( ) const

Return the capsule radius.

- void **set_height** (*float* height)

Set the capsule height.

- *float* **get_height** ( ) const

Return the capsule height.

9.44 CapsuleShape2D

Inherits: *Shape2D* < *Resource* < *Reference* < *Object*

Category: Core

9.44.1 Brief Description

Capsule 2D shape resource for physics.

9.44.2 Member Functions

void	*set_radius* (*float* radius)
float	*get_radius* ( ) const
void	*set_height* (*float* height)
float	*get_height* ( ) const

9.44.3 Description

Capsule 2D shape resource for physics. A capsule (or sometimes called "pill") is like a line grown in all directions. It has a radius and a height, and is often useful for modeling biped characters.

9.44.4 Member Function Description

- void **set_radius** (*float* radius)

Set the radius of the *CapsuleShape2D*.

- *float* **get_radius** ( ) const

Return the radius of the *CapsuleShape2D*.

- void **set_height** (*float* height)

Set the height of the *CapsuleShape2D*.

- *float* **get_height** ( ) const

Return the height of the *CapsuleShape2D*.

9.45 CenterContainer

Inherits: *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.45.1 Brief Description

Keeps children controls centered.

9.45.2 Member Functions

void	*set_use_top_left* (*bool* enable)
bool	*is_using_top_left* ( ) const

9.45.3 Description

CenterContainer Keeps children controls centered. This container keeps all children to their minimum size, in the center.

9.45.4 Member Function Description

- void **set_use_top_left** (*bool* enable)

This function will anchor the container children to the top left corner of the the container boundaries, moving all its children to that position, (the children new center will be the top left corner of the container).

- *bool* **is_using_top_left** ( ) const

Should put children to the top left corner instead of center of the container.

9.46 CheckBox

Inherits: *Button* < *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.46.1 Brief Description

Binary choice user interface widget

9.46.2 Description

A checkbox allows the user to make a binary choice (choosing only one of two posible options), for example Answer 'yes' or 'no'.

9.47 CheckButton

Inherits: *Button* < *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.47.1 Brief Description

Checkable button.

9.47.2 Description

CheckButton is a toggle button displayed as a check field.

9.48 CircleShape2D

Inherits: *Shape2D* < *Resource* < *Reference* < *Object*

Category: Core

9.48.1 Brief Description

Circular Shape for 2D Physics.

9.48.2 Member Functions

void	*set_radius* (*float* radius)
float	*get_radius* ( ) const

9.48.3 Description

Circular Shape for 2D Physics. This shape is useful for modeling balls or small characters and it's collision detection with everything else is very fast.

9.48.4 Member Function Description

- void **set_radius** (*float* radius)

Set the radius of the circle shape.

- *float* **get_radius** ( ) const

Return the radius of the circle shape.

9.49 CollisionObject

Inherits: *Spatial* < *Node* < *Object*

Inherited By: *PhysicsBody*, *Area*

Category: Core

9.49.1 Brief Description

9.49.2 Member Functions

void	*_input_event* (*Object* camera, *InputEvent* event, *Vector3* click_pos, *Vector3* click_normal, *int* shape_idx) virtual
void	*add_shape* (*Shape* shape, *Transform* transform=Transform())
int	*get_shape_count* ( ) const
void	*set_shape* (*int* shape_idx, *Shape* shape)
void	*set_shape_transform* (*int* shape_idx, *Transform* transform)
void	*set_shape_as_trigger* (*int* shape_idx, *bool* enable)
bool	*is_shape_set_as_trigger* (*int* shape_idx) const
Shape	*get_shape* (*int* shape_idx) const
Transform	*get_shape_transform* (*int* shape_idx) const
void	*remove_shape* (*int* shape_idx)
void	*clear_shapes* ( )
void	*set_ray_pickable* (*bool* ray_pickable)
bool	*is_ray_pickable* ( ) const
void	*set_capture_input_on_drag* (*bool* enable)
bool	*get_capture_input_on_drag* ( ) const
RID	*get_rid* ( ) const

9.49.3 Signals

- **mouse_enter** ( )
- **input_event** (*Object* camera, *InputEvent* event, *Vector3* click_pos, *Vector3* click_normal, *int* shape_idx)
- **mouse_exit** ( )

9.49.4 Member Function Description

- void **_input_event** (*Object* camera, *InputEvent* event, *Vector3* click_pos, *Vector3* click_normal, *int* shape_idx) virtual
- void **add_shape** (*Shape* shape, *Transform* transform=Transform())
- *int* **get_shape_count** ( ) const
- void **set_shape** (*int* shape_idx, *Shape* shape)
- void **set_shape_transform** (*int* shape_idx, *Transform* transform)
- void **set_shape_as_trigger** (*int* shape_idx, *bool* enable)
- *bool* **is_shape_set_as_trigger** (*int* shape_idx) const

- *Shape* **get_shape** (*int* shape_idx) const
- *Transform* **get_shape_transform** (*int* shape_idx) const
- void **remove_shape** (*int* shape_idx)
- void **clear_shapes** ( )
- void **set_ray_pickable** (*bool* ray_pickable)
- *bool* **is_ray_pickable** ( ) const
- void **set_capture_input_on_drag** (*bool* enable)
- *bool* **get_capture_input_on_drag** ( ) const
- *RID* **get_rid** ( ) const

9.50 CollisionObject2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Inherited By: *Area2D*, *PhysicsBody2D*

Category: Core

9.50.1 Brief Description

Base node for 2D collisionables.

9.50.2 Member Functions

void	*_input_event* (*Object* viewport, *InputEvent* event, *int* shape_idx) virtual
void	*add_shape* (*Shape2D* shape, *Matrix32* transform=1,0, 0,1, 0,0)
int	*get_shape_count* ( ) const
void	*set_shape* (*int* shape_idx, *Shape* shape)
void	*set_shape_transform* (*int* shape_idx, *Matrix32* transform)
void	*set_shape_as_trigger* (*int* shape_idx, *bool* enable)
Shape2D	*get_shape* (*int* shape_idx) const
Matrix32	*get_shape_transform* (*int* shape_idx) const
bool	*is_shape_set_as_trigger* (*int* shape_idx) const
void	*remove_shape* (*int* shape_idx)
void	*clear_shapes* ( )
RID	*get_rid* ( ) const
void	*set_pickable* (*bool* enabled)
bool	*is_pickable* ( ) const

9.50.3 Signals

- **mouse_enter** ( )
- **input_event** (*Object* viewport, *InputEvent* event, *int* shape_idx)
- **mouse_exit** ( )

9.50.4 Description

CollisionObject2D is the base class for 2D physics collisionables. They can hold any number of 2D collision shapes. Usually, they are edited by placing *CollisionShape2D* and/or *CollisionPolygon2D* nodes as children. Such nodes are for reference and not present outside the editor, so code should use the regular shape API.

9.50.5 Member Function Description

- void **_input_event** (*Object* viewport, *InputEvent* event, *int* shape_idx) virtual

This method can be used to override normal input processing. The first parameter is the viewport where the event took place. The second holds the input event received, and the third the shape of this object where it happened.

- void **add_shape** (*Shape2D* shape, *Matrix32* transform=1,0, 0,1, 0,0)

Add a *Shape2D* to the collision body, with a given custom transform.

- *int* **get_shape_count** ( ) const

Return the amount of shapes in the collision body. Because a *CollisionPolygon2D* can generate more than one *Shape2D*, the amount returned does not have to match the sum of *CollisionShape2D* and *CollisionPolygon2D*.

- void **set_shape** (*int* shape_idx, *Shape* shape)

Change a shape in the collision body.

- void **set_shape_transform** (*int* shape_idx, *Matrix32* transform)

Change the shape transform in the collision body.

- void **set_shape_as_trigger** (*int* shape_idx, *bool* enable)

Set whether a shape is a trigger. A trigger shape detects collisions, but is otherwise unaffected by physics (i.e. colliding objects will not get blocked).

- *Shape2D* **get_shape** (*int* shape_idx) const

Return the shape in the given index.

- *Matrix32* **get_shape_transform** (*int* shape_idx) const

Return the shape transform in the given index.

- *bool* **is_shape_set_as_trigger** (*int* shape_idx) const

Return whether a shape is a trigger. A trigger shape detects collisions, but is otherwise unaffected by physics (i.e. colliding objects will not get blocked).

- void **remove_shape** (*int* shape_idx)

Remove the shape in the given index.

- void **clear_shapes** ( )

Remove all shapes.

- *RID* **get_rid** ( ) const

Return the RID of this object.

- void **set_pickable** (*bool* enabled)

Set whether this object is pickable. A pickable object can detect the mouse pointer enter/leave it and, if the mouse is inside it, report input events.

- *bool* **is_pickable** ( ) const

Return whether this object is pickable.

9.51 CollisionPolygon

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.51.1 Brief Description

9.51.2 Member Functions

void	*set_build_mode* (*int* build_mode)
int	*get_build_mode* ( ) const
void	*set_depth* (*float* depth)
float	*get_depth* ( ) const
void	*set_polygon* (*Vector2Array* polygon)
Vector2Array	*get_polygon* ( ) const
int	*get_collision_object_first_shape* ( ) const
int	*get_collision_object_last_shape* ( ) const

9.51.3 Member Function Description

- void **set_build_mode** (*int* build_mode)
- *int* **get_build_mode** ( ) const
- void **set_depth** (*float* depth)
- *float* **get_depth** ( ) const
- void **set_polygon** (*Vector2Array* polygon)
- *Vector2Array* **get_polygon** ( ) const
- *int* **get_collision_object_first_shape** ( ) const
- *int* **get_collision_object_last_shape** ( ) const

9.52 CollisionPolygon2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.52.1 Brief Description

Editor-only class for easy editing of collision polygons.

9.52.2 Member Functions

void	*set_polygon* (*Vector2Array* polygon)
Vector2Array	*get_polygon* ( ) const
void	*set_build_mode* (*int* build_mode)
int	*get_build_mode* ( ) const
void	*set_trigger* (*bool* trigger)
bool	*is_trigger* ( ) const
int	*get_collision_object_first_shape* ( ) const
int	*get_collision_object_last_shape* ( ) const

9.52.3 Description

Editor-only class. This is not present when running the game. It's used in the editor to properly edit and position collision shapes in *CollisionObject2D*. This is not accessible from regular code. This class is for editing custom shape polygons.

9.52.4 Member Function Description

- void **set_polygon** (*Vector2Array* polygon)

Set the array of points forming the polygon.

When editing the point list via the editor, depending on *get_build_mode*, it has to be a list of points (for `build_mode==0`), or a list of lines (for `build_mode==1`). In the second case, the even elements of the array define the start point of the line, and the odd elements the end point.

- *Vector2Array* **get_polygon** ( ) const

Return the list of points that define the polygon.

- void **set_build_mode** (*int* build_mode)

Set whether the polygon is to be a *ConvexPolygonShape2D* (`build_mode==0`), or a *ConcavePolygonShape2D* (`build_mode==1`).

- *int* **get_build_mode** ( ) const

Return whether the polygon is a *ConvexPolygonShape2D* (`build_mode==0`), or a *ConcavePolygonShape2D* (`build_mode==1`).

- void **set_trigger** (*bool* trigger)

Set whether this polygon is a trigger. A trigger polygon detects collisions, but is otherwise unaffected by physics (i.e. colliding objects will not get blocked).

- *bool* **is_trigger** ( ) const

Return whether this polygon is a trigger.

- *int* **get_collision_object_first_shape** ( ) const

Return the index of the first shape generated by the editor.

When `build_mode` is set to generate convex polygons, the shape shown in the editor may be decomposed into many convex polygons. In that case, a range of indexes is needed to directly access the *Shape2D*.

When `build_mode` is set to generate concave polygons, there is only one *Shape2D* generated, so the start index and the end index are the same.

- *int* **get_collision_object_last_shape** ( ) const

Return the index of the last shape generated by the editor.

9.53 CollisionShape

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.53.1 Brief Description

9.53.2 Member Functions

void	*resource_changed* (*Object* resource)
void	*set_shape* (*Object* shape)
Object	*get_shape* ( ) const
void	*set_trigger* (*bool* enable)
bool	*is_trigger* ( ) const
void	*make_convex_from_brothers* ( )
int	*get_collision_object_shape_index* ( ) const

9.53.3 Member Function Description

- void **resource_changed** (*Object* resource)
- void **set_shape** (*Object* shape)
- *Object* **get_shape** ( ) const
- void **set_trigger** (*bool* enable)
- *bool* **is_trigger** ( ) const
- void **make_convex_from_brothers** ( )
- *int* **get_collision_object_shape_index** ( ) const

9.54 CollisionShape2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.54.1 Brief Description

Editor-only class for easy editing of shapes.

9.54.2 Member Functions

void	*set_shape* (*Object* shape)
Object	*get_shape* ( ) const
void	*set_trigger* (*bool* enable)
bool	*is_trigger* ( ) const
int	*get_collision_object_shape_index* ( ) const

9.54.3 Description

Editor-only class. This is not present when running the game. It's used in the editor to properly edit and position collision shapes in *CollisionObject2D*. This is not accessible from regular code.

9.54.4 Member Function Description

- void **set_shape** (*Object* shape)

Set this shape's *Shape2D*. This will not appear as a node, but can be directly edited as a property.

- *Object* **get_shape** ( ) const

Return this shape's *Shape2D*.

- void **set_trigger** (*bool* enable)

Set whether this shape is a trigger. A trigger shape detects collisions, but is otherwise unaffected by physics (i.e. will not block movement of colliding objects).

- *bool* **is_trigger** ( ) const

Return whether this shape is a trigger.

- *int* **get_collision_object_shape_index** ( ) const

Return the index of this shape inside its container *CollisionObject2D*. This can be used to directly access the underlying *Shape2D*.

9.55 Color

Category: Built-In Types

9.55.1 Brief Description

Color in RGBA format.

9.55.2 Member Functions

Color	*blend* (*Color* over)
Color	*contrasted* ( )
float	*gray* ( )
Color	*inverted* ( )
Color	*linear_interpolate* (*Color* b, *float* t)
int	*to_32* ( )
int	*to_ARGB32* ( )
String	*to_html* (*bool* with_alpha=True)
Color	Color (*float* r, *float* g, *float* b, *float* a)
Color	Color (*float* r, *float* g, *float* b)
Color	Color (*int* from)
Color	Color (*String* from)

9.55.3 Member Variables

- *float* **r** - Red (0 to 1)
- *float* **g** - Green (0 to 1)
- *float* **b** - Blue (0 to 1)
- *float* **a** - Alpha (0 to 1)
- *float* **h** - Hue (0 to 1)
- *float* **s** - Saturation (0 to 1)
- *float* **v** - Value (0 to 1)
- *int* **r8** - Red (0 to 255)
- *int* **g8** - Green (0 to 255)
- *int* **b8** - Blue (0 to 255)
- *int* **a8** - Alpha (0 to 255)

9.55.4 Description

A color is represented as red, green and blue (r,g,b) components. Additionally, "a" represents the alpha component, often used for transparency. Values are in floating point and usually range from 0 to 1. Some methods (such as set_modulate()) may accept values > 1.

9.55.5 Member Function Description

- *Color* **blend** (*Color* over)

Return a new color blended with anothor one.

- *Color* **contrasted** ( )

Return the most contrasting color with this one.

- *float* **gray** ( )

Convert the color to gray.

- *Color* **inverted ( )**

Return the inverted color (1-r, 1-g, 1-b, 1-a).

- *Color* **linear_interpolate (*Color* b, *float* t)**

Return the linear interpolation with another color.

- *int* **to_32 ( )**

Convert the color to a 32 its integer (each byte represents a RGBA).

- *int* **to_ARGB32 ( )**

Convert color to ARGB32, more compatible with DirectX.

- *String* **to_html (*bool* with_alpha=True)**

Return the HTML hexadecimal color string.

- *Color* **Color (*float* r, *float* g, *float* b, *float* a)**

Construct the color from an RGBA profile.

- *Color* **Color (*float* r, *float* g, *float* b)**

Construct the color from an RGBA profile.

- *Color* **Color (*int* from)**

Construct the color from an RGBA profile.

- *Color* **Color (*String* from)**

Construct the color from an RGBA profile.

9.56 ColorArray

Category: Built-In Types

9.56.1 Brief Description

Array of Colors

9.56.2 Member Functions

void	*push_back* (*Color* color)
void	*resize* (*int* idx)
void	*set* (*int* idx, *Color* color)
int	*size* ( )
ColorArray	*ColorArray* (*Array* from)

9.56.3 Description

Array of Color, can only contains colors. Optimized for memory usage, can't fragment the memory.

9.56.4 Member Function Description

- void **push_back** (*Color* color)

Append a value to the array.

- void **resize** (*int* idx)

Resize the array.

- void **set** (*int* idx, *Color* color)

Set an index in the array.

- *int* **size** ( )

Return the array size.

- *ColorArray* **ColorArray** (*Array* from)

Create from a generic array.

9.57 ColorPicker

Inherits: *BoxContainer* < *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.57.1 Brief Description

Color picker control.

9.57.2 Member Functions

void	*set_color* (*Color* color)
Color	*get_color* ( ) const
void	*set_raw_mode* (*bool* mode)
bool	*is_raw_mode* ( ) const
void	*set_edit_alpha* (*bool* show)
bool	*is_editing_alpha* ( ) const
void	*add_preset* (*Color* arg0)

9.57.3 Signals

- **color_changed** (*Color* color)

9.57.4 Description

This is a simple color picker *Control*. It's useful for selecting a color from an RGB/RGBA colorspace.

9.57.5 Member Function Description

- void **set_color** (*Color* color)

Select the current color.

- *Color* **get_color** ( ) const

Return the current (edited) color.

- void **set_raw_mode** (*bool* mode)

When set to true, every color channel will be represented as a value from 0 to 1, insetead of 0, 255.

- *bool* **is_raw_mode** ( ) const

Returns whether this color picker is in raw mode or not

- void **set_edit_alpha** (*bool* show)

Set true if you want the color to have an alpha channel (transparency), or false if you want a solid color.

- *bool* **is_editing_alpha** ( ) const

Returns whether the color has transparency or not.

- void **add_preset** (*Color* arg0)

Adds the current selected to color to a list of colors (presets), the presets will be displayed in the color picker and the user will be able to select them, notice that the presets list is only for this color picker.

9.58 ColorPickerButton

Inherits: *Button* < *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.58.1 Brief Description

Button that pops out a *ColorPicker*

9.58.2 Member Functions

void	*set_color* (*Color* color)
Color	*get_color* ( ) const
void	*set_edit_alpha* (*bool* show)
bool	*is_editing_alpha* ( ) const

9.58.3 Signals

- **color_changed** (*Color* color)

9.58.4 Description

Encapsulates a *ColorPicker* making it accesible by pressing a button, pressing the button will toggle the *ColorPicker* visibility

9.58.5 Member Function Description

- void **set_color** (*Color* color)

Sets the current color

- *Color* **get_color** ( ) const

Gets the current color

- void **set_edit_alpha** (*bool* show)

See *ColorPicker.set_edit_alpha*

- *bool* **is_editing_alpha** ( ) const

See ColorPicker.is_edit_alpha

9.59 ColorRamp

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.59.1 Brief Description

Color interpolator node

9.59.2 Member Functions

void	*add_point* (*float* offset, *Color* color)
void	*remove_point* (*int* offset)
void	*set_offset* (*int* point, *float* offset)
float	*get_offset* (*int* point) const
void	*set_color* (*int* point, *Color* color)
Color	*get_color* (*int* point) const
Color	*interpolate* (*float* offset)
int	*get_point_count* ( ) const
void	*set_offsets* (*RealArray* offsets)
RealArray	*get_offsets* ( ) const
void	*set_colors* (*ColorArray* colors)
ColorArray	*get_colors* ( ) const

9.59.3 Description

Given a set of colors, this node will interpolate them in order, meaning, that if you have color 1, color 2 and color3, the ramp will interpolate (generate the colors between two colors) from color 1 to color 2 and from color 2 to color 3. Initially the ramp will have 2 colors (black and white), one (black) at ramp lower offset offset 0 and the other (white) at the ramp higher offset 1.

9.59.4 Member Function Description

- void **add_point** (*float* offset, *Color* color)

Adds the specified color to the end of the ramp, with the specified offset

- void **remove_point** (*int* offset)

Removes the color at the index *offset*

- void **set_offset** (*int* point, *float* offset)

Sets the offset for the ramp color at index *point*

- *float* **get_offset** (*int* point) const

Returns the offset of the ramp color at index *point*

- void **set_color** (*int* point, *Color* color)

Sets the color of the ramp color at index *point*

- *Color* **get_color** (*int* point) const

Returns the color of the ramp color at index *point*

- *Color* **interpolate** (*float* offset)

Returns the interpolated color specified by *offset*

- *int* **get_point_count** ( ) const

Returns the number of colors in the ramp

- void **set_offsets** (*RealArray* offsets)

Sets the offset for the specified amount of elements. Calling this function with a different number of elements than previously defined causes the ramp to resize its colors and offsets array to accomodate the new elements, all new colors will be black by default.

- *RealArray* **get_offsets** ( ) const

Returns the offsets for the colors in this ramp

- void **set_colors** (*ColorArray* colors)

Sets the colors for the specified amount of elements. Calling this function with a different number of elements than previously defined causes the ramp to resize its colors and offsets array to accomodate the new elements.

- *ColorArray* **get_colors** ( ) const

Returns the colors in the ramp

9.60 ConcavePolygonShape

Inherits: *Shape* < *Resource* < *Reference* < *Object*

Category: Core

9.60.1 Brief Description

Concave polygon shape.

9.60.2 Member Functions

void	*set_faces* (*Vector3Array* faces)
Vector3Array	*get_faces* ( ) const

9.60.3 Description

Concave polygon shape resource, which can be set into a *PhysicsBody* or area. This shape is created by feeding a list of triangles.

9.60.4 Member Function Description

- void **set_faces** (*Vector3Array* faces)

Set the faces (an array of triangles).

- *Vector3Array* **get_faces** ( ) const

Return the faces (an array of triangles).

9.61 ConcavePolygonShape2D

Inherits: *Shape2D* < *Resource* < *Reference* < *Object*

Category: Core

9.61.1 Brief Description

Concave polygon 2D shape resource for physics.

9.61.2 Member Functions

void	*set_segments* (*Vector2Array* segments)
Vector2Array	*get_segments* ( ) const

9.61.3 Description

Concave polygon 2D shape resource for physics. It is made out of segments and is very optimal for complex polygonal concave collisions. It is really not advised to use for RigidBody nodes. A CollisionPolygon2D in convex decomposition mode (solids) or several convex objects are advised for that instead. Otherwise, a concave polygon 2D shape is better for static collisions.

The main difference between a *ConvexPolygonShape2D* and a *ConcavePolygonShape2D* is that a concave polygon assumes it is concave and uses a more complex method of collision detection, and a convex one forces itself to be convex in order to speed up collision detection.

9.61.4 Member Function Description

- void **set_segments** (*Vector2Array* segments)

Set the array of segments.

- *Vector2Array* **get_segments** ( ) const

Return the array of segments.

9.62 ConeTwistJoint

Inherits: *Joint* < *Spatial* < *Node* < *Object*

Category: Core

9.62.1 Brief Description

9.62.2 Member Functions

void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const

9.62.3 Numeric Constants

- **PARAM_SWING_SPAN = 0**
- **PARAM_TWIST_SPAN = 1**
- **PARAM_BIAS = 2**
- **PARAM_SOFTNESS = 3**
- **PARAM_RELAXATION = 4**
- **PARAM_MAX = 5**

9.62.4 Member Function Description

- void **set_param** (*int* param, *float* value)
- *float* **get_param** (*int* param) const

9.63 ConfigFile

Inherits: *Reference* < *Object*

Category: Core

9.63.1 Brief Description

Helper class to handle INI-style files.

9.63.2 Member Functions

void	*set_value* (*String* section, *String* key, var value)
Variant	*get_value* (*String* section, *String* key, var default=NULL) const
bool	*has_section* (*String* section) const
bool	*has_section_key* (*String* section, *String* key) const
StringArray	*get_sections* ( ) const
StringArray	*get_section_keys* (*String* section) const
Error	*load* (*String* path)
Error	*save* (*String* path)

9.63.3 Description

This helper class can be used to store Variant values on the filesystem using an INI-style formatting. The stored values as referenced by a section and a key. The stored data can be saved to or parsed from a file, though ConfigFile objects can also be used directly with accessing the filesystem.

The following example shows how to parse a INI-style file from the system, read its contents and store new values in it:

```
var config = ConfigFile.new()
var err = config.load("user://settings.cfg")
if err == OK: # if not, something went wrong with the file loading
    # Look for the display/width pair, and default to 1024 if missing
    var screen_width = get_value("display", "width", 1024)
    # Store a variable if and only it hasn't been defined yet
if not config.has_section_key("audio", "mute"):
    config.set_value("audio", "mute", false)
    # Save the changes by overwriting the previous file
config.save("user://settings.cfg")
```

9.63.4 Member Function Description

- void **set_value** (*String* section, *String* key, var value)

Assign a value to the specified key of the the specified section. If the section and/or the key do not exist, they are created. Passing a `NULL` value deletes the specified key if it exists (and deletes the section if it ends up empty once the key has been removed).

- Variant **get_value** (*String* section, *String* key, var default=NULL) const

Return the current value for the specified section and key. If the section and/or the key do not exist, the method returns the value of the optional *default* argument (and thus `NULL` if not specified).

- *bool* **has_section** (*String* section) const

Check if the specified section exists.

- *bool* **has_section_key** (*String* section, *String* key) const

Check if the specified section-key pair exists.

- *StringArray* **get_sections** ( ) const

Return an array of all defined section identifiers.

- *StringArray* **get_section_keys** (*String* section) const

Return an array of all defined key identifiers in the specified section.

- Error **load** (*String* path)

Load the config file specified as a parameter. The file's contents are parsed and loaded in the ConfigFile object from which the method was called. The return value is one of the OK, FAILED or ERR_* constants listed in *@Global Scope* (if the load was successful, it returns OK).

- Error **save** (*String* path)

Save the contents of the ConfigFile object to the file specified as a parameter. The output file uses an INI-style structure.

The return value is one of the OK, FAILED or ERR_* constants listed in *@Global Scope* (if the save was successful, it returns OK).

9.64 ConfirmationDialog

Inherits: *AcceptDialog* < *WindowDialog* < *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *EditorFileDialog*, *FileDialog*

Category: Core

9.64.1 Brief Description

Dialog for confirmation of actions.

9.64.2 Member Functions

Button	*get_cancel* ( )

9.64.3 Description

Dialog for confirmation of actions. This dialog inherits from *AcceptDialog*, but has by default an OK and Cancel button (in host OS order).

9.64.4 Member Function Description

- *Button* **get_cancel** ( )

Return the cancel button.

9.65 Container

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *PanelContainer*, *GridContainer*, *ScrollContainer*, *MarginContainer*, *CenterContainer*, *GraphNode*, *SplitContainer*, *BoxContainer*

Category: Core

9.65.1 Brief Description

Base node for containers.

9.65.2 Member Functions

void	*queue_sort* **( )**
void	*fit_child_in_rect* **(** *Control* child, *Rect2* rect **)**

9.65.3 Signals

- **sort_children ( )**

9.65.4 Numeric Constants

- **NOTIFICATION_SORT_CHILDREN = 50** — Notification for when sorting the children, it must be obeyed immediately.

9.65.5 Description

Base node for containers. A *Container* contains other controls and automatically arranges them in a certain way.

A Control can inherit this to create custom container classes.

9.65.6 Member Function Description

- void **queue_sort ( )**

Queue resort of the contained children. This is called automatically anyway, but can be called upon request.

- void **fit_child_in_rect (** *Control* child, *Rect2* rect **)**

Fit a child control in a given rect. This is mainly a helper for creating custom container classes.

9.66 Control

Inherits: *CanvasItem* **<** *Node* **<** *Object*

Inherited By: *Label*, *Tabs*, *TextureFrame*, *ButtonArray*, *VideoPlayer*, *LineEdit*, *Container*, *ReferenceFrame*, *Patch9Frame*, *TextEdit*, *BaseButton*, *Popup*, *Tree*, *Separator*, *Panel*, *TabContainer*, *Range*, *RichTextLabel*, *GraphEdit*, *ItemList*

Category: Core

9.66.1 Brief Description

Control is the base node for all the GUI components.

9.66.2 Member Functions

void	*_input_event* (*InputEvent* event) virtual
bool	*can_drop_data* (*Vector2* pos, var data) virtual
void	*drop_data* (*Vector2* pos, var data) virtual
Object	*get_drag_data* (*Vector2* pos) virtual
Vector2	get_minimum_size ( ) virtual
void	*accept_event* ( )
Vector2	get_minimum_size ( ) const
Vector2	*get_combined_minimum_size* ( ) const
void	*set_anchor* (*int* margin, *int* anchor_mode)
int	*get_anchor* (*int* margin) const
void	*set_margin* (*int* margin, *float* offset)
void	*set_anchor_and_margin* (*int* margin, *int* anchor_mode, *float* offset)
void	*set_begin* (*Vector2* pos)
void	*set_end* (*Vector2* pos)
void	*set_pos* (*Vector2* pos)
void	*set_size* (*Vector2* size)
void	*set_custom_minimum_size* (*Vector2* size)
void	*set_global_pos* (*Vector2* pos)
void	*set_rotation* (*float* radians)
void	*set_rotation_deg* (*float* degrees)
void	*set_scale* (*Vector2* scale)
float	*get_margin* (*int* margin) const
Vector2	*get_begin* ( ) const
Vector2	*get_end* ( ) const
Vector2	*get_pos* ( ) const
Vector2	*get_size* ( ) const
float	*get_rotation* ( ) const
float	*get_rotation_deg* ( ) const
Vector2	*get_scale* ( ) const
Vector2	*get_custom_minimum_size* ( ) const
Vector2	*get_parent_area_size* ( ) const
Vector2	*get_global_pos* ( ) const
Rect2	*get_rect* ( ) const
Rect2	*get_global_rect* ( ) const
void	*set_area_as_parent_rect* (*int* margin=0)
void	*show_modal* (*bool* exclusive=false)
void	*set_focus_mode* (*int* mode)
bool	*has_focus* ( ) const
void	*grab_focus* ( )
void	*release_focus* ( )
Control	*get_focus_owner* ( ) const
void	*set_h_size_flags* (*int* flags)
int	*get_h_size_flags* ( ) const
void	*set_stretch_ratio* (*float* ratio)
float	*get_stretch_ratio* ( ) const
void	*set_v_size_flags* (*int* flags)
int	*get_v_size_flags* ( ) const
void	*set_theme* (*Theme* theme)
Theme	*get_theme* ( ) const
void	*add_icon_override* (*String* name, *Texture* texture)

Continúa en la página siguiente

Tabla 9.9 – proviene de la página anterior

void	*add_shader_override* (*String* name, *Shader* shader)
void	*add_style_override* (*String* name, *StyleBox* stylebox)
void	*add_font_override* (*String* name, *Font* font)
void	*add_color_override* (*String* name, *Color* color)
void	*add_constant_override* (*String* name, *int* constant)
Texture	*get_icon* (*String* name, *String* type="") const
StyleBox	*get_stylebox* (*String* name, *String* type="") const
Font	*get_font* (*String* name, *String* type="") const
Color	*get_color* (*String* name, *String* type="") const
int	*get_constant* (*String* name, *String* type="") const
Control	*get_parent_control* ( ) const
void	*set_tooltip* (*String* tooltip)
String	*get_tooltip* (*Vector2* atpos=Vector2(0,0)) const
void	*set_default_cursor_shape* (*int* shape)
int	*get_default_cursor_shape* ( ) const
int	*get_cursor_shape* (*Vector2* pos=Vector2(0,0)) const
void	*set_focus_neighbour* (*int* margin, *NodePath* neighbour)
NodePath	*get_focus_neighbour* (*int* margin) const
void	*set_ignore_mouse* (*bool* ignore)
bool	*is_ignoring_mouse* ( ) const
void	*force_drag* (var data, *Object* preview)
void	*set_stop_mouse* (*bool* stop)
bool	*is_stopping_mouse* ( ) const
void	*grab_click_focus* ( )
void	*set_drag_preview* (*Control* control)
void	*warp_mouse* (*Vector2* to_pos)

9.66.3 Signals

- **focus_enter** ( )
- **mouse_enter** ( )
- **resized** ( )
- **minimum_size_changed** ( )
- **size_flags_changed** ( )
- **focus_exit** ( )
- **input_event** (*InputEvent* ev)
- **modal_close** ( )
- **mouse_exit** ( )

9.66.4 Numeric Constants

- **ANCHOR_BEGIN = 0** — X is relative to MARGIN_LEFT, Y is relative to MARGIN_TOP.
- **ANCHOR_END = 1** — X is relative to -MARGIN_RIGHT, Y is relative to -MARGIN_BOTTOM.
- **ANCHOR_RATIO = 2** — X and Y are a ratio (0 to 1) relative to the parent size 0 is left/top, 1 is right/bottom.
- **ANCHOR_CENTER = 3**

- **FOCUS_NONE = 0** — Control can't acquire focus.
- **FOCUS_CLICK = 1** — Control can acquire focus only if clicked.
- **FOCUS_ALL = 2** — Control can acquire focus if clicked, or by pressing TAB/Directionals in the keyboard from another Control.
- **NOTIFICATION_RESIZED = 40** — Control changed size (get_size() reports the new size).
- **NOTIFICATION_MOUSE_ENTER = 41** — Mouse pointer entered the area of the Control.
- **NOTIFICATION_MOUSE_EXIT = 42** — Mouse pointer exited the area of the Control.
- **NOTIFICATION_FOCUS_ENTER = 43** — Control gained focus.
- **NOTIFICATION_FOCUS_EXIT = 44** — Control lost focus.
- **NOTIFICATION_THEME_CHANGED = 45** — Theme changed. Redrawing is desired.
- **NOTIFICATION_MODAL_CLOSE = 46** — Modal control was closed.
- **CURSOR_ARROW = 0**
- **CURSOR_IBEAM = 1**
- **CURSOR_POINTING_HAND = 2**
- **CURSOR_CROSS = 3**
- **CURSOR_WAIT = 4**
- **CURSOR_BUSY = 5**
- **CURSOR_DRAG = 6**
- **CURSOR_CAN_DROP = 7**
- **CURSOR_FORBIDDEN = 8**
- **CURSOR_VSIZE = 9**
- **CURSOR_HSIZE = 10**
- **CURSOR_BDIAGSIZE = 11**
- **CURSOR_FDIAGSIZE = 12**
- **CURSOR_MOVE = 13**
- **CURSOR_VSPLIT = 14**
- **CURSOR_HSPLIT = 15**
- **CURSOR_HELP = 16**
- **SIZE_EXPAND = 1**
- **SIZE_FILL = 2**
- **SIZE_EXPAND_FILL = 3**

9.66.5 Description

Control is the base class Node for all the GUI components. Every GUI component inherits from it, directly or indirectly. In this way, sections of the scene tree made of contiguous control nodes, become user interfaces.

Controls are relative to the parent position and size by using anchors and margins. This ensures that they can adapt easily in most situation to changing dialog and screen sizes. When more flexibility is desired, *Container* derived nodes can be used.

Anchors work by defining which margin do they follow, and a value relative to it. Allowed anchoring modes are ANCHOR_BEGIN, where the margin is relative to the top or left margins of the parent (in pixels), ANCHOR_END for the right and bottom margins of the parent and ANCHOR_RATIO, which is a ratio from 0 to 1 in the parent range.

Input device events (*InputEvent*) are first sent to the root controls via the *Node._input*, which distribute it through the tree, then delivers them to the adequate one (under cursor or keyboard focus based) by calling *MainLoop._input_event*. There is no need to enable input processing on controls to receive such events. To ensure that no one else will receive the event (not even *Node._unhandled_input*), the control can accept it by calling *accept_event*.

Only one control can hold the keyboard focus (receiving keyboard events), for that the control must define the focus mode with *set_focus_mode*. Focus is lost when another control gains it, or the current focus owner is hidden.

It is sometimes desired for a control to ignore mouse/pointer events. This is often the case when placing other controls on top of a button, in such cases. Calling *set_ignore_mouse* enables this function.

Finally, controls are skinned according to a *Theme*. Setting a *Theme* on a control will propagate all the skinning down the tree. Optionally, skinning can be overridden per each control by calling the add_*_override functions, or from the editor.

9.66.6 Member Function Description

- void **_input_event** **(** *InputEvent* event **)** virtual

Called when an input event reaches the control.

- *bool* **can_drop_data** **(** *Vector2* pos, var data **)** virtual
- void **drop_data** **(** *Vector2* pos, var data **)** virtual
- *Object* **get_drag_data** **(** *Vector2* pos **)** virtual
- *Vector2* **get_minimum_size** **( )** virtual

Return the minimum size this Control can shrink to. A control will never be displayed or resized smaller than its minimum size.

- void **accept_event** **( )**

Handles the event, no other control will receive it and it will not be sent to nodes waiting on *Node._unhandled_input* or *Node._unhandled_key_input*.

- *Vector2* **get_minimum_size** **( )** const

Return the minimum size this Control can shrink to. A control will never be displayed or resized smaller than its minimum size.

- *Vector2* **get_combined_minimum_size** **( )** const
- void **set_anchor** **(** *int* margin, *int* anchor_mode **)**

Change the anchor (ANCHOR_BEGIN, ANCHOR_END, ANCHOR_RATIO) type for a margin (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM). Changing the anchor mode converts the current margin offset from the previous anchor mode to the new one, so margin offsets (*set_margin*) must be done after setting anchors, or at the same time (*set_anchor_and_margin*).

- *int* **get_anchor** **(** *int* margin **)** const

Return the anchor type (ANCHOR_BEGIN, ANCHOR_END, ANCHOR_RATIO) for a given margin (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM).

- void **set_margin** (*int* margin, *float* offset)

Set a margin offset. Margin can be one of (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM). Offset value being set depends on the anchor mode.

- void **set_anchor_and_margin** (*int* margin, *int* anchor_mode, *float* offset)

Change the anchor (ANCHOR_BEGIN, ANCHOR_END, ANCHOR_RATIO) type for a margin (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM), and also set its offset. This is a helper (see *set_anchor* and *set_margin*).

- void **set_begin** (*Vector2* pos)

Sets MARGIN_LEFT and MARGIN_TOP at the same time. This is a helper (see *set_margin*).

- void **set_end** (*Vector2* pos)

Sets MARGIN_RIGHT and MARGIN_BOTTOM at the same time. This is a helper (see *set_margin*).

- void **set_pos** (*Vector2* pos)

Move the Control to a new position, relative to the top-left corner of the parent Control, changing all margins if needed and without changing current anchor mode. This is a helper (see *set_margin*).

- void **set_size** (*Vector2* size)

Changes MARGIN_RIGHT and MARGIN_BOTTOM to fit a given size. This is a helper (see *set_margin*).

- void **set_custom_minimum_size** (*Vector2* size)
- void **set_global_pos** (*Vector2* pos)

Move the Control to a new position, relative to the top-left corner of the *window* Control, and without changing current anchor mode. (see *set_margin*).

- void **set_rotation** (*float* radians)
- void **set_rotation_deg** (*float* degrees)
- void **set_scale** (*Vector2* scale)
- *float* **get_margin** (*int* margin) const

Return a margin offset. Margin can be one of (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM). Offset value being returned depends on the anchor mode.

- *Vector2* **get_begin** ( ) const
- *Vector2* **get_end** ( ) const

Returns MARGIN_LEFT and MARGIN_TOP at the same time. This is a helper (see *set_margin*).

- *Vector2* **get_pos** ( ) const

Returns the Control position, relative to the top-left corner of the parent Control and independent of the anchor mode.

- *Vector2* **get_size** ( ) const

Returns the size of the Control, computed from all margins, however the size returned will **never be smaller than the minimum size reported by :ref:‘get_minimum_size<class_Control_get_minimum_size>‘**. This means that even if end position of the Control rectangle is smaller than the begin position, the Control will still display and interact correctly. (see description, get_minimum_size, *set_margin*, *set_anchor*).

- *float* **get_rotation** ( ) const
- *float* **get_rotation_deg** ( ) const
- *Vector2* **get_scale** ( ) const

- *Vector2* **get_custom_minimum_size** ( ) const
- *Vector2* **get_parent_area_size** ( ) const
- *Vector2* **get_global_pos** ( ) const

Returns the Control position, relative to the top-left corner of the parent Control and independent of the anchor mode.

- *Rect2* **get_rect** ( ) const

Return position and size of the Control, relative to the top-left corner of the parent Control. This is a helper (see *get_pos*, *get_size*).

- *Rect2* **get_global_rect** ( ) const

Return position and size of the Control, relative to the top-left corner of the *window* Control. This is a helper (see *get_global_pos*, *get_size*).

- void **set_area_as_parent_rect** (*int* margin=0)

Change all margins and anchors, so this Control always takes up the same area as the parent Control. This is a helper (see *set_anchor*, *set_margin*).

- void **show_modal** (*bool* exclusive=false)

Display a Control as modal. Control must be a subwindow. Modal controls capture the input signals until closed or the area outside them is accessed. When a modal control loses focus, or the ESC key is pressed, they automatically hide. Modal controls are used extensively for popup dialogs and menus.

- void **set_focus_mode** (*int* mode)

Set the focus access mode for the control (FOCUS_NONE, FOCUS_CLICK, FOCUS_ALL). Only one Control can be focused at the same time, and it will receive keyboard signals.

- *bool* **has_focus** ( ) const

Return whether the Control is the current focused control (see *set_focus_mode*).

- void **grab_focus** ( )

Steal the focus from another control and become the focused control (see *set_focus_mode*).

- void **release_focus** ( )

Give up the focus, no other control will be able to receive keyboard input.

- *Control* **get_focus_owner** ( ) const

Return which control is owning the keyboard focus, or null if no one.

- void **set_h_size_flags** (*int* flags)

Hint for containers, set horizontal positioning flags.

- *int* **get_h_size_flags** ( ) const

Hint for containers, return horizontal positioning flags.

- void **set_stretch_ratio** (*float* ratio)

Hint for containers, set the stretch ratio. This value is relative to other stretch ratio, so if this control has 2 and another has 1, this one will be twice as big.

- *float* **get_stretch_ratio** ( ) const

Hint for containers, return the stretch ratio. This value is relative to other stretch ratio, so if this control has 2 and another has 1, this one will be twice as big.

- void **set_v_size_flags** (*int* flags)

Hint for containers, set vertical positioning flags.

- *int* **get_v_size_flags ( )** const

Hint for containers, return vertical positioning flags.

- void **set_theme (*Theme* theme)**

Override whole the *Theme* for this Control and all its children controls.

- *Theme* **get_theme ( )** const

Return a *Theme* override, if one exists (see *set_theme*).

- void **add_icon_override (*String* name, *Texture* texture)**

Override a single icon (*Texture*) in the theme of this Control. If texture is empty, override is cleared.

- void **add_shader_override (*String* name, *Shader* shader)**
- void **add_style_override (*String* name, *StyleBox* stylebox)**

Override a single stylebox (*Stylebox*) in the theme of this Control. If stylebox is empty, override is cleared.

- void **add_font_override (*String* name, *Font* font)**

Override a single font (font) in the theme of this Control. If font is empty, override is cleared.

- void **add_color_override (*String* name, *Color* color)**
- void **add_constant_override (*String* name, *int* constant)**

Override a single constant (integer) in the theme of this Control. If constant equals Theme.INVALID_CONSTANT, override is cleared.

- *Texture* **get_icon (*String* name, *String* type="")** const
- *StyleBox* **get_stylebox (*String* name, *String* type="")** const
- *Font* **get_font (*String* name, *String* type="")** const
- *Color* **get_color (*String* name, *String* type="")** const
- *int* **get_constant (*String* name, *String* type="")** const
- *Control* **get_parent_control ( )** const
- void **set_tooltip (*String* tooltip)**

Set a tooltip, which will appear when the cursor is resting over this control.

- *String* **get_tooltip (*Vector2* atpos=Vector2(0,0))** const

Return the tooltip, which will appear when the cursor is resting over this control.

- void **set_default_cursor_shape (*int* shape)**

Set the default cursor shape for this control. See enum CURSOR_* for the list of shapes.

- *int* **get_default_cursor_shape ( )** const

Return the default cursor shape for this control. See enum CURSOR_* for the list of shapes.

- *int* **get_cursor_shape (*Vector2* pos=Vector2(0,0))** const

Return the cursor shape at a certain position in the control.

- void **set_focus_neighbour (*int* margin, *NodePath* neighbour)**

Force a neighbour for moving the input focus to. When pressing TAB or directional/joypad directions focus is moved to the next control in that direction. However, the neighbour to move to can be forced with this function.

- *NodePath* **get_focus_neighbour** (*int* margin) const

Return the forced neighbour for moving the input focus to. When pressing TAB or directional/joypad directions focus is moved to the next control in that direction. However, the neighbour to move to can be forced with this function.

- void **set_ignore_mouse** (*bool* ignore)

Ignore mouse events on this control (even touchpad events send mouse events).

- *bool* **is_ignoring_mouse** ( ) const

Return if the control is ignoring mouse events (even touchpad events send mouse events).

- void **force_drag** (var data, *Object* preview)
- void **set_stop_mouse** (*bool* stop)
- *bool* **is_stopping_mouse** ( ) const
- void **grab_click_focus** ( )
- void **set_drag_preview** (*Control* control)
- void **warp_mouse** (*Vector2* to_pos)

9.67 ConvexPolygonShape

Inherits: *Shape* < *Resource* < *Reference* < *Object*

Category: Core

9.67.1 Brief Description

Convex Polygon Shape.

9.67.2 Member Functions

void	*set_points* (*Vector3Array* points)
Vector3Array	*get_points* ( ) const

9.67.3 Description

Convex polygon shape resource, which can be set into a *PhysicsBody* or area.

9.67.4 Member Function Description

- void **set_points** (*Vector3Array* points)
- *Vector3Array* **get_points** ( ) const

9.68 ConvexPolygonShape2D

Inherits: *Shape2D* < *Resource* < *Reference* < *Object*

Category: Core

9.68.1 Brief Description

Convex Polygon Shape for 2D physics.

9.68.2 Member Functions

void	*set_point_cloud* (*Vector2Array* point_cloud)
void	*set_points* (*Vector2Array* points)
Vector2Array	*get_points* ( ) const

9.68.3 Description

Convex Polygon Shape for 2D physics. A convex polygon, whatever its shape, is internally decomposed into as many convex polygons as needed to ensure all collision checks against it are always done on convex polygons (which are faster to check).

The main difference between a *ConvexPolygonShape2D* and a *ConcavePolygonShape2D* is that a concave polygon assumes it is concave and uses a more complex method of collision detection, and a convex one forces itself to be convex in order to speed up collision detection.

9.68.4 Member Function Description

- void **set_point_cloud** (*Vector2Array* point_cloud)

Currently, this method does nothing.

- void **set_points** (*Vector2Array* points)

Set a list of points in either clockwise or counter clockwise order, forming a convex polygon.

- *Vector2Array* **get_points** ( ) const

Return a list of points in either clockwise or counter clockwise order, forming a convex polygon.

9.69 CubeMap

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.69.1 Brief Description

9.69.2 Member Functions

int	*get_width* ( ) const
int	*get_height* ( ) const
RID	*get_rid* ( ) const
void	*set_flags* (*int* flags)
int	*get_flags* ( ) const
void	*set_side* (*int* side, *Image* image)
Image	*get_side* (*int* side) const
void	*set_storage* (*int* mode)
int	*get_storage* ( ) const
void	*set_lossy_storage_quality* (*float* quality)
float	*get_lossy_storage_quality* ( ) const

9.69.3 Numeric Constants

- **STORAGE_RAW = 0**
- **STORAGE_COMPRESS_LOSSY = 1**
- **STORAGE_COMPRESS_LOSSLESS = 2**
- **SIDE_LEFT = 0**
- **SIDE_RIGHT = 1**
- **SIDE_BOTTOM = 2**
- **SIDE_TOP = 3**
- **SIDE_FRONT = 4**
- **SIDE_BACK = 5**
- **FLAG_MIPMAPS = 1**
- **FLAG_REPEAT = 2**
- **FLAG_FILTER = 4**
- **FLAGS_DEFAULT = 7**

9.69.4 Member Function Description

- *int* **get_width** ( ) const
- *int* **get_height** ( ) const
- *RID* **get_rid** ( ) const
- void **set_flags** (*int* flags)
- *int* **get_flags** ( ) const
- void **set_side** (*int* side, *Image* image)
- *Image* **get_side** (*int* side) const
- void **set_storage** (*int* mode)

- *int* **get_storage** ( ) const
- void **set_lossy_storage_quality** (*float* quality)
- *float* **get_lossy_storage_quality** ( ) const

9.70 Curve2D

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.70.1 Brief Description

Describes a Bezier curve in 2D space.

9.70.2 Member Functions

int	*get_point_count* ( ) const
void	*add_point* (*Vector2* pos, *Vector2* in=Vector2(0,0), *Vector2* out=Vector2(0,0), *int* atpos=-1)
void	*set_point_pos* (*int* idx, *Vector2* pos)
Vector2	*get_point_pos* (*int* idx) const
void	*set_point_in* (*int* idx, *Vector2* pos)
Vector2	*get_point_in* (*int* idx) const
void	*set_point_out* (*int* idx, *Vector2* pos)
Vector2	*get_point_out* (*int* idx) const
void	*remove_point* (*int* idx)
Vector2	*interpolate* (*int* idx, *float* t) const
Vector2	*interpolatef* (*float* fofs) const
void	*set_bake_interval* (*float* distance)
float	*get_bake_interval* ( ) const
float	*get_baked_length* ( ) const
Vector2	*interpolate_baked* (*float* offset, *bool* cubic=false) const
Vector2Array	*get_baked_points* ( ) const
Vector2Array	*tesselate* (*int* max_stages=5, *float* tolerance_degrees=4) const

9.70.3 Description

This class describes a Bezier curve in 2D space. It is mainly used to give a shape to a *Path2D*, but can be manually sampled for other purposes.

It keeps a cache of precalculated points along the curve, to speed further calculations up.

9.70.4 Member Function Description

- *int* **get_point_count** ( ) const

Returns the number of points describing the curve.

- void **add_point** (*Vector2* pos, *Vector2* in=Vector2(0,0), *Vector2* out=Vector2(0,0), *int* atpos=-1)

Adds a point to a curve, at position "pos", with control points "in" and "out".

If "atpos" is given, the point is inserted before the point number "atpos", moving that point (and every point after) after the inserted point. If "atpos" is not given, or is an illegal value (atpos <0 or atpos >= *get_point_count*), the point will be appended at the end of the point list.

- void **set_point_pos** (*int* idx, *Vector2* pos)

Sets the position for the vertex "idx". If the index is out of bounds, the function sends an error to the console.

- *Vector2* **get_point_pos** (*int* idx) const

Returns the position of the vertex "idx". If the index is out of bounds, the function sends an error to the console, and returns (0, 0).

- void **set_point_in** (*int* idx, *Vector2* pos)

Sets the position of the control point leading to the vertex "idx". If the index is out of bounds, the function sends an error to the console.

- *Vector2* **get_point_in** (*int* idx) const

Returns the position of the control point leading to the vertex "idx". If the index is out of bounds, the function sends an error to the console, and returns (0, 0).

- void **set_point_out** (*int* idx, *Vector2* pos)

Sets the position of the control point leading out of the vertex "idx". If the index is out of bounds, the function sends an error to the console.

- *Vector2* **get_point_out** (*int* idx) const

Returns the position of the control point leading out of the vertex "idx". If the index is out of bounds, the function sends an error to the console, and returns (0, 0).

- void **remove_point** (*int* idx)

Deletes the point "idx" from the curve. Sends an error to the console if "idx" is out of bounds.

- *Vector2* **interpolate** (*int* idx, *float* t) const

Returns the position between the vertex "idx" and the vertex "idx"+1, where "t" controls if the point is the first vertex (t = 0.0), the last vertex (t = 1.0), or in between. Values of "t" outside the range (0.0 >= t <=1) give strange, but predictable results.

If "idx" is out of bounds it is truncated to the first or last vertex, and "t" is ignored. If the curve has no points, the function sends an error to the console, and returns (0, 0).

- *Vector2* **interpolatef** (*float* fofs) const

Returns the position at the vertex "fofs". It calls *interpolate* using the integer part of fofs as "idx", and its fractional part as "t".

- void **set_bake_interval** (*float* distance)

Sets the distance in pixels between two adjacent cached points. Changing it forces the cache to be recomputed the next time a xxx_baked_xxx function is called. The less distance, the more points the cache will have, and the more memory it will consume, so use with care.

- *float* **get_bake_interval** ( ) const

Returns the distance between two adjacent cached points.

- *float* **get_baked_length** ( ) const

Returns the total length of the curve, based on the cached points. Given enough density (see *set_bake_interval*), it should be approximate enough.

- *Vector2* **interpolate_baked** (*float* offset, *bool* cubic=false) const

Returns a point within the curve at position "offset", where "offset" is measured as a pixel distance along the curve.

To do that, it finds the two cached points where the "offset" lies between, then interpolates the values. This interpolation is cubic if "cubic" is set to true, or linear if set to false.

Cubic interpolation tends to follow the curves better, but linear is faster (and often, precise enough).

- *Vector2Array* **get_baked_points** ( ) const

Returns the cache of points as a *Vector2Array*.

- *Vector2Array* **tesselate** (*int* max_stages=5, *float* tolerance_degrees=4) const

Returns a list of points along the curve, with a curvature controlled point density. That is, the curvier parts will have more points than the straighter parts.

This approximation makes straight segments between each point, then subdivides those segments until the resulting shape is similar enough.

"max_stages" controls how many subdivisions a curve segment may face before it is considered approximate enough. Each subdivision splits the segment in half, so the default 5 stages may mean up to 32 subdivisions per curve segment. Increase with care!

"tolerance_degrees" controls how many degrees the midpoint of a segment may deviate from the real curve, before the segment has to be subdivided.

9.71 Curve3D

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.71.1 Brief Description

Describes a Bezier curve in 3D space.

9.71.2 Member Functions

int	*get_point_count* ( ) const
void	*add_point* (*Vector3* pos, *Vector3* in=Vector3(0, 0, 0), *Vector3* out=Vector3(0, 0, 0), *int* atpos=-1)
void	*set_point_pos* (*int* idx, *Vector3* pos)
Vector3	*get_point_pos* (*int* idx) const
void	*set_point_tilt* (*int* idx, *float* tilt)
float	*get_point_tilt* (*int* idx) const
void	*set_point_in* (*int* idx, *Vector3* pos)
Vector3	*get_point_in* (*int* idx) const
void	*set_point_out* (*int* idx, *Vector3* pos)
Vector3	*get_point_out* (*int* idx) const
void	*remove_point* (*int* idx)
Vector3	*interpolate* (*int* idx, *float* t) const
Vector3	*interpolatef* (*float* fofs) const
void	*set_bake_interval* (*float* distance)
float	*get_bake_interval* ( ) const
float	*get_baked_length* ( ) const
Vector3	*interpolate_baked* (*float* offset, *bool* cubic=false) const
Vector3Array	*get_baked_points* ( ) const
RealArray	*get_baked_tilts* ( ) const
Vector3Array	*tesselate* (*int* max_stages=5, *float* tolerance_degrees=4) const

9.71.3 Description

This class describes a Bezier curve in 3D space. It is mainly used to give a shape to a *Path*, but can be manually sampled for other purposes.

It keeps a cache of precalculated points along the curve, to speed further calculations up.

9.71.4 Member Function Description

- *int* **get_point_count** ( ) const

Returns the number of points describing the curve.

- void **add_point** (*Vector3* pos, *Vector3* in=Vector3(0, 0, 0), *Vector3* out=Vector3(0, 0, 0), *int* atpos=-1)

Adds a point to a curve, at position "pos", with control points "in" and "out".

If "atpos" is given, the point is inserted before the point number "atpos", moving that point (and every point after) after the inserted point. If "atpos" is not given, or is an illegal value (atpos <0 or atpos >= *get_point_count*), the point will be appended at the end of the point list.

- void **set_point_pos** (*int* idx, *Vector3* pos)

Sets the position for the vertex "idx". If the index is out of bounds, the function sends an error to the console.

- *Vector3* **get_point_pos** (*int* idx) const

Returns the position of the vertex "idx". If the index is out of bounds, the function sends an error to the console, and returns (0, 0, 0).

- void **set_point_tilt** (*int* idx, *float* tilt)

Sets the tilt angle in radians for the point "idx". If the index is out of bounds, the function sends an error to the console.

The tilt controls the rotation along the look-at axis an object traveling the path would have. In the case of a curve controlling a *PathFollow*, this tilt is an offset over the natural tilt the PathFollow calculates.

- *float* **get_point_tilt** (*int* idx) const

Returns the tilt angle in radians for the point "idx". If the index is out of bounds, the function sends an error to the console, and returns 0.

- void **set_point_in** (*int* idx, *Vector3* pos)

Sets the position of the control point leading to the vertex "idx". If the index is out of bounds, the function sends an error to the console.

- *Vector3* **get_point_in** (*int* idx) const

Returns the position of the control point leading to the vertex "idx". If the index is out of bounds, the function sends an error to the console, and returns (0, 0, 0).

- void **set_point_out** (*int* idx, *Vector3* pos)

Sets the position of the control point leading out of the vertex "idx". If the index is out of bounds, the function sends an error to the console.

- *Vector3* **get_point_out** (*int* idx) const

Returns the position of the control point leading out of the vertex "idx". If the index is out of bounds, the function sends an error to the console, and returns (0, 0, 0).

- void **remove_point** (*int* idx)

Deletes the point "idx" from the curve. Sends an error to the console if "idx" is out of bounds.

- *Vector3* **interpolate** (*int* idx, *float* t) const

Returns the position between the vertex "idx" and the vertex "idx"+1, where "t" controls if the point is the first vertex (t = 0.0), the last vertex (t = 1.0), or in between. Values of "t" outside the range (0.0 >= t <=1) give strange, but predictable results.

If "idx" is out of bounds it is truncated to the first or last vertex, and "t" is ignored. If the curve has no points, the function sends an error to the console, and returns (0, 0, 0).

- *Vector3* **interpolatef** (*float* fofs) const

Returns the position at the vertex "fofs". It calls *interpolate* using the integer part of fofs as "idx", and its fractional part as "t".

- void **set_bake_interval** (*float* distance)

Sets the distance in 3D units between two adjacent cached points. Changing it forces the cache to be recomputed the next time a xxx_baked_xxx function is called. The less distance, the more points the cache will have, and the more memory it will consume, so use with care.

- *float* **get_bake_interval** ( ) const

Returns the distance between two adjacent cached points.

- *float* **get_baked_length** ( ) const

Returns the total length of the curve, based on the cached points. Given enough density (see *set_bake_interval*), it should be approximate enough.

- *Vector3* **interpolate_baked** (*float* offset, *bool* cubic=false) const

Returns a point within the curve at position "offset", where "offset" is measured as a distance in 3D units along the curve.

To do that, it finds the two cached points where the "offset" lies between, then interpolates the values. This interpolation is cubic if "cubic" is set to true, or linear if set to false.

Cubic interpolation tends to follow the curves better, but linear is faster (and often, precise enough).

- *Vector3Array* **get_baked_points** ( ) const

Returns the cache of points as a *Vector3Array*.

- *RealArray* **get_baked_tilts** ( ) const

Returns the cache of tilts as a *RealArray*.

- *Vector3Array* **tesselate** (*int* max_stages=5, *float* tolerance_degrees=4) const

Returns a list of points along the curve, with a curvature controlled point density. That is, the curvier parts will have more points than the straighter parts.

This approximation makes straight segments between each point, then subdivides those segments until the resulting shape is similar enough.

"max_stages" controls how many subdivisions a curve segment may face before it is considered approximate enough. Each subdivision splits the segment in half, so the default 5 stages may mean up to 32 subdivisions per curve segment. Increase with care!

"tolerance_degrees" controls how many degrees the midpoint of a segment may deviate from the real curve, before the segment has to be subdivided.

9.72 DampedSpringJoint2D

Inherits: *Joint2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.72.1 Brief Description

Damped spring constraint for 2D physics.

9.72.2 Member Functions

void	*set_length* (*float* length)
float	*get_length* ( ) const
void	*set_rest_length* (*float* rest_length)
float	*get_rest_length* ( ) const
void	*set_stiffness* (*float* stiffness)
float	*get_stiffness* ( ) const
void	*set_damping* (*float* damping)
float	*get_damping* ( ) const

9.72.3 Description

Damped spring constraint for 2D physics. This resembles a spring joint that always wants to go back to a given length.

9.72.4 Member Function Description

- void **set_length** (*float* length)

Set the maximum length of the spring joint.

- *float* **get_length** ( ) const

Return the maximum length of the spring joint.

- void **set_rest_length** (*float* rest_length)

Set the resting length of the spring joint. The joint will always try to go to back this length when pulled apart.

- *float* **get_rest_length** ( ) const

Return the resting length of the spring joint. The joint will always try to go to back this length when pulled apart.

- void **set_stiffness** (*float* stiffness)

Set the stiffness of the spring joint. The joint applies a force equal to the stiffness times the distance from its resting length.

- *float* **get_stiffness** ( ) const

Return the stiffness of the spring joint. The joint applies a force equal to the stiffness times the distance from its resting length.

- void **set_damping** (*float* damping)

Set the damping ratio of the spring joint. A value of 0 indicates an undamped spring, while 1 causes the system to reach equilibrium as fast as possible (critical damping).

- *float* **get_damping** ( ) const

Return the damping ratio of the spring joint. A value of 0 indicates an undamped spring, while 1 causes the system to reach equilibrium as fast as possible (critical damping).

9.73 Dictionary

Category: Built-In Types

9.73.1 Brief Description

Dictionary type.

9.73.2 Member Functions

void	*clear* ( )
bool	*empty* ( )
void	*erase* (var value)
bool	*has* (var value)
bool	*has_all* (*Array* values)
int	*hash* ( )
Array	*keys* ( )
int	*parse_json* (*String* json)
int	*size* ( )
String	*to_json* ( )

9.73.3 Description

Dictionary type. Associative container which contains values referenced by unique keys. Dictionaries are always passed by reference.

9.73.4 Member Function Description

- void **clear** **(** **)**

Clear the dictionary, removing all key/value pairs.

- *bool* **empty** **(** **)**

Return true if the dictionary is empty.

- void **erase** **(** var value **)**

Erase a dictionary key/value pair by key.

- *bool* **has** **(** var value **)**

Return true if the dictionary has a given key.

- *bool* **has_all** **(** *Array* values **)**
- *int* **hash** **(** **)**

Return a hashed integer value representing the dictionary contents.

- *Array* **keys** **(** **)**

Return the list of keys in the dictionary.

- *int* **parse_json** **(** *String* json **)**

Parse json text to the dictionary. Return OK when successed or the error code when failed.

- *int* **size** **(** **)**

Return the size of the dictionary (in pairs).

- *String* **to_json** **(** **)**

Return the dictionary as json text.

9.74 DirectionalLight

Inherits: *Light* **<** *VisualInstance* **<** *Spatial* **<** *Node* **<** *Object*

Category: Core

9.74.1 Brief Description

Directional Light, such as the Sun or the Moon.

9.74.2 Member Functions

void	*set_shadow_mode* (*int* mode)
int	*get_shadow_mode* ( ) const
void	*set_shadow_param* (*int* param, *float* value)
float	*get_shadow_param* (*int* param) const

9.74.3 Numeric Constants

- **SHADOW_ORTHOGONAL = 0**
- **SHADOW_PERSPECTIVE = 1**
- **SHADOW_PARALLEL_2_SPLITS = 2**
- **SHADOW_PARALLEL_4_SPLITS = 3**
- **SHADOW_PARAM_MAX_DISTANCE = 0**
- **SHADOW_PARAM_PSSM_SPLIT_WEIGHT = 1**
- **SHADOW_PARAM_PSSM_ZOFFSET_SCALE = 2**

9.74.4 Description

A DirectionalLight is a type of *Light* node that emits light constantly in one direction (the negative z axis of the node). It is used lights with strong intensity that are located far away from the scene to model sunlight or moonlight. The worldspace location of the DirectionalLight transform (origin) is ignored, only the basis is used do determine light direction.

9.74.5 Member Function Description

- void **set_shadow_mode** (*int* mode)
- *int* **get_shadow_mode** ( ) const
- void **set_shadow_param** (*int* param, *float* value)
- *float* **get_shadow_param** (*int* param) const

9.75 Directory

Inherits: *Reference* < *Object*

Category: Core

9.75.1 Brief Description

Type used to handle the filesystem.

9.75.2 Member Functions

Error	*open* (*String* path)
bool	*list_dir_begin* ( )
String	*get_next* ( )
bool	*current_is_dir* ( ) const
void	*list_dir_end* ( )
int	*get_drive_count* ( )
String	*get_drive* (*int* idx)
Error	*change_dir* (*String* todir)
String	*get_current_dir* ( )
Error	*make_dir* (*String* path)
Error	*make_dir_recursive* (*String* path)
bool	*file_exists* (*String* path)
bool	*dir_exists* (*String* path)
int	*get_space_left* ( )
Error	*copy* (*String* from, *String* to)
Error	*rename* (*String* from, *String* to)
Error	*remove* (*String* path)

9.75.3 Description

Directory type. It is used to manage directories and their content (not restricted to the project folder).

Here is an example on how to iterate through the files of a directory:

```
func dir_contents(path):
    var dir = Directory.new()
    if dir.open(path) == OK:
        dir.list_dir_begin()
        var file_name = dir.get_next()
        while (file_name != ""):
            if dir.current_is_dir():
                print("Found directory: " + file_name)
            else:
                print("Found file: " + file_name)
            file_name = dir.get_next()
    else:
        print("An error occurred when trying to access the path.")
```

9.75.4 Member Function Description

- Error **open** (*String* path)

Open an existing directory of the filesystem. The *path* argument can be within the project tree (`res://folder`), the user directory (`user://folder`) or an absolute path of the user filesystem (e.g. `/tmp/folder` or `C:\tmp\folder`).

The method returns one of the error code constants defined in *@Global Scope* (OK or ERR_*).

- *bool* **list_dir_begin** ( )

Initialise the stream used to list all files and directories using the *get_next* function, closing the current opened stream if needed. Once the stream has been processed, it should typically be closed with *list_dir_end*.

Return false if the stream could not be initialised.

- *String* **get_next ( )**

Return the next element (file or directory) in the current directory (including `.` and `..`). The name of the file or directory is returned (and not its full path). Once the stream has been fully processed, the method returns an empty String and closes the stream automatically (i.e. *list_dir_end* would not be mandatory in such a case).

- *bool* **current_is_dir ( )** const

Return whether the current item processed with the last *get_next* call is a directory (`.` and `..` are considered directories).

- void **list_dir_end ( )**

Close the current stream opened with *list_dir_begin* (whether it has been fully processed with *get_next* or not does not matter).

- *int* **get_drive_count ( )**

On Windows, return the number of drives (partitions) mounted on the current filesystem. On other platforms, the method returns 0.

- *String* **get_drive (*int* idx)**

On Windows, return the name of the drive (partition) passed as an argument (e.g. `C:`). On other platforms, or if the requested drive does not existed, the method returns an empty String.

- Error **change_dir (*String* todir)**

Change the currently opened directory to the one passed as an argument. The argument can be relative to the current directory (e.g. `newdir` or `../newdir`), or an absolute path (e.g. `/tmp/newdir` or `res://somedir/newdir`).

The method returns one of the error code constants defined in *@Global Scope* (OK or ERR_*).

- *String* **get_current_dir ( )**

Return the absolute path to the currently opened directory (e.g. `res://folder` or `C:\tmp\folder`).

- Error **make_dir (*String* path)**

Create a directory. The argument can be relative to the current directory, or an absolute path. The target directory should be placed in an already existing directory (to create the full path recursively, see *make_dir_recursive*).

The method returns one of the error code constants defined in *@Global Scope* (OK, FAILED or ERR_*).

- Error **make_dir_recursive (*String* path)**

Create a target directory and all necessary intermediate directories in its path, by calling *make_dir* recursively. The argument can be relative to the current directory, or an absolute path.

Returns one of the error code constants defined in *@Global Scope* (OK, FAILED or ERR_*).

- *bool* **file_exists (*String* path)**

Return whether the target file exists. The argument can be relative to the current directory, or an absolute path.

- *bool* **dir_exists (*String* path)**

Return whether the target directory exists. The argument can be relative to the current directory, or an absolute path.

- *int* **get_space_left ( )**

On Unix desktop systems, return the available space on the current directory's disk. On other platforms, this information is not available and the method returns 0 or -1.

- Error **copy (*String* from, *String* to)**

Copy the *from* file to the *to* destination. Both arguments should be paths to files, either relative or absolute. If the destination file exists and is not access-protected, it will be overwritten.

Returns one of the error code constants defined in *@Global Scope* (OK, FAILED or ERR_*).

- Error **rename** (*String* from, *String* to)

Rename (move) the *from* file to the *to* destination. Both arguments should be paths to files, either relative or absolute. If the destination file exists and is not access-protected, it will be overwritten.

Returns one of the error code constants defined in *@Global Scope* (OK or FAILED).

- Error **remove** (*String* path)

Delete the target file or an empty directory. The argument can be relative to the current directory, or an absolute path. If the target directory is not empty, the operation will fail.

Returns one of the error code constants defined in *@Global Scope* (OK or FAILED).

9.76 EditorFileDialog

Inherits: *ConfirmationDialog* < *AcceptDialog* < *WindowDialog* < *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.76.1 Brief Description

9.76.2 Member Functions

void	*clear_filters* ( )
void	*add_filter* (*String* filter)
String	*get_current_dir* ( ) const
String	*get_current_file* ( ) const
String	*get_current_path* ( ) const
void	*set_current_dir* (*String* dir)
void	*set_current_file* (*String* file)
void	*set_current_path* (*String* path)
void	*set_mode* (*int* mode)
int	*get_mode* ( ) const
VBoxContainer	*get_vbox* ( )
void	*set_access* (*int* access)
int	*get_access* ( ) const
void	*set_show_hidden_files* (*bool* show)
bool	*is_showing_hidden_files* ( ) const
void	*set_display_mode* (*int* mode)
int	*get_display_mode* ( ) const
void	*invalidate* ( )

9.76.3 Signals

- **files_selected** (*StringArray* paths)
- **dir_selected** (*String* dir)
- **file_selected** (*String* path)

9.76.4 Numeric Constants

- **MODE_OPEN_FILE = 0**
- **MODE_OPEN_FILES = 1**
- **MODE_OPEN_DIR = 2**
- **MODE_SAVE_FILE = 3**
- **ACCESS_RESOURCES = 0**
- **ACCESS_USERDATA = 1**
- **ACCESS_FILESYSTEM = 2**

9.76.5 Member Function Description

- void **clear_filters** ( )
- void **add_filter** (*String* filter)
- *String* **get_current_dir** ( ) const
- *String* **get_current_file** ( ) const
- *String* **get_current_path** ( ) const
- void **set_current_dir** (*String* dir)
- void **set_current_file** (*String* file)
- void **set_current_path** (*String* path)
- void **set_mode** (*int* mode)
- *int* **get_mode** ( ) const
- *VBoxContainer* **get_vbox** ( )
- void **set_access** (*int* access)
- *int* **get_access** ( ) const
- void **set_show_hidden_files** (*bool* show)
- *bool* **is_showing_hidden_files** ( ) const
- void **set_display_mode** (*int* mode)
- *int* **get_display_mode** ( ) const
- void **invalidate** ( )

9.77 EditorImportPlugin

Inherits: *Reference* < *Object*

Category: Core

9.77.1 Brief Description

9.77.2 Member Functions

RawArray	*custom_export* (*String* path) virtual
String	*get_name* ( ) virtual
String	*get_visible_name* ( ) virtual
int	*import* (*String* path, ResourceImportMetaData from) virtual
void	*import_dialog* (*String* from) virtual

9.77.3 Member Function Description

- *RawArray* **custom_export** (*String* path) virtual
- *String* **get_name** ( ) virtual
- *String* **get_visible_name** ( ) virtual
- *int* **import** (*String* path, ResourceImportMetaData from) virtual
- void **import_dialog** (*String* from) virtual

9.78 EditorPlugin

Inherits: *Node* < *Object*

Category: Core

9.78.1 Brief Description

9.78.2 Member Functions

void	*apply_changes* ( ) virtual
void	*clear* ( ) virtual
void	*edit* (*Object* object) virtual
bool	*forward_input_event* (*InputEvent* event) virtual
bool	*forward_spatial_input_event* (*Camera* camera, *InputEvent* event) virtual
StringArray	*get_breakpoints* ( ) virtual
String	*get_name* ( ) virtual
Dictionary	*get_state* ( ) virtual
bool	*handles* (*Object* object) virtual
bool	*has_main_screen* ( ) virtual
void	*make_visible* (*bool* visible) virtual
void	*set_state* (*Dictionary* state) virtual
Object	*get_undo_redo* ( )
void	*add_custom_control* (*int* container, *Object* control)
void	*add_custom_type* (*String* type, *String* base, *Script* script, *Texture* icon)
void	*remove_custom_type* (*String* type)

9.78.3 Numeric Constants

- **CONTAINER_TOOLBAR = 0**
- **CONTAINER_SPATIAL_EDITOR_MENU = 1**
- **CONTAINER_SPATIAL_EDITOR_SIDE = 2**
- **CONTAINER_SPATIAL_EDITOR_BOTTOM = 3**
- **CONTAINER_CANVAS_EDITOR_MENU = 4**
- **CONTAINER_CANVAS_EDITOR_SIDE = 5**

9.78.4 Member Function Description

- void **apply_changes** ( ) virtual
- void **clear** ( ) virtual
- void **edit** (*Object* object) virtual
- *bool* **forward_input_event** (*InputEvent* event) virtual
- *bool* **forward_spatial_input_event** (*Camera* camera, *InputEvent* event) virtual
- *StringArray* **get_breakpoints** ( ) virtual
- *String* **get_name** ( ) virtual
- *Dictionary* **get_state** ( ) virtual
- *bool* **handles** (*Object* object) virtual
- *bool* **has_main_screen** ( ) virtual
- void **make_visible** (*bool* visible) virtual
- void **set_state** (*Dictionary* state) virtual
- *Object* **get_undo_redo** ( )
- void **add_custom_control** (*int* container, *Object* control)
- void **add_custom_type** (*String* type, *String* base, *Script* script, *Texture* icon)
- void **remove_custom_type** (*String* type)

9.79 EditorScenePostImport

Inherits: *Reference* < *Object*

Category: Core

9.79.1 Brief Description

9.79.2 Member Functions

void	*post_import* (*Object* scene) virtual

9.79.3 Member Function Description

- void **post_import** (*Object* scene) virtual

9.80 EditorScript

Inherits: *Reference* < *Object*

Category: Core

9.80.1 Brief Description

9.80.2 Member Functions

void	*_run* ( ) virtual
void	*add_root_node* (*Object* node)
Object	*get_scene* ( )

9.80.3 Member Function Description

- void **_run** ( ) virtual
- void **add_root_node** (*Object* node)
- *Object* **get_scene** ( )

9.81 Environment

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.81.1 Brief Description

9.81.2 Member Functions

void	*set_background* (*int* bgmode)
int	*get_background* ( ) const
void	*set_background_param* (*int* param, var value)
void	*get_background_param* (*int* param) const
void	*set_enable_fx* (*int* effect, *bool* enabled)
bool	*is_fx_enabled* (*int* effect) const
void	*fx_set_param* (*int* param, var value)
void	*fx_get_param* (*int* param) const

9.81.3 Numeric Constants

- **BG_KEEP = 0**
- **BG_DEFAULT_COLOR = 1**
- **BG_COLOR = 2**
- **BG_TEXTURE = 3**
- **BG_CUBEMAP = 4**
- **BG_CANVAS = 5**
- **BG_MAX = 6**
- **BG_PARAM_CANVAS_MAX_LAYER = 0**
- **BG_PARAM_COLOR = 1**
- **BG_PARAM_TEXTURE = 2**
- **BG_PARAM_CUBEMAP = 3**
- **BG_PARAM_ENERGY = 4**
- **BG_PARAM_GLOW = 6**
- **BG_PARAM_MAX = 7**
- **FX_AMBIENT_LIGHT = 0**
- **FX_FXAA = 1**
- **FX_GLOW = 2**
- **FX_DOF_BLUR = 3**
- **FX_HDR = 4**
- **FX_FOG = 5**
- **FX_BCS = 6**
- **FX_SRGB = 7**
- **FX_MAX = 8**
- **FX_BLUR_BLEND_MODE_ADDITIVE = 0**
- **FX_BLUR_BLEND_MODE_SCREEN = 1**
- **FX_BLUR_BLEND_MODE_SOFTLIGHT = 2**
- **FX_HDR_TONE_MAPPER_LINEAR = 0**
- **FX_HDR_TONE_MAPPER_LOG = 1**
- **FX_HDR_TONE_MAPPER_REINHARDT = 2**
- **FX_HDR_TONE_MAPPER_REINHARDT_AUTOWHITE = 3**
- **FX_PARAM_AMBIENT_LIGHT_COLOR = 0**
- **FX_PARAM_AMBIENT_LIGHT_ENERGY = 1**
- **FX_PARAM_GLOW_BLUR_PASSES = 2**
- **FX_PARAM_GLOW_BLUR_SCALE = 3**
- **FX_PARAM_GLOW_BLUR_STRENGTH = 4**

- **FX_PARAM_GLOW_BLUR_BLEND_MODE = 5**
- **FX_PARAM_GLOW_BLOOM = 6**
- **FX_PARAM_GLOW_BLOOM_TRESHOLD = 7**
- **FX_PARAM_DOF_BLUR_PASSES = 8**
- **FX_PARAM_DOF_BLUR_BEGIN = 9**
- **FX_PARAM_DOF_BLUR_RANGE = 10**
- **FX_PARAM_HDR_TONEMAPPER = 11**
- **FX_PARAM_HDR_EXPOSURE = 12**
- **FX_PARAM_HDR_WHITE = 13**
- **FX_PARAM_HDR_GLOW_TRESHOLD = 14**
- **FX_PARAM_HDR_GLOW_SCALE = 15**
- **FX_PARAM_HDR_MIN_LUMINANCE = 16**
- **FX_PARAM_HDR_MAX_LUMINANCE = 17**
- **FX_PARAM_HDR_EXPOSURE_ADJUST_SPEED = 18**
- **FX_PARAM_FOG_BEGIN = 19**
- **FX_PARAM_FOG_ATTENUATION = 22**
- **FX_PARAM_FOG_BEGIN_COLOR = 20**
- **FX_PARAM_FOG_END_COLOR = 21**
- **FX_PARAM_FOG_BG = 23**
- **FX_PARAM_BCS_BRIGHTNESS = 24**
- **FX_PARAM_BCS_CONTRAST = 25**
- **FX_PARAM_BCS_SATURATION = 26**
- **FX_PARAM_MAX = 27**

9.81.4 Member Function Description

- void **set_background** (*int* bgmode)
- *int* **get_background** ( ) const
- void **set_background_param** (*int* param, var value)
- void **get_background_param** (*int* param) const
- void **set_enable_fx** (*int* effect, *bool* enabled)
- *bool* **is_fx_enabled** (*int* effect) const
- void **fx_set_param** (*int* param, var value)
- void **fx_get_param** (*int* param) const

9.82 EventPlayer

Inherits: *Node* < *Object*

Category: Core

9.82.1 Brief Description

Class for event stream playback.

9.82.2 Member Functions

void	*set_stream* (*EventStream* stream)
EventStream	*get_stream* ( ) const
void	*play* ( )
void	*stop* ( )
bool	*is_playing* ( ) const
void	*set_paused* (*bool* paused)
bool	*is_paused* ( ) const
void	*set_loop* (*bool* enabled)
bool	*has_loop* ( ) const
void	*set_volume* (*float* volume)
float	*get_volume* ( ) const
void	*set_pitch_scale* (*float* pitch_scale)
float	*get_pitch_scale* ( ) const
void	*set_tempo_scale* (*float* tempo_scale)
float	*get_tempo_scale* ( ) const
void	*set_volume_db* (*float* db)
float	*get_volume_db* ( ) const
String	*get_stream_name* ( ) const
int	*get_loop_count* ( ) const
float	*get_pos* ( ) const
void	*seek_pos* (*float* time)
float	*get_length* ( ) const
void	*set_autoplay* (*bool* enabled)
bool	*has_autoplay* ( ) const
void	*set_channel_volume* (*int* channel, *float* channel_volume)
float	*get_channel_volume* (*int* channel) const
float	*get_channel_last_note_time* (*int* channel) const

9.82.3 Description

Class for event stream playback. Event streams are music expressed as a series of events (note on, note off, instrument change...), as opposed to audio streams, which are just audio data. Examples of event-based streams are MIDI files, or MOD music.

Currently, only MOD, S3M, IT, and XM music is supported.

9.82.4 Member Function Description

- void **set_stream** (*EventStream* stream)

Set the *EventStream* this player will play.

- *EventStream* **get_stream** ( ) const

Return the currently assigned stream.

- void **play** ( )

Play the currently assigned stream.

- void **stop** ( )

Stop playing.

- *bool* **is_playing** ( ) const

Return whether this player is playing.

- void **set_paused** (*bool* paused)

Pause stream playback.

- *bool* **is_paused** ( ) const

Return whether the playback is currently paused.

- void **set_loop** (*bool* enabled)

Set whether the stream will be restarted at the end.

- *bool* **has_loop** ( ) const

Return whether this player will be restart the playback at the end.

- void **set_volume** (*float* volume)

Set the playback volume for this player. This is a float between 0.0 (silent) and 1.0 (full volume). Values over 1.0 may amplify sound even more, but may introduce distortion. Negative values may just invert the output waveform, which produces no audible difference.

The effect of these special values ultimately depends on the low-level implementation of the file format being played.

- *float* **get_volume** ( ) const

Return the playback volume for this player.

- void **set_pitch_scale** (*float* pitch_scale)

Set the pitch multiplier for all sounds coming from this stream. A value of 2.0 shifts all pitches one octave up, and a value of 0.5 shifts pitches one octave down.

- *float* **get_pitch_scale** ( ) const

Return the pitch scale factor for this player.

- void **set_tempo_scale** (*float* tempo_scale)

Set the tempo multiplier. This allows to slow down or speed up the music, without affecting its pitch.

- *float* **get_tempo_scale** ( ) const

Return the tempo multiplier.

- void **set_volume_db** (*float* db)

Set the playback volume for this player, in decibels. This is a float between -80.0 (silent) and 0.0 (full volume). Values under -79.0 get truncated to -80, but values over 0.0 do not, so the warnings for over amplifying (see *set_volume*) still apply.

- *float* **get_volume_db** **(** **)** const

Return the playback volume for this player, in decibels.

- *String* **get_stream_name** **(** **)** const

Return the name of the currently assigned stream. This is not the file name, but a field inside the file. If no stream is assigned, if returns "<No Stream>".

- *int* **get_loop_count** **(** **)** const

Return the number of times the playback has looped.

- *float* **get_pos** **(** **)** const

Return the playback position. May be in seconds, but depends on the stream type.

- void **seek_pos** **(** *float* time **)**

Set the playback position. May be in seconds, but depends on the stream type.

- *float* **get_length** **(** **)** const

Return the song length. May be in seconds, but depends on the stream type.

- void **set_autoplay** **(** *bool* enabled **)**

Set whether this player will start playing as soon as it enters the scene tree.

- *bool* **has_autoplay** **(** **)** const

Return whether this player will start playing as soon as it enters the scene tree.

- void **set_channel_volume** **(** *int* channel, *float* channel_volume **)**

Set the volume scale for an individual channel of the stream, with the same value range as *set_volume*. The channel number depends on the stream format. For example, MIDIs range from 0 to 15, and MODs from 0 to 63.

Many stream formats are multichannel, so this allows to affect only a part of the music.

- *float* **get_channel_volume** **(** *int* channel **)** const

Return the volume scale for an individual channel of the stream.

- *float* **get_channel_last_note_time** **(** *int* channel **)** const

Return the time at which the last note of a given channel in the stream plays.

9.83 EventStream

Inherits: *Resource* **<** *Reference* **<** *Object*

Inherited By: *EventStreamChibi*

Category: Core

9.83.1 Brief Description

Base class for all event-based stream drivers.

9.83.2 Description

Base class for all event-based stream drivers. Event streams are music expressed as a series of events (note on, note off, instrument change...), as opposed to audio streams, which are just audio data. Examples of event-based streams are MIDI files, of MOD music.

This class exposes no methods.

9.84 EventStreamChibi

Inherits: *EventStream* < *Resource* < *Reference* < *Object*

Category: Core

9.84.1 Brief Description

Driver for MOD playback.

9.84.2 Description

This driver plays MOD music. MOD music, as all event-based streams, is a music format defined by note events occurring at defined moments, instead of a stream of audio samples.

Currently, this driver supports the MOD, S3M, IT, and XM formats.

This class exposes no methods.

This class can return its playback position in seconds, but does not allow to set it, failing with only a console warning.

This class can not return its song length, returning 1.0 when queried.

This class does not limit its volume settings, allowing for overflow/distortion and wave inversion.

9.85 File

Inherits: *Reference* < *Object*

Category: Core

9.85.1 Brief Description

9.85.2 Member Functions

int	*open_encrypted* (*String* path, *int* mode_flags, *RawArray* key)
int	*open_encrypted_with_pass* (*String* path, *int* mode_flags, *String* pass)
int	*open* (*String* path, *int* flags)
void	*close* ( )
bool	*is_open* ( ) const
void	*seek* (*int* pos)
void	*seek_end* (*int* pos=0)

Continúa en la página siguiente

Tabla 9.10 – proviene de la página anterior

int	*get_pos* ( ) const
int	*get_len* ( ) const
bool	*eof_reached* ( ) const
int	*get_8* ( ) const
int	*get_16* ( ) const
int	*get_32* ( ) const
int	*get_64* ( ) const
float	*get_float* ( ) const
float	*get_double* ( ) const
float	*get_real* ( ) const
RawArray	*get_buffer* (*int* len) const
String	*get_line* ( ) const
String	*get_as_text* ( ) const
bool	*get_endian_swap* ( )
void	*set_endian_swap* (*bool* enable)
Error	*get_error* ( ) const
void	*get_var* ( ) const
StringArray	*get_csv_line* (*String* delim=”,”) const
void	*store_8* (*int* value)
void	*store_16* (*int* value)
void	*store_32* (*int* value)
void	*store_64* (*int* value)
void	*store_float* (*float* value)
void	*store_double* (*float* value)
void	*store_real* (*float* value)
void	*store_buffer* (*RawArray* buffer)
void	*store_line* (*String* line)
void	*store_string* (*String* string)
void	*store_var* (var value)
void	*store_pascal_string* (*String* string)
String	*get_pascal_string* ( )
bool	*file_exists* (*String* path) const

9.85.3 Numeric Constants

- **READ = 1**
- **WRITE = 2**
- **READ_WRITE = 3**
- **WRITE_READ = 7**

9.85.4 Member Function Description

- *int* **open_encrypted** (*String* path, *int* mode_flags, *RawArray* key)
- *int* **open_encrypted_with_pass** (*String* path, *int* mode_flags, *String* pass)
- *int* **open** (*String* path, *int* flags)
- void **close** ( )
- *bool* **is_open** ( ) const

- void **seek** (*int* pos)
- void **seek_end** (*int* pos=0)
- *int* **get_pos** ( ) const
- *int* **get_len** ( ) const
- *bool* **eof_reached** ( ) const
- *int* **get_8** ( ) const
- *int* **get_16** ( ) const
- *int* **get_32** ( ) const
- *int* **get_64** ( ) const
- *float* **get_float** ( ) const
- *float* **get_double** ( ) const
- *float* **get_real** ( ) const
- *RawArray* **get_buffer** (*int* len) const
- *String* **get_line** ( ) const
- *String* **get_as_text** ( ) const
- *bool* **get_endian_swap** ( )
- void **set_endian_swap** (*bool* enable)
- Error **get_error** ( ) const
- void **get_var** ( ) const
- *StringArray* **get_csv_line** (*String* delim=”,”) const
- void **store_8** (*int* value)
- void **store_16** (*int* value)
- void **store_32** (*int* value)
- void **store_64** (*int* value)
- void **store_float** (*float* value)
- void **store_double** (*float* value)
- void **store_real** (*float* value)
- void **store_buffer** (*RawArray* buffer)
- void **store_line** (*String* line)
- void **store_string** (*String* string)
- void **store_var** (var value)
- void **store_pascal_string** (*String* string)
- *String* **get_pascal_string** ( )
- *bool* **file_exists** (*String* path) const

9.86 FileDialog

Inherits: *ConfirmationDialog* < *AcceptDialog* < *WindowDialog* < *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.86.1 Brief Description

Dialog for selecting files or directories in the filesystem.

9.86.2 Member Functions

void	*clear_filters* ( )
void	*add_filter* (*String* filter)
String	*get_current_dir* ( ) const
String	*get_current_file* ( ) const
String	*get_current_path* ( ) const
void	*set_current_dir* (*String* dir)
void	*set_current_file* (*String* file)
void	*set_current_path* (*String* path)
void	*set_mode* (*int* mode)
int	*get_mode* ( ) const
VBoxContainer	*get_vbox* ( )
void	*set_access* (*int* access)
int	*get_access* ( ) const
void	*set_show_hidden_files* (*bool* show)
bool	*is_showing_hidden_files* ( ) const
void	*invalidate* ( )

9.86.3 Signals

- **files_selected** (*StringArray* paths)
- **dir_selected** (*String* dir)
- **file_selected** (*String* path)

9.86.4 Numeric Constants

- **MODE_OPEN_FILE = 0** — The dialog allows the selection of one, and only one file.
- **MODE_OPEN_FILES = 1** — The dialog allows the selection of multiple files.
- **MODE_OPEN_DIR = 2** — The dialog functions as a folder selector, disallowing the selection of any file.
- **MODE_SAVE_FILE = 3** — The dialog will warn when a file exists.
- **ACCESS_RESOURCES = 0** — The dialog allows the selection of file and directory.
- **ACCESS_USERDATA = 1** — The dialog allows ascess files under *Resource* path(res://) .
- **ACCESS_FILESYSTEM = 2** — The dialog allows ascess files in whole file system.

9.86.5 Description

FileDialog is a preset dialog used to choose files and directories in the filesystem. It supports filter masks.

9.86.6 Member Function Description

- void **clear_filters** ( )

Clear all the added filters in the dialog.

- void **add_filter** (*String* filter)

Add a custom filter. Filter format is: "mask ; description", example (C++): dialog->add_filter("*.png ; PNG Images");

- *String* **get_current_dir** ( ) const

Get the current working directory of the file dialog.

- *String* **get_current_file** ( ) const

Get the current selected file of the file dialog (empty if none).

- *String* **get_current_path** ( ) const

Get the current selected path (directory and file) of the file dialog (empty if none).

- void **set_current_dir** (*String* dir)

Set the current working directory of the file dialog.

- void **set_current_file** (*String* file)

Set the current selected file name of the file dialog.

- void **set_current_path** (*String* path)

Set the current selected file path of the file dialog.

- void **set_mode** (*int* mode)

Set the file dialog mode from the MODE_* enum.

- *int* **get_mode** ( ) const

Get the file dialog mode from the MODE_* enum.

- *VBoxContainer* **get_vbox** ( )

Return the vertical box container of the dialog, custom controls can be added to it.

- void **set_access** (*int* access)

Set the file access permission of the dialog(Must be one of ACCESS_RESOURCES, ACCESS_USERDATA or ACCESS_FILESYSTEM).

- *int* **get_access** ( ) const

Return the file access permission of the dialog.

- void **set_show_hidden_files** (*bool* show)

Set the dialog should show hidden files.

- *bool* **is_showing_hidden_files** ( ) const

Return true if the diaog allows show hidden files.

- void **invalidate** ( )

Invalidate and update the current dialog content list.

9.87 FixedMaterial

Inherits: *Material* < *Resource* < *Reference* < *Object*

Category: Core

9.87.1 Brief Description

Simple Material with a fixed parameter set.

9.87.2 Member Functions

void	*set_parameter* (*int* param, var value)
void	*get_parameter* (*int* param) const
void	*set_texture* (*int* param, *Texture* texture)
Texture	*get_texture* (*int* param) const
void	*set_texcoord_mode* (*int* param, *int* mode)
int	*get_texcoord_mode* (*int* param) const
void	*set_fixed_flag* (*int* flag, *bool* value)
bool	*get_fixed_flag* (*int* flag) const
void	*set_uv_transform* (*Transform* transform)
Transform	*get_uv_transform* ( ) const
void	*set_light_shader* (*int* shader)
int	*get_light_shader* ( ) const
void	*set_point_size* (*float* size)
float	*get_point_size* ( ) const

9.87.3 Numeric Constants

- **PARAM_DIFFUSE = 0** — Diffuse Lighting (light scattered from surface).
- **PARAM_DETAIL = 1** — Detail Layer for diffuse lighting.
- **PARAM_SPECULAR = 2** — Specular Lighting (light reflected from the surface).
- **PARAM_EMISSION = 3** — Emission Lighting (light emitted from the surface).
- **PARAM_SPECULAR_EXP = 4** — Specular Exponent (size of the specular dot).
- **PARAM_GLOW = 5** — Glow (Visible emitted scattered light).
- **PARAM_NORMAL = 6** — Normal Map (irregularity map).
- **PARAM_SHADE_PARAM = 7**
- **PARAM_MAX = 8** — Maximum amount of parameters.
- **TEXCOORD_SPHERE = 3**
- **TEXCOORD_UV = 0** — Read texture coordinates from the UV array.

- **TEXCOORD_UV_TRANSFORM = 1** — Read texture coordinates from the UV array and transform them by uv_xform.
- **TEXCOORD_UV2 = 2** — Read texture coordinates from the UV2 array.
- **FLAG_USE_ALPHA = 0**
- **FLAG_USE_COLOR_ARRAY = 1**
- **FLAG_USE_POINT_SIZE = 2**
- **FLAG_DISCARD_ALPHA = 3**
- **LIGHT_SHADER_LAMBERT = 0**
- **LIGHT_SHADER_WRAP = 1**
- **LIGHT_SHADER_VELVET = 2**
- **LIGHT_SHADER_TOON = 3**

9.87.4 Description

FixedMaterial is a simple type of material *Resource*, which contains a fixed amount of parameters. It is the only type of material supported in fixed-pipeline devices and APIs. It is also an often a better alternative to *ShaderMaterial* for most simple use cases.

9.87.5 Member Function Description

- void **set_parameter** (*int* param, var value)

Set a parameter, parameters are defined in the PARAM_* enum. The type of each parameter may change, so it's best to check the enum.

- void **get_parameter** (*int* param) const

Return a parameter, parameters are defined in the PARAM_* enum. The type of each parameter may change, so it's best to check the enum.

- void **set_texture** (*int* param, *Texture* texture)

Set a texture. Textures change parameters per texel and are mapped to the model depending on the texcoord mode (see *set_texcoord_mode*).

- *Texture* **get_texture** (*int* param) const

Return a texture. Textures change parameters per texel and are mapped to the model depending on the texcoord mode (see *set_texcoord_mode*).

- void **set_texcoord_mode** (*int* param, *int* mode)

Set the texture coordinate mode. Each texture param (from the PARAM_* enum) has one. It defines how the textures are mapped to the object.

- *int* **get_texcoord_mode** (*int* param) const

Return the texture coordinate mode. Each texture param (from the PARAM_* enum) has one. It defines how the textures are mapped to the object.

- void **set_fixed_flag** (*int* flag, *bool* value)
- *bool* **get_fixed_flag** (*int* flag) const
- void **set_uv_transform** (*Transform* transform)

Sets a special transform used to post-transform UV coordinates of the uv_xform texcoord mode: TEXCOORD_UV_TRANSFORM.

- *Transform* **get_uv_transform** ( ) const

Returns the special transform used to post-transform UV coordinates of the uv_xform texcoord mode: TEXCOORD_UV_TRANSFORM.

- void **set_light_shader** (*int* shader)
- *int* **get_light_shader** ( ) const
- void **set_point_size** (*float* size)
- *float* **get_point_size** ( ) const

9.88 float

Category: Built-In Types

9.88.1 Brief Description

Float built-in type

9.88.2 Member Functions

float	float (*bool* from)
float	float (*int* from)
float	float (*String* from)

9.88.3 Description

Float built-in type.

9.88.4 Member Function Description

- *float* **float** (*bool* from)

Cast a *bool* value to a floating point value, `float(true)` will be equals to 1.0 and `float(false)` will be equals to 0.0.

- *float* **float** (*int* from)

Cast an *int* value to a floating point value, `float(1)` will be equals to 1.0.

- *float* **float** (*String* from)

Cast a *String* value to a floating point value. This method accepts float value strings like ‘‘ ‘1.23’ ‘‘ and exponential notation strings for its parameter so calling ‘‘ float(‘1e3’) ‘‘ will return 1000.0 and calling ‘‘ float(‘1e-3’) ‘‘ will return -0.001.

9.89 Font

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.89.1 Brief Description

Internationalized font and text drawing support.

9.89.2 Member Functions

int	*create_from_fnt* (*String* path)
void	*set_height* (*float* px)
float	*get_height* ( ) const
void	*set_ascent* (*float* px)
float	*get_ascent* ( ) const
float	*get_descent* ( ) const
void	*add_kerning_pair* (*int* char_a, *int* char_b, *int* kerning)
int	*get_kerning_pair* (*int* char_a, *int* char_b) const
void	*add_texture* (*Texture* texture)
void	*add_char* (*int* character, *int* texture, *Rect2* rect, *Vector2* align=Vector2(0,0), *float* advance=-1)
int	*get_texture_count* ( ) const
Texture	*get_texture* (*int* idx) const
Vector2	*get_char_size* (*int* char, *int* next=0) const
Vector2	*get_string_size* (*String* string) const
void	*set_distance_field_hint* (*bool* enable)
bool	*is_distance_field_hint* ( ) const
void	*clear* ( )
void	*draw* (*RID* canvas_item, *Vector2* pos, *String* string, *Color* modulate=Color(1,1,1,1), *int* clip_w=-1) const
float	*draw_char* (*RID* canvas_item, *Vector2* pos, *int* char, *int* next=-1, *Color* modulate=Color(1,1,1,1)) const
void	*set_fallback* (*Object* fallback)
Object	*get_fallback* ( ) const

9.89.3 Description

Font contains an unicode compatible character set, as well as the ability to draw it with variable width, ascent, descent and kerning. For creating fonts from TTF files (or other font formats), see the editor support for fonts. TODO check wikipedia for graph of ascent/baseline/descent/height/etc.

9.89.4 Member Function Description

- *int* **create_from_fnt** (*String* path)
- void **set_height** (*float* px)

Set the total font height (ascent plus descent) in pixels.

- *float* **get_height** ( ) const

Return the total font height (ascent plus descent) in pixels.

- void **set_ascent** (*float* px)

Set the font ascent (number of pixels above the baseline).

- *float* **get_ascent** ( ) const

Return the font ascent (number of pixels above the baseline).

- *float* **get_descent** ( ) const

Return the font descent (number of pixels below the baseline).

- void **add_kerning_pair** (*int* char_a, *int* char_b, *int* kerning)

Add a kerning pair to the *Font* as a difference. Kerning pairs are special cases where a typeface advance is determined by the next character.

- *int* **get_kerning_pair** (*int* char_a, *int* char_b) const

Return a kerning pair as a difference. Kerning pairs are special cases where a typeface advance is determined by the next character.

- void **add_texture** (*Texture* texture)

Add a texture to the *Font*.

- void **add_char** (*int* character, *int* texture, *Rect2* rect, *Vector2* align=Vector2(0,0), *float* advance=-1)

Add a character to the font, where "character" is the unicode value, "texture" is the texture index, "rect" is the region in the texture (in pixels!), "align" is the (optional) alignment for the character and "advance" is the (optional) advance.

- *int* **get_texture_count** ( ) const
- *Texture* **get_texture** (*int* idx) const
- *Vector2* **get_char_size** (*int* char, *int* next=0) const

Return the size of a character, optionally taking kerning into account if the next character is provided.

- *Vector2* **get_string_size** (*String* string) const

Return the size of a string, taking kerning and advance into account.

- void **set_distance_field_hint** (*bool* enable)
- *bool* **is_distance_field_hint** ( ) const
- void **clear** ( )

Clear all the font data.

- void **draw** (*RID* canvas_item, *Vector2* pos, *String* string, *Color* modulate=Color(1,1,1,1), *int* clip_w=-1) const

Draw "string" into a canvas item using the font at a given "pos" position, with "modulate" color, and optionally clipping the width. "pos" specifies the baseline, not the top. To draw from the top, *ascent* must be added to the Y axis.

- *float* **draw_char** (*RID* canvas_item, *Vector2* pos, *int* char, *int* next=-1, *Color* modulate=Color(1,1,1,1)) const

Draw character "char" into a canvas item using the font at a given "pos" position, with "modulate" color, and optionally kerning if "next" is passed. clipping the width. "pos" specifies the baseline, not the top. To draw from the top, *ascent* must be added to the Y axis. The width used by the character is returned, making this function useful for drawing strings character by character.

- void **set_fallback** (*Object* fallback)
- *Object* **get_fallback** ( ) const

9.90 FuncRef

Inherits: *Reference* < *Object*

Category: Core

9.90.1 Brief Description

9.90.2 Member Functions

void	*call_func* (var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL, var arg6=NULL, var arg7=NULL, var arg8=NULL, var arg9=NULL)
void	*set_instance* (*Object* instance)
void	*set_function* (*String* name)

9.90.3 Member Function Description

- void **call_func** (var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL, var arg6=NULL, var arg7=NULL, var arg8=NULL, var arg9=NULL)
- void **set_instance** (*Object* instance)
- void **set_function** (*String* name)

9.91 GDFunctionState

Inherits: *Reference* < *Object*

Category: Core

9.91.1 Brief Description

9.91.2 Member Functions

Variant	*resume* (var arg=NULL)
bool	*is_valid* ( ) const

9.91.3 Member Function Description

- Variant **resume** (var arg=NULL)
- *bool* **is_valid** ( ) const

Should put children to the top left corner instead of center of the container.

9.92 GDNativeClass

Inherits: *Reference* < *Object*

Category: Core

9.92.1 Brief Description

9.92.2 Member Functions

void	*new* ( )

9.92.3 Member Function Description

- void **new** ( )

9.93 GDScript

Inherits: *Script* < *Resource* < *Reference* < *Object*

Category: Core

9.93.1 Brief Description

9.93.2 Member Functions

void	*new* ( )
RawArray	*get_as_byte_code* ( ) const

9.93.3 Member Function Description

- void **new** ( )
- *RawArray* **get_as_byte_code** ( ) const

9.94 Generic6DOFJoint

Inherits: *Joint* < *Spatial* < *Node* < *Object*

Category: Core

9.94.1 Brief Description

9.94.2 Member Functions

void	*set_param_x* (*int* param, *float* value)
float	*get_param_x* (*int* param) const
void	*set_param_y* (*int* param, *float* value)
float	*get_param_y* (*int* param) const
void	*set_param_z* (*int* param, *float* value)
float	*get_param_z* (*int* param) const
void	*set_flag_x* (*int* flag, *bool* value)
bool	*get_flag_x* (*int* flag) const
void	*set_flag_y* (*int* flag, *bool* value)
bool	*get_flag_y* (*int* flag) const
void	*set_flag_z* (*int* flag, *bool* value)
bool	*get_flag_z* (*int* flag) const

9.94.3 Numeric Constants

- **PARAM_LINEAR_LOWER_LIMIT = 0**
- **PARAM_LINEAR_UPPER_LIMIT = 1**
- **PARAM_LINEAR_LIMIT_SOFTNESS = 2**
- **PARAM_LINEAR_RESTITUTION = 3**
- **PARAM_LINEAR_DAMPING = 4**
- **PARAM_ANGULAR_LOWER_LIMIT = 5**
- **PARAM_ANGULAR_UPPER_LIMIT = 6**
- **PARAM_ANGULAR_LIMIT_SOFTNESS = 7**
- **PARAM_ANGULAR_DAMPING = 8**
- **PARAM_ANGULAR_RESTITUTION = 9**
- **PARAM_ANGULAR_FORCE_LIMIT = 10**
- **PARAM_ANGULAR_ERP = 11**
- **PARAM_ANGULAR_MOTOR_TARGET_VELOCITY = 12**
- **PARAM_ANGULAR_MOTOR_FORCE_LIMIT = 13**
- **PARAM_MAX = 14**
- **FLAG_ENABLE_LINEAR_LIMIT = 0**
- **FLAG_ENABLE_ANGULAR_LIMIT = 1**
- **FLAG_ENABLE_MOTOR = 2**
- **FLAG_MAX = 3**

9.94.4 Member Function Description

- void **set_param_x** (*int* param, *float* value)
- *float* **get_param_x** (*int* param) const
- void **set_param_y** (*int* param, *float* value)
- *float* **get_param_y** (*int* param) const
- void **set_param_z** (*int* param, *float* value)
- *float* **get_param_z** (*int* param) const
- void **set_flag_x** (*int* flag, *bool* value)
- *bool* **get_flag_x** (*int* flag) const
- void **set_flag_y** (*int* flag, *bool* value)
- *bool* **get_flag_y** (*int* flag) const
- void **set_flag_z** (*int* flag, *bool* value)
- *bool* **get_flag_z** (*int* flag) const

9.95 Geometry

Inherits: *Object*

Category: Core

9.95.1 Brief Description

9.95.2 Member Functions

Array	*build_box_planes* (*Vector3* extents)
Array	*build_cylinder_planes* (*float* radius, *float* height, *int* sides, *int* axis=2)
Array	*build_capsule_planes* (*float* radius, *float* height, *int* sides, *int* lats, *int* axis=2)
float	*segment_intersects_circle* (*Vector2* segment_from, *Vector2* segment_to, *Vector2* circle_pos, *float* circle_radius)
void	*segment_intersects_segment_2d* (*Vector2* from_a, *Vector2* to_a, *Vector2* from_b, *Vector2* to_b)
Vector2Array	*get_closest_points_between_segments_2d* (*Vector2* p1, *Vector2* q1, *Vector2* p2, *Vector2* q2)
Vector3Array	*get_closest_points_between_segments* (*Vector3* p1, *Vector3* p2, *Vector3* q1, *Vector3* q2)
Vector3	*get_closest_point_to_segment* (*Vector3* point, *Vector3* s1, *Vector3* s2)
int	*get_uv84_normal_bit* (*Vector3* normal)
void	*ray_intersects_triangle* (*Vector3* from, *Vector3* dir, *Vector3* a, *Vector3* b, *Vector3* c)
void	*segment_intersects_triangle* (*Vector3* from, *Vector3* to, *Vector3* a, *Vector3* b, *Vector3* c)
Vector3Array	*segment_intersects_sphere* (*Vector3* from, *Vector3* to, *Vector3* spos, *float* sradius)
Vector3Array	*segment_intersects_cylinder* (*Vector3* from, *Vector3* to, *float* height, *float* radius)
Vector3Array	*segment_intersects_convex* (*Vector3* from, *Vector3* to, *Array* planes)
bool	*point_is_inside_triangle* (*Vector2* point, *Vector2* a, *Vector2* b, *Vector2* c) const
IntArray	*triangulate_polygon* (*Vector2Array* polygon)
Dictionary	*make_atlas* (*Vector2Array* sizes)

9.95.3 Member Function Description

- *Array* **build_box_planes** (*Vector3* extents)
- *Array* **build_cylinder_planes** (*float* radius, *float* height, *int* sides, *int* axis=2)
- *Array* **build_capsule_planes** (*float* radius, *float* height, *int* sides, *int* lats, *int* axis=2)
- *float* **segment_intersects_circle** (*Vector2* segment_from, *Vector2* segment_to, *Vector2* circle_pos, *float* circle_radius)
- void **segment_intersects_segment_2d** (*Vector2* from_a, *Vector2* to_a, *Vector2* from_b, *Vector2* to_b)
- *Vector2Array* **get_closest_points_between_segments_2d** (*Vector2* p1, *Vector2* q1, *Vector2* p2, *Vector2* q2)
- *Vector3Array* **get_closest_points_between_segments** (*Vector3* p1, *Vector3* p2, *Vector3* q1, *Vector3* q2)
- *Vector3* **get_closest_point_to_segment** (*Vector3* point, *Vector3* s1, *Vector3* s2)
- *int* **get_uv84_normal_bit** (*Vector3* normal)
- void **ray_intersects_triangle** (*Vector3* from, *Vector3* dir, *Vector3* a, *Vector3* b, *Vector3* c)
- void **segment_intersects_triangle** (*Vector3* from, *Vector3* to, *Vector3* a, *Vector3* b, *Vector3* c)
- *Vector3Array* **segment_intersects_sphere** (*Vector3* from, *Vector3* to, *Vector3* spos, *float* sradius)
- *Vector3Array* **segment_intersects_cylinder** (*Vector3* from, *Vector3* to, *float* height, *float* radius)
- *Vector3Array* **segment_intersects_convex** (*Vector3* from, *Vector3* to, *Array* planes)

- *bool* **point_is_inside_triangle** (*Vector2* point, *Vector2* a, *Vector2* b, *Vector2* c) const
- *IntArray* **triangulate_polygon** (*Vector2Array* polygon)
- *Dictionary* **make_atlas** (*Vector2Array* sizes)

9.96 GeometryInstance

Inherits: *VisualInstance* < *Spatial* < *Node* < *Object*

Inherited By: *MultiMeshInstance*, *SpriteBase3D*, *MeshInstance*, *Particles*, *Quad*, *TestCube*, *ImmediateGeometry*

Category: Core

9.96.1 Brief Description

Base node for geometry based visual instances.

9.96.2 Member Functions

void	*set_material_override* (*Object* material)
Object	*get_material_override* ( ) const
void	*set_flag* (*int* flag, *bool* value)
bool	*get_flag* (*int* flag) const
void	*set_draw_range_begin* (*float* mode)
float	*get_draw_range_begin* ( ) const
void	*set_draw_range_end* (*float* mode)
float	*get_draw_range_end* ( ) const
void	*set_baked_light_texture_id* (*int* id)
int	*get_baked_light_texture_id* ( ) const
void	*set_extra_cull_margin* (*float* margin)
float	*get_extra_cull_margin* ( ) const

9.96.3 Numeric Constants

- **FLAG_VISIBLE = 0**
- **FLAG_CAST_SHADOW = 3**
- **FLAG_RECEIVE_SHADOWS = 4**
- **FLAG_BILLBOARD = 1**
- **FLAG_BILLBOARD_FIX_Y = 2**
- **FLAG_DEPH_SCALE = 5**
- **FLAG_VISIBLE_IN_ALL_ROOMS = 6**
- **FLAG_MAX = 8**

9.96.4 Description

Base node for geometry based visual instances. Shares some common functionality like visibility and custom materials.

9.96.5 Member Function Description

- void **set_material_override** (*Object* material)

Set the material override for the whole geometry.

- *Object* **get_material_override** ( ) const

Return the material override for the whole geometry.

- void **set_flag** (*int* flag, *bool* value)
- *bool* **get_flag** (*int* flag) const
- void **set_draw_range_begin** (*float* mode)
- *float* **get_draw_range_begin** ( ) const
- void **set_draw_range_end** (*float* mode)
- *float* **get_draw_range_end** ( ) const
- void **set_baked_light_texture_id** (*int* id)
- *int* **get_baked_light_texture_id** ( ) const
- void **set_extra_cull_margin** (*float* margin)
- *float* **get_extra_cull_margin** ( ) const

9.97 Globals

Inherits: *Object*

Category: Core

9.97.1 Brief Description

Contains global variables accessible from everywhere.

9.97.2 Member Functions

bool	*has* (*String* name) const
void	*set_order* (*String* name, *int* pos)
int	*get_order* (*String* name) const
void	*set_persisting* (*String* name, *bool* enable)
bool	*is_persisting* (*String* name) const
void	*clear* (*String* name)
String	*localize_path* (*String* path) const
String	*globalize_path* (*String* path) const
int	*save* ( )
bool	*has_singleton* (*String* name) const
Object	*get_singleton* (*String* name) const
bool	*load_resource_pack* (*String* pack)
int	*save_custom* (*String* file)

9.97.3 Description

Contains global variables accessible from everywhere. Use the normal *Object* API, such as "Globals.get(variable)", "Globals.set(variable,value)" or "Globals.has(variable)" to access them. Variables stored in engine.cfg are also loaded into globals, making this object very useful for reading custom game configuration options.

9.97.4 Member Function Description

- *bool* **has** (*String* name) const

Return true if a configuration value is present.

- void **set_order** (*String* name, *int* pos)

Set the order of a configuration value (influences when saved to the config file).

- *int* **get_order** (*String* name) const

Return the order of a configuration value (influences when saved to the config file).

- void **set_persisting** (*String* name, *bool* enable)

If set to true, this value can be saved to the configuration file. This is useful for editors.

- *bool* **is_persisting** (*String* name) const

If returns true, this value can be saved to the configuration file. This is useful for editors.

- void **clear** (*String* name)

Clear the whole configuration (not recommended, may break things).

- *String* **localize_path** (*String* path) const

Convert a path to a localized path (res:// path).

- *String* **globalize_path** (*String* path) const

Convert a localized path (res://) to a full native OS path.

- *int* **save** ( )
- *bool* **has_singleton** (*String* name) const
- *Object* **get_singleton** (*String* name) const
- *bool* **load_resource_pack** (*String* pack)
- *int* **save_custom** (*String* file)

9.98 GraphEdit

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.98.1 Brief Description

GraphEdit is an area capable of showing various GraphNodes. It manages connection events between them.

9.98.2 Member Functions

Error	*connect_node* (*String* from, *int* from_port, *String* to, *int* to_port)
bool	*is_node_connected* (*String* from, *int* from_port, *String* to, *int* to_port)
void	*disconnect_node* (*String* from, *int* from_port, *String* to, *int* to_port)
Array	*get_connection_list* ( ) const
Vector2	*get_scroll_ofs* ( ) const
void	*set_zoom* (*float* p_zoom)
float	*get_zoom* ( ) const
void	*set_right_disconnects* (*bool* enable)
bool	*is_right_disconnects_enabled* ( ) const

9.98.3 Signals

- **delete_nodes_request** ( )
- **duplicate_nodes_request** ( )
- **popup_request** (*Vector2* p_position)
- **_begin_node_move** ( )
- **disconnection_request** (*String* from, *int* from_slot, *String* to, *int* to_slot)
- **connection_request** (*String* from, *int* from_slot, *String* to, *int* to_slot)
- **_end_node_move** ( )

9.98.4 Description

GraphEdit manages the showing of GraphNodes it contains, as well as connections an disconnections between them. Signals are sent for each of these two events. Disconnection between GraphNodes slots is disabled by default.

It is greatly advised to enable low processor usage mode (see *OS.set_low_processor_usage_mode*) when using GraphEdits.

9.98.5 Member Function Description

- Error **connect_node** (*String* from, *int* from_port, *String* to, *int* to_port)

Create a connection between ‘from_port’ slot of ‘from’ GraphNode and ‘to_port’ slot of ‘to’ GraphNode. If the connection already exists, no connection is created.

- *bool* **is_node_connected** (*String* from, *int* from_port, *String* to, *int* to_port)

Return true if the ‘from_port’ slot of ‘from’ GraphNode is connected to the ‘to_port’ slot of ‘to’ GraphNode.

- void **disconnect_node** (*String* from, *int* from_port, *String* to, *int* to_port)

Remove the connection between ‘from_port’ slot of ‘from’ GraphNode and ‘to_port’ slot of ‘to’ GraphNode, if connection exists.

- *Array* **get_connection_list** ( ) const

Return an Array containing the list of connections. A connection consists in a structure of the form {from_slot: 0, from: “GraphNode name 0”, to_slot: 1, to: “GraphNode name 1” }

- *Vector2* **get_scroll_ofs** ( ) const

Return the scroll offset.

- void **set_zoom** (*float* p_zoom)

Set the zoom value of the GraphEdit. Zoom value is between 0.01; 1.728.

- *float* **get_zoom** ( ) const

Return the current zoom value.

- void **set_right_disconnects** (*bool* enable)

Enable the disconnection of existing connections in the visual GraphEdit by left-clicking a connection and releasing into the void.

- *bool* **is_right_disconnects_enabled** ( ) const

Return true is the disconnection of connections is enable in the visual GraphEdit. False otherwise.

9.99 GraphNode

Inherits: *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.99.1 Brief Description

A GraphNode is a container with several input and output slots allowing connections between GraphNodes. Slots can have different, incompatible types.

9.99.2 Member Functions

void	*set_title* (*String* title)
String	*get_title* ( ) const
void	*set_slot* (*int* idx, *bool* enable_left, *int* type_left, *Color* color_left, *bool* enable_right, *int* type_right, *Color* color_right)
void	*clear_slot* (*int* idx)
void	*clear_all_slots* ( )
bool	*is_slot_enabled_left* (*int* idx) const
int	*get_slot_type_left* (*int* idx) const
Color	*get_slot_color_left* (*int* idx) const
bool	*is_slot_enabled_right* (*int* idx) const
int	*get_slot_type_right* (*int* idx) const
Color	*get_slot_color_right* (*int* idx) const
void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const
int	*get_connection_output_count* ( )
int	*get_connection_input_count* ( )
Vector2	*get_connection_output_pos* (*int* idx)
int	*get_connection_output_type* (*int* idx)
Color	*get_connection_output_color* (*int* idx)
Vector2	*get_connection_input_pos* (*int* idx)
int	*get_connection_input_type* (*int* idx)
Color	*get_connection_input_color* (*int* idx)
void	*set_show_close_button* (*bool* show)
bool	*is_close_button_visible* ( ) const

9.99.3 Signals

- **raise_request** ( )
- **close_request** ( )
- **dragged** (*Vector2* from, *Vector2* to)
- **offset_changed** ( )

9.99.4 Description

A GraphNode is a container defined by a title. It can have 1 or more input and output slots, which can be enabled (shown) or disabled (not shown) and have different (incompatible) types. Colors can also be assigned to slots. A tuple of input and output slots is defined for each GUI element included in the GraphNode. Input and output connections are left and right slots, but only enabled slots are counted as connections.

9.99.5 Member Function Description

- void **set_title** (*String* title)

Set the title of the GraphNode.

- *String* **get_title ( )** const

Return the title of the GraphNode.

- void **set_slot (** *int* idx, *bool* enable_left, *int* type_left, *Color* color_left, *bool* enable_right, *int* type_right, *Color* color_right **)**

Set the tuple of input/output slots defined by 'idx' ID. 'left' slots are input, 'right' are output. 'type' is an integer defining the type of the slot. Refer to description for the compatibility between slot types.

- void **clear_slot (** *int* idx **)**

Disable input and output slot whose index is 'idx'.

- void **clear_all_slots ( )**

Disable all input and output slots of the GraphNode.

- *bool* **is_slot_enabled_left (** *int* idx **)** const

Return true if left (input) slot 'idx' is enabled. False otherwise.

- *int* **get_slot_type_left (** *int* idx **)** const

Return the (integer) type of left (input) 'idx' slot.

- *Color* **get_slot_color_left (** *int* idx **)** const

Return the color set to 'idx' left (input) slot.

- *bool* **is_slot_enabled_right (** *int* idx **)** const

Return true if right (output) slot 'idx' is enabled. False otherwise.

- *int* **get_slot_type_right (** *int* idx **)** const

Return the (integer) type of right (output) 'idx' slot.

- *Color* **get_slot_color_right (** *int* idx **)** const

Return the color set to 'idx' right (output) slot.

- void **set_offset (** *Vector2* offset **)**

Set the offset of the GraphNode.

- *Vector2* **get_offset ( )** const

Return the offset of the GraphNode.

- *int* **get_connection_output_count ( )**

Return the number of enabled output slots (connections) of the GraphNode.

- *int* **get_connection_input_count ( )**

Return the number of enabled input slots (connections) to the GraphNode.

- *Vector2* **get_connection_output_pos (** *int* idx **)**

Return the position of the output connection 'idx'.

- *int* **get_connection_output_type (** *int* idx **)**

Return the type of the output connection 'idx'.

- *Color* **get_connection_output_color (** *int* idx **)**

Return the color of the output connection 'idx'.

- *Vector2* **get_connection_input_pos (** *int* idx **)**

Return the position of the input connection 'idx'.

- *int* **get_connection_input_type** (*int* idx)

Return the type of the input connection 'idx'.

- *Color* **get_connection_input_color** (*int* idx)

Return the color of the input connection 'idx'.

- void **set_show_close_button** (*bool* show)

Show the close button on the GraphNode if 'show' is true (disabled by default). If enabled, a connection on the signal close_request is needed for the close button to work.

- *bool* **is_close_button_visible** ( ) const

Returns true if the close button is shown. False otherwise.

9.100 GridContainer

Inherits: *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.100.1 Brief Description

Grid container used to arrange elements in a grid like layout

9.100.2 Member Functions

void	*set_columns* (*int* columns)
int	*get_columns* ( ) const

9.100.3 Description

Grid container will arrange its children in a grid like structure, the grid columns are specified using the *set_columns* method and the number of rows will be equal to the number of children in the container divided by the number of columns, for example: if the container has 5 children, and 2 columns, there will be 3 rows in the container. Notice that grid layout will preserve the columns and rows for every size of the container.

9.100.4 Member Function Description

- void **set_columns** (*int* columns)

Sets the numbers of columns in the container, then reorder its children to accommodate the new layout

- *int* **get_columns** ( ) const

Returns the number of columns in this container

9.101 GridMap

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.101.1 Brief Description

9.101.2 Member Functions

void	*set_theme* (*MeshLibrary* theme)
MeshLibrary	*get_theme* ( ) const
void	*set_bake* (*bool* enable)
bool	*is_baking_enabled* ( ) const
void	*set_cell_size* (*float* size)
float	*get_cell_size* ( ) const
void	*set_octant_size* (*int* size)
int	*get_octant_size* ( ) const
void	*set_cell_item* (*int* x, *int* y, *int* z, *int* item, *int* orientation=0)
int	*get_cell_item* (*int* x, *int* y, *int* z) const
int	*get_cell_item_orientation* (*int* x, *int* y, *int* z) const
void	*resource_changed* (*Object* resource)
void	*set_center_x* (*bool* enable)
bool	*get_center_x* ( ) const
void	*set_center_y* (*bool* enable)
bool	*get_center_y* ( ) const
void	*set_center_z* (*bool* enable)
bool	*get_center_z* ( ) const
void	*set_clip* (*bool* enabled, *bool* clipabove=true, *int* floor=0, *int* axis=0)
int	*create_area* (*int* id, *AABB* area)
AABB	*area_get_bounds* (*int* area) const
void	*area_set_exterior_portal* (*int* area, *bool* enable)
void	*area_set_name* (*int* area, *String* name)
String	*area_get_name* (*int* area) const
bool	*area_is_exterior_portal* (*int* area) const
void	*area_set_portal_disable_distance* (*int* area, *float* distance)
float	*area_get_portal_disable_distance* (*int* area) const
void	*area_set_portal_disable_color* (*int* area, *Color* color)
Color	*area_get_portal_disable_color* (*int* area) const
void	*erase_area* (*int* area)
int	*get_unused_area_id* ( ) const
void	*bake_geometry* ( )
void	*set_use_baked_light* (*bool* use)
bool	*is_using_baked_light* ( ) const
void	*clear* ( )

9.101.3 Numeric Constants

- **INVALID_CELL_ITEM = -1**

9.101.4 Member Function Description

- void **set_theme** (*MeshLibrary* theme)
- *MeshLibrary* **get_theme** ( ) const
- void **set_bake** (*bool* enable)
- *bool* **is_baking_enabled** ( ) const
- void **set_cell_size** (*float* size)
- *float* **get_cell_size** ( ) const
- void **set_octant_size** (*int* size)
- *int* **get_octant_size** ( ) const
- void **set_cell_item** (*int* x, *int* y, *int* z, *int* item, *int* orientation=0)
- *int* **get_cell_item** (*int* x, *int* y, *int* z) const
- *int* **get_cell_item_orientation** (*int* x, *int* y, *int* z) const
- void **resource_changed** (*Object* resource)
- void **set_center_x** (*bool* enable)
- *bool* **get_center_x** ( ) const
- void **set_center_y** (*bool* enable)
- *bool* **get_center_y** ( ) const
- void **set_center_z** (*bool* enable)
- *bool* **get_center_z** ( ) const
- void **set_clip** (*bool* enabled, *bool* clipabove=true, *int* floor=0, *int* axis=0)
- *int* **create_area** (*int* id, *AABB* area)
- *AABB* **area_get_bounds** (*int* area) const
- void **area_set_exterior_portal** (*int* area, *bool* enable)
- void **area_set_name** (*int* area, *String* name)
- *String* **area_get_name** (*int* area) const
- *bool* **area_is_exterior_portal** (*int* area) const
- void **area_set_portal_disable_distance** (*int* area, *float* distance)
- *float* **area_get_portal_disable_distance** (*int* area) const
- void **area_set_portal_disable_color** (*int* area, *Color* color)
- *Color* **area_get_portal_disable_color** (*int* area) const
- void **erase_area** (*int* area)
- *int* **get_unused_area_id** ( ) const
- void **bake_geometry** ( )
- void **set_use_baked_light** (*bool* use)
- *bool* **is_using_baked_light** ( ) const
- void **clear** ( )

9.102 GrooveJoint2D

Inherits: *Joint2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.102.1 Brief Description

Groove constraint for 2D physics.

9.102.2 Member Functions

void	*set_length* (*float* length)
float	*get_length* ( ) const
void	*set_initial_offset* (*float* offset)
float	*get_initial_offset* ( ) const

9.102.3 Description

Groove constraint for 2D physics. This is useful for making a body "slide" through a segment placed in another.

9.102.4 Member Function Description

- void **set_length** (*float* length)

Set the length of the groove.

- *float* **get_length** ( ) const

Return the length of the groove.

- void **set_initial_offset** (*float* offset)

Set the initial offset of the groove on body A.

- *float* **get_initial_offset** ( ) const

Set the final offset of the groove on body A.

9.103 HBoxContainer

Inherits: *BoxContainer* < *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.103.1 Brief Description

Horizontal box container.

9.103.2 Description

Horizontal box container. See *BoxContainer*.

9.104 HButtonArray

Inherits: *ButtonArray* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.104.1 Brief Description

Horizontal button array.

9.104.2 Description

Horizontal button array. See *ButtonArray*.

9.105 HingeJoint

Inherits: *Joint* < *Spatial* < *Node* < *Object*

Category: Core

9.105.1 Brief Description

9.105.2 Member Functions

void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const
void	*set_flag* (*int* flag, *bool* enabled)
bool	*get_flag* (*int* flag) const

9.105.3 Numeric Constants

- **PARAM_BIAS = 0**
- **PARAM_LIMIT_UPPER = 1**
- **PARAM_LIMIT_LOWER = 2**
- **PARAM_LIMIT_BIAS = 3**
- **PARAM_LIMIT_SOFTNESS = 4**
- **PARAM_LIMIT_RELAXATION = 5**
- **PARAM_MOTOR_TARGET_VELOCITY = 6**
- **PARAM_MOTOR_MAX_IMPULSE = 7**
- **PARAM_MAX = 8**

- **FLAG_USE_LIMIT = 0**
- **FLAG_ENABLE_MOTOR = 1**
- **FLAG_MAX = 2**

9.105.4 Member Function Description

- void **set_param** (*int* param, *float* value)
- *float* **get_param** (*int* param) const
- void **set_flag** (*int* flag, *bool* enabled)
- *bool* **get_flag** (*int* flag) const

9.106 HScrollBar

Inherits: *ScrollBar* < *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.106.1 Brief Description

Horizontal scroll bar.

9.106.2 Description

Horizontal scroll bar. See *ScrollBar*. This one goes from left (min) to right (max).

9.107 HSeparator

Inherits: *Separator* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.107.1 Brief Description

Horizontal separator.

9.107.2 Description

Horizontal separator. See *Separator*. It is used to separate objects vertically, though (but it looks horizontal!).

9.108 HSlider

Inherits: *Slider* < *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.108.1 Brief Description

Horizontal slider.

9.108.2 Description

Horizontal slider. See *Slider*. This one goes from left (min) to right (max).

9.109 HSplitContainer

Inherits: *SplitContainer* < *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.109.1 Brief Description

Horizontal split container.

9.109.2 Description

Horizontal split container. See *SplitContainer*. This goes from left to right.

9.110 HTTPClient

Inherits: *Reference* < *Object*

Category: Core

9.110.1 Brief Description

Hyper-text transfer protocol client.

9.110.2 Member Functions

Error	*connect* (*String* host, *int* port, *bool* use_ssl=false, *bool* verify_host=true)
void	*set_connection* (*StreamPeer* connection)
int	*request* (*int* method, *String* url, *StringArray* headers, *String* body="")
int	*send_body_text* (*String* body)
int	*send_body_data* (*RawArray* body)
void	*close* ( )
bool	*has_response* ( ) const
bool	*is_response_chunked* ( ) const
int	*get_response_code* ( ) const
StringArray	*get_response_headers* ( )
Dictionary	*get_response_headers_as_dictionary* ( )
int	*get_response_body_length* ( ) const
RawArray	*read_response_body_chunk* ( )
void	*set_read_chunk_size* (*int* bytes)
void	*set_blocking_mode* (*bool* enabled)
bool	*is_blocking_mode_enabled* ( ) const
int	*get_status* ( ) const
Error	*poll* ( )
String	*query_string_from_dict* (*Dictionary* fields)

9.110.3 Numeric Constants

- **METHOD_GET = 0**
- **METHOD_HEAD = 1**
- **METHOD_POST = 2**
- **METHOD_PUT = 3**
- **METHOD_DELETE = 4**
- **METHOD_OPTIONS = 5**
- **METHOD_TRACE = 6**
- **METHOD_CONNECT = 7**
- **METHOD_MAX = 8**
- **STATUS_DISCONNECTED = 0**
- **STATUS_RESOLVING = 1**
- **STATUS_CANT_RESOLVE = 2**
- **STATUS_CONNECTING = 3**
- **STATUS_CANT_CONNECT = 4**
- **STATUS_CONNECTED = 5**
- **STATUS_REQUESTING = 6**
- **STATUS_BODY = 7**
- **STATUS_CONNECTION_ERROR = 8**
- **STATUS_SSL_HANDSHAKE_ERROR = 9**

- **RESPONSE_CONTINUE = 100**
- **RESPONSE_SWITCHING_PROTOCOLS = 101**
- **RESPONSE_PROCESSING = 102**
- **RESPONSE_OK = 200**
- **RESPONSE_CREATED = 201**
- **RESPONSE_ACCEPTED = 202**
- **RESPONSE_NON_AUTHORITATIVE_INFORMATION = 203**
- **RESPONSE_NO_CONTENT = 204**
- **RESPONSE_RESET_CONTENT = 205**
- **RESPONSE_PARTIAL_CONTENT = 206**
- **RESPONSE_MULTI_STATUS = 207**
- **RESPONSE_IM_USED = 226**
- **RESPONSE_MULTIPLE_CHOICES = 300**
- **RESPONSE_MOVED_PERMANENTLY = 301**
- **RESPONSE_FOUND = 302**
- **RESPONSE_SEE_OTHER = 303**
- **RESPONSE_NOT_MODIFIED = 304**
- **RESPONSE_USE_PROXY = 305**
- **RESPONSE_TEMPORARY_REDIRECT = 307**
- **RESPONSE_BAD_REQUEST = 400**
- **RESPONSE_UNAUTHORIZED = 401**
- **RESPONSE_PAYMENT_REQUIRED = 402**
- **RESPONSE_FORBIDDEN = 403**
- **RESPONSE_NOT_FOUND = 404**
- **RESPONSE_METHOD_NOT_ALLOWED = 405**
- **RESPONSE_NOT_ACCEPTABLE = 406**
- **RESPONSE_PROXY_AUTHENTICATION_REQUIRED = 407**
- **RESPONSE_REQUEST_TIMEOUT = 408**
- **RESPONSE_CONFLICT = 409**
- **RESPONSE_GONE = 410**
- **RESPONSE_LENGTH_REQUIRED = 411**
- **RESPONSE_PRECONDITION_FAILED = 412**
- **RESPONSE_REQUEST_ENTITY_TOO_LARGE = 413**
- **RESPONSE_REQUEST_URI_TOO_LONG = 414**
- **RESPONSE_UNSUPPORTED_MEDIA_TYPE = 415**
- **RESPONSE_REQUESTED_RANGE_NOT_SATISFIABLE = 416**

- **RESPONSE_EXPECTATION_FAILED = 417**
- **RESPONSE_UNPROCESSABLE_ENTITY = 422**
- **RESPONSE_LOCKED = 423**
- **RESPONSE_FAILED_DEPENDENCY = 424**
- **RESPONSE_UPGRADE_REQUIRED = 426**
- **RESPONSE_INTERNAL_SERVER_ERROR = 500**
- **RESPONSE_NOT_IMPLEMENTED = 501**
- **RESPONSE_BAD_GATEWAY = 502**
- **RESPONSE_SERVICE_UNAVAILABLE = 503**
- **RESPONSE_GATEWAY_TIMEOUT = 504**
- **RESPONSE_HTTP_VERSION_NOT_SUPPORTED = 505**
- **RESPONSE_INSUFFICIENT_STORAGE = 507**
- **RESPONSE_NOT_EXTENDED = 510**

9.110.4 Description

Hyper-text transfer protocol client. Supports SSL and SSL server certificate verification.

Can be reused to connect to different hosts and make many requests.

9.110.5 Member Function Description

- Error **connect** (*String* host, *int* port, *bool* use_ssl=false, *bool* verify_host=true)

Connect to a host. This needs to be done before any requests are sent.

The host should not have http:// prepended but will strip the protocol identifier if provided.

verify_host will check the SSL identity of the host if set to true.

- void **set_connection** (*StreamPeer* connection)

Set connection to use, for this client.

- *int* **request** (*int* method, *String* url, *StringArray* headers, *String* body="")

Sends a request to the connected host. The url is what is normally behind the hostname, i.e. in `http://somehost.com/index.php`, url would be "index.php".

Headers are HTTP request headers.

To create a POST request with query strings to push to the server, do:

```
var fields = {"username" : "user", "password" : "pass"}
var queryString = httpClient.query_string_from_dict(fields)
var headers = :ref:`"Content-Type: application/x-www-form-urlencoded", "Content-
↪Length: " + str(queryString.length())<class_"content-type: application/x-www-form-
↪urlencoded", "content-length: " + str(querystring.length())>`
var result = httpClient.request(httpClient.METHOD_POST, "index.php", headers,␣
↪queryString)
```

- *int* **send_body_text** (*String* body)

Stub function

- *int* **send_body_data** (*RawArray* body)

Stub function

- void **close** ( )

Cloces the current connection, allows for reusal of *HTTPClient*.

- *bool* **has_response** ( ) const

Return whether this *HTTPClient* has a response available.

- *bool* **is_response_chunked** ( ) const

Return whether this *HTTPClient* has a response that is chunked.

- *int* **get_response_code** ( ) const

Return the HTTP status code of the response.

- *StringArray* **get_response_headers** ( )

Return the response headers.

- *Dictionary* **get_response_headers_as_dictionary** ( )

Returns all response headers as dictionary where the case-sensitivity of the keys and values is kept like the server delivers it. A value is a simple String, this string can have more than one value where "; " is used as separator.

Structure: ("key":"value1; value2")

Example: (content-length:12), (Content-Type:application/json; charset=UTF-8)

- *int* **get_response_body_length** ( ) const

Return the response's body length.

- *RawArray* **read_response_body_chunk** ( )

Reads one chunk from the response.

- void **set_read_chunk_size** (*int* bytes)

Sets the size of the buffer used and maximum bytes to read per iteration. see *read_response_body_chunk*

- void **set_blocking_mode** (*bool* enabled)

If set to true, execution will block until all data is read from the response.

- *bool* **is_blocking_mode_enabled** ( ) const

Return whether blocking mode is enabled.

- *int* **get_status** ( ) const

Returns a status string like STATUS_REQUESTING. Need to call *poll* in order to get status updates.

- Error **poll** ( )

This needs to be called in order to have any request processed. Check results with *get_status*

- *String* **query_string_from_dict** (*Dictionary* fields)

Generates a GET/POST application/x-www-form-urlencoded style query string from a provided dictionary, e.g.:

```
var fields = {"username": "user", "password": "pass"}
String queryString = httpClient.query_string_from_dict(fields)
returns:= "username=user&password=pass"
```

9.111 Image

Category: Built-In Types

9.111.1 Brief Description

Image datatype.

9.111.2 Member Functions

void	*blit_rect* (*Image* src, *Rect2* src_rect, *Vector2* dest=0)
void	*brush_transfer* (*Image* src, *Image* brush, *Vector2* pos=0)
Image	*brushed* (*Image* src, *Image* brush, *Vector2* pos=0)
Image	*compressed* (*int* format=0)
Image	*converted* (*int* format=0)
Image	*decompressed* ( )
bool	*empty* ( )
void	*fix_alpha_edges* ( )
RawArray	*get_data* ( )
int	*get_format* ( )
int	*get_height* ( )
Color	*get_pixel* (*int* x, *int* y, *int* mipmap_level=0)
Image	*get_rect* (*Rect2* area=0)
Rect2	*get_used_rect* ( )
int	*get_width* ( )
int	*load* (*String* path=0)
void	*put_pixel* (*int* x, *int* y, *Color* color, *int* mipmap_level=0)
Image	*resized* (*int* x, *int* y, *int* interpolation=1)
int	*save_png* (*String* path=0)
Image	*Image* (*int* width, *int* height, *bool* mipmaps, *int* format)

9.111.3 Numeric Constants

- **COMPRESS_BC = 0**
- **COMPRESS_PVRTC2 = 1**
- **COMPRESS_PVRTC4 = 2**
- **COMPRESS_ETC = 3**
- **FORMAT_GRAYSCALE = 0**
- **FORMAT_INTENSITY = 1**
- **FORMAT_GRAYSCALE_ALPHA = 2**
- **FORMAT_RGB = 3**
- **FORMAT_RGBA = 4**
- **FORMAT_INDEXED = 5**
- **FORMAT_INDEXED_ALPHA = 6**

- **FORMAT_YUV_422 = 7**
- **FORMAT_YUV_444 = 8**
- **FORMAT_BC1 = 9**
- **FORMAT_BC2 = 10**
- **FORMAT_BC3 = 11**
- **FORMAT_BC4 = 12**
- **FORMAT_BC5 = 13**
- **FORMAT_PVRTC2 = 14**
- **FORMAT_PVRTC2_ALPHA = 15**
- **FORMAT_PVRTC4 = 16**
- **FORMAT_PVRTC4_ALPHA = 17**
- **FORMAT_ETC = 18**
- **FORMAT_ATC = 19**
- **FORMAT_ATC_ALPHA_EXPLICIT = 20**
- **FORMAT_ATC_ALPHA_INTERPOLATED = 21**
- **FORMAT_CUSTOM = 22**

9.111.4 Description

Built in native image datatype. Contains image data, which can be converted to a texture, and several functions to interact with it.

9.111.5 Member Function Description

- void **blit_rect** (*Image* src, *Rect2* src_rect, *Vector2* dest=0)
- void **brush_transfer** (*Image* src, *Image* brush, *Vector2* pos=0)
- *Image* **brushed** (*Image* src, *Image* brush, *Vector2* pos=0)
- *Image* **compressed** (*int* format=0)
- *Image* **converted** (*int* format=0)
- *Image* **decompressed** ( )
- *bool* **empty** ( )
- void **fix_alpha_edges** ( )
- *RawArray* **get_data** ( )
- *int* **get_format** ( )
- *int* **get_height** ( )
- *Color* **get_pixel** (*int* x, *int* y, *int* mipmap_level=0)
- *Image* **get_rect** (*Rect2* area=0)
- *Rect2* **get_used_rect** ( )

- *int* **get_width** ( )
- *int* **load** (*String* path=0)
- void **put_pixel** (*int* x, *int* y, *Color* color, *int* mipmap_level=0)
- *Image* **resized** (*int* x, *int* y, *int* interpolation=1)
- *int* **save_png** (*String* path=0)
- *Image* **Image** (*int* width, *int* height, *bool* mipmaps, *int* format)

Create an empty image of a specific size and format.

9.112 ImageTexture

Inherits: *Texture* < *Resource* < *Reference* < *Object*

Category: Core

9.112.1 Brief Description

9.112.2 Member Functions

void	*create* (*int* width, *int* height, *int* format, *int* flags=7)
void	*create_from_image* (*Image* image, *int* flags=7)
int	*get_format* ( ) const
void	*load* (*String* path)
void	*set_data* (*Image* image)
Image	*get_data* ( ) const
void	*set_storage* (*int* mode)
int	*get_storage* ( ) const
void	*set_lossy_storage_quality* (*float* quality)
float	*get_lossy_storage_quality* ( ) const
void	*fix_alpha_edges* ( )
void	*premultiply_alpha* ( )
void	*normal_to_xy* ( )
void	*shrink_x2_and_keep_size* ( )
void	*set_size_override* (*Vector2* size)

9.112.3 Numeric Constants

- **STORAGE_RAW = 0**
- **STORAGE_COMPRESS_LOSSY = 1**
- **STORAGE_COMPRESS_LOSSLESS = 2**

9.112.4 Member Function Description

- void **create** (*int* width, *int* height, *int* format, *int* flags=7)
- void **create_from_image** (*Image* image, *int* flags=7)

- *int* **get_format** ( ) const
- void **load** (*String* path)
- void **set_data** (*Image* image)
- *Image* **get_data** ( ) const
- void **set_storage** (*int* mode)
- *int* **get_storage** ( ) const
- void **set_lossy_storage_quality** (*float* quality)
- *float* **get_lossy_storage_quality** ( ) const
- void **fix_alpha_edges** ( )
- void **premultiply_alpha** ( )
- void **normal_to_xy** ( )
- void **shrink_x2_and_keep_size** ( )
- void **set_size_override** (*Vector2* size)

9.113 ImmediateGeometry

Inherits: *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.113.1 Brief Description

9.113.2 Member Functions

void	*begin* (*int* primitive, *Texture* texture)
void	*set_normal* (*Vector3* normal)
void	*set_tangent* (*Plane* tangent)
void	*set_color* (*Color* color)
void	*set_uv* (*Vector2* uv)
void	*set_uv2* (*Vector2* uv)
void	*add_vertex* (*Vector3* pos)
void	*add_sphere* (*int* lats, *int* lons, *float* radius)
void	*end* ( )
void	*clear* ( )

9.113.3 Member Function Description

- void **begin** (*int* primitive, *Texture* texture)
- void **set_normal** (*Vector3* normal)
- void **set_tangent** (*Plane* tangent)
- void **set_color** (*Color* color)
- void **set_uv** (*Vector2* uv)

- void **set_uv2** (*Vector2* uv)
- void **add_vertex** (*Vector3* pos)
- void **add_sphere** (*int* lats, *int* lons, *float* radius)
- void **end** ( )
- void **clear** ( )

9.114 Input

Inherits: *Object*

Inherited By: *InputDefault*

Category: Core

9.114.1 Brief Description

A Singleton that deals with inputs.

9.114.2 Member Functions

bool	*is_key_pressed* (*int* scancode)
bool	*is_mouse_button_pressed* (*int* button)
bool	*is_joy_button_pressed* (*int* device, *int* button)
bool	*is_action_pressed* (*String* action)
void	*add_joy_mapping* (*String* mapping, *bool* update_existing=false)
void	*remove_joy_mapping* (*String* guid)
bool	*is_joy_known* (*int* device)
float	*get_joy_axis* (*int* device, *int* axis)
String	*get_joy_name* (*int* device)
String	*get_joy_guid* (*int* device) const
Vector3	*get_accelerometer* ( )
Vector2	*get_mouse_speed* ( ) const
int	*get_mouse_button_mask* ( ) const
void	*set_mouse_mode* (*int* mode)
int	*get_mouse_mode* ( ) const
void	*warp_mouse_pos* (*Vector2* to)
void	*action_press* (*String* action)
void	*action_release* (*String* action)
void	*set_custom_mouse_cursor* (*Texture* image, *Vector2* hotspot=Vector2(0,0))

9.114.3 Signals

- **joy_connection_changed** (*int* index, *bool* connected)

9.114.4 Numeric Constants

- **MOUSE_MODE_VISIBLE = 0** — Makes the mouse cursor visible if it is hidden.
- **MOUSE_MODE_HIDDEN = 1** — Makes the mouse cursor hidden if it is visible.
- **MOUSE_MODE_CAPTURED = 2** — Captures the mouse. The mouse will be hidden and unable to leave the game window. But it will still register movement and mouse button presses.

9.114.5 Description

A Singleton that deals with inputs. This includes key presses, mouse buttons and movement, joysticks, and input actions.

9.114.6 Member Function Description

- *bool* **is_key_pressed** (*int* scancode)

Returns true or false depending on whether the key is pressed or not. You can pass KEY_*, which are pre-defined constants listed in *@Global Scope*.

- *bool* **is_mouse_button_pressed** (*int* button)

Returns true or false depending on whether mouse button is pressed or not. You can pass BUTTON_*, which are pre-defined constants listed in *@Global Scope*.

- *bool* **is_joy_button_pressed** (*int* device, *int* button)

Returns if the joystick button at the given index is currently pressed. (see JOY_* constants in *@Global Scope*)

- *bool* **is_action_pressed** (*String* action)

Returns true or false depending on whether the action event is pressed. Actions and their events can be set in the Project Settings / Input Map tab. Or be set with *InputMap*.

- void **add_joy_mapping** (*String* mapping, *bool* update_existing=false)

Add a new mapping entry (in SDL2 format) to the mapping database. Optionally update already connected devices.

- void **remove_joy_mapping** (*String* guid)

Removes all mappings from the internal db that match the given uid.

- *bool* **is_joy_known** (*int* device)

Returns if the specified device is known by the system. This means that it sets all button and axis indices exactly as defined in the JOY_* constants (see *@Global Scope*). Unknown joysticks are not expected to match these constants, but you can still retrieve events from them.

- *float* **get_joy_axis** (*int* device, *int* axis)

Returns the current value of the joystick axis at given index (see JOY_* constants in *@Global Scope*)

- *String* **get_joy_name** (*int* device)

Returns the name of the joystick at the specified device index

- *String* **get_joy_guid** (*int* device) const

Returns a SDL2 compatible device guid on platforms that use gamepad remapping. Returns "Default Gamepad" otherwise.

- *Vector3* **get_accelerometer** ( )

If the device has an accelerometer, this will return the movement.

- *Vector2* **get_mouse_speed ( )** const

Returns the mouse speed.

- *int* **get_mouse_button_mask ( )** const

Returns mouse buttons as a bitmask. If multiple mouse buttons are pressed at the same time the bits are added together.

- void **set_mouse_mode (** *int* mode **)**

Set the mouse mode. See the constants for more information.

- *int* **get_mouse_mode ( )** const

Return the mouse mode. See the constants for more information.

- void **warp_mouse_pos (** *Vector2* to **)**

Sets the mouse position to the specified vector.

- void **action_press (** *String* action **)**

This will simulate pressing the specificed action.

- void **action_release (** *String* action **)**

If the specified action is already pressed, this will release it.

- void **set_custom_mouse_cursor (** *Texture* image, *Vector2* hotspot=Vector2(0,0) **)**

Set a custom mouse cursor image, which is only visible inside the game window. The hotspot can also be specified.

9.115 InputDefault

Inherits: *Input* **<** *Object*

Category: Core

9.115.1 Brief Description

9.116 InputEvent

Category: Built-In Types

9.116.1 Brief Description

Built-in input event data.

9.116.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.116.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**

9.116.4 Numeric Constants

- **NONE = 0** — Empty input event.
- **KEY = 1** — Key event.
- **MOUSE_MOTION = 2** — Mouse motion event.
- **MOUSE_BUTTON = 3** — Mouse button event.
- **JOYSTICK_MOTION = 4** — Joystick motion event.
- **JOYSTICK_BUTTON = 5** — Joystick button event.
- **SCREEN_TOUCH = 6**
- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.116.5 Description

Built-in input event data. InputEvent is a built-in engine datatype, given that it's passed around and used so much. Depending on it's type, the members contained can be different, so read the documentation well!. Input events can also represent actions (editable from the project settings).

9.116.6 Member Function Description

- *bool* **is_action** (*String* action)

Return if this input event matches a pre-defined action, no matter the type.

- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)
- *bool* **is_echo** ( )

Return if this input event is an echo event (usually for key events).

- *bool* **is_pressed** ( )

Return if this input event is pressed (for key, mouse, joy button or screen press events).

- void **set_as_action** (*String* action, *bool* pressed)

9.117 InputEventAction

Category: Built-In Types

9.117.1 Brief Description

9.117.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.117.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**

9.117.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE_MOTION = 2**
- **MOUSE_BUTTON = 3**
- **JOYSTICK_MOTION = 4**
- **JOYSTICK_BUTTON = 5**
- **SCREEN_TOUCH = 6**
- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.117.5 Member Function Description

- *bool* **is_action** (*String* action)
- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)

- *bool* **is_echo** ( )
- *bool* **is_pressed** ( )
- void **set_as_action** (*String* action, *bool* pressed)

9.118 InputEventJoystickButton

Category: Built-In Types

9.118.1 Brief Description

9.118.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.118.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**
- *int* **button_index**
- *bool* **pressed**
- *float* **pressure**

9.118.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE_MOTION = 2**
- **MOUSE_BUTTON = 3**
- **JOYSTICK_MOTION = 4**
- **JOYSTICK_BUTTON = 5**
- **SCREEN_TOUCH = 6**
- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.118.5 Member Function Description

- *bool* **is_action** (*String* action)
- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)
- *bool* **is_echo** ( )
- *bool* **is_pressed** ( )
- void **set_as_action** (*String* action, *bool* pressed)

9.119 InputEventJoystickMotion

Category: Built-In Types

9.119.1 Brief Description

9.119.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.119.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**
- *int* **axis**
- *float* **value**

9.119.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE_MOTION = 2**
- **MOUSE_BUTTON = 3**
- **JOYSTICK_MOTION = 4**
- **JOYSTICK_BUTTON = 5**
- **SCREEN_TOUCH = 6**

- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.119.5 Member Function Description

- *bool* **is_action** (*String* action)
- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)
- *bool* **is_echo** ( )
- *bool* **is_pressed** ( )
- void **set_as_action** (*String* action, *bool* pressed)

9.120 InputEventKey

Category: Built-In Types

9.120.1 Brief Description

9.120.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.120.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**
- *bool* **shift**
- *bool* **alt**
- *bool* **control**
- *bool* **meta**
- *bool* **pressed**
- *bool* **echo**
- *int* **scancode**
- *int* **unicode**

9.120.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE_MOTION = 2**
- **MOUSE_BUTTON = 3**
- **JOYSTICK_MOTION = 4**
- **JOYSTICK_BUTTON = 5**
- **SCREEN_TOUCH = 6**
- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.120.5 Member Function Description

- *bool* **is_action** (*String* action)
- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)
- *bool* **is_echo** ( )
- *bool* **is_pressed** ( )
- void **set_as_action** (*String* action, *bool* pressed)

9.121 InputEventMouseButton

Category: Built-In Types

9.121.1 Brief Description

9.121.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.121.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**

- *bool* **shift**
- *bool* **alt**
- *bool* **control**
- *bool* **meta**
- *int* **button_mask**
- *int* **x**
- *int* **y**
- *Vector2* **pos**
- *int* **global_x**
- *int* **global_y**
- *Vector2* **global_pos**
- *int* **button_index**
- *bool* **pressed**
- *bool* **doubleclick**

9.121.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE_MOTION = 2**
- **MOUSE_BUTTON = 3**
- **JOYSTICK_MOTION = 4**
- **JOYSTICK_BUTTON = 5**
- **SCREEN_TOUCH = 6**
- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.121.5 Member Function Description

- *bool* **is_action** (*String* action)
- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)
- *bool* **is_echo** ( )
- *bool* **is_pressed** ( )
- void **set_as_action** (*String* action, *bool* pressed)

9.122 InputEventMouseMotion

Category: Built-In Types

9.122.1 Brief Description

9.122.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.122.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**
- *bool* **shift**
- *bool* **alt**
- *bool* **control**
- *bool* **meta**
- *int* **button_mask**
- *int* **x**
- *int* **y**
- *Vector2* **pos**
- *int* **global_x**
- *int* **global_y**
- *Vector2* **global_pos**
- *int* **relative_x**
- *int* **relative_y**
- *Vector2* **relative_pos**
- *float* **speed_x**
- *float* **speed_y**
- *Vector2* **speed**

9.122.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE_MOTION = 2**
- **MOUSE_BUTTON = 3**
- **JOYSTICK_MOTION = 4**
- **JOYSTICK_BUTTON = 5**
- **SCREEN_TOUCH = 6**
- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.122.5 Member Function Description

- *bool* **is_action** (*String* action)
- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)
- *bool* **is_echo** ( )
- *bool* **is_pressed** ( )
- void **set_as_action** (*String* action, *bool* pressed)

9.123 InputEventScreenDrag

Category: Built-In Types

9.123.1 Brief Description

9.123.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.123.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**

- *int* **index**
- *float* **x**
- *float* **y**
- *Vector2* **pos**
- *float* **relative_x**
- *float* **relative_y**
- *Vector2* **relative_pos**
- *float* **speed_x**
- *float* **speed_y**
- *Vector2* **speed**

9.123.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE_MOTION = 2**
- **MOUSE_BUTTON = 3**
- **JOYSTICK_MOTION = 4**
- **JOYSTICK_BUTTON = 5**
- **SCREEN_TOUCH = 6**
- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.123.5 Member Function Description

- *bool* **is_action** (*String* action)
- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)
- *bool* **is_echo** ( )
- *bool* **is_pressed** ( )
- void **set_as_action** (*String* action, *bool* pressed)

9.124 InputEventScreenTouch

Category: Built-In Types

9.124.1 Brief Description

9.124.2 Member Functions

bool	*is_action* (*String* action)
bool	*is_action_pressed* (*String* is_action_pressed)
bool	*is_action_released* (*String* is_action_released)
bool	*is_echo* ( )
bool	*is_pressed* ( )
void	*set_as_action* (*String* action, *bool* pressed)

9.124.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**
- *int* **index**
- *float* **x**
- *float* **y**
- *Vector2* **pos**
- *bool* **pressed**

9.124.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE_MOTION = 2**
- **MOUSE_BUTTON = 3**
- **JOYSTICK_MOTION = 4**
- **JOYSTICK_BUTTON = 5**
- **SCREEN_TOUCH = 6**
- **SCREEN_DRAG = 7**
- **ACTION = 8**

9.124.5 Member Function Description

- *bool* **is_action** (*String* action)
- *bool* **is_action_pressed** (*String* is_action_pressed)
- *bool* **is_action_released** (*String* is_action_released)
- *bool* **is_echo** ( )
- *bool* **is_pressed** ( )

- void **set_as_action** (*String* action, *bool* pressed)

9.125 InputMap

Inherits: *Object*

Category: Core

9.125.1 Brief Description

Singleton that manages actions.

9.125.2 Member Functions

bool	*has_action* (*String* action) const
int	*get_action_id* (*String* action) const
String	*get_action_from_id* (*int* id) const
Array	*get_actions* ( )
void	*add_action* (*String* action)
void	*erase_action* (*String* action)
void	*action_add_event* (*String* action, *InputEvent* event)
bool	*action_has_event* (*String* action, *InputEvent* event)
void	*action_erase_event* (*String* action, *InputEvent* event)
Array	*get_action_list* (*String* action)
bool	*event_is_action* (*InputEvent* event, *String* action) const
void	*load_from_globals* ( )

9.125.3 Description

Singleton that manages actions. InputMap has a list of the actions used in InputEvent, which can be modified.

9.125.4 Member Function Description

- *bool* **has_action** (*String* action) const

Whether this InputMap has an action with name "action".

- *int* **get_action_id** (*String* action) const

Return the id of an action.

- *String* **get_action_from_id** (*int* id) const

Return the action from an id.

- *Array* **get_actions** ( )

Return an *Array* of all actions in the *InputMap*.

- void **add_action** (*String* action)

Add an action to the *InputMap*.

- void **erase_action** (*String* action)

Remove an action from the *InputMap*.

- void **action_add_event** (*String* action, *InputEvent* event)

Add an *InputEvent* to action. This *InputEvent* will trigger the action.

- *bool* **action_has_event** (*String* action, *InputEvent* event)

Whether an action has an *InputEvent* associated with it.

- void **action_erase_event** (*String* action, *InputEvent* event)

Remove an *InputEvent* from an action.

- *Array* **get_action_list** (*String* action)

Return an *Array* of :ref:‘InputEvent<class_inputevent>‘s associated with an action.

- *bool* **event_is_action** (*InputEvent* event, *String* action) const
- void **load_from_globals** ( )

Clears the *InputMap* and loads it from *Globals*.

9.126 InstancePlaceholder

Inherits: *Node* < *Object*

Category: Core

9.126.1 Brief Description

9.126.2 Member Functions

void	*replace_by_instance* (*PackedScene* custom_scene=NULL)
String	*get_instance_path* ( ) const

9.126.3 Member Function Description

- void **replace_by_instance** (*PackedScene* custom_scene=NULL)
- *String* **get_instance_path** ( ) const

9.127 int

Category: Built-In Types

9.127.1 Brief Description

Integer built-in type.

9.127.2 Member Functions

int	int (*bool* from)
int	int (*float* from)
int	int (*String* from)

9.127.3 Description

Integer built-in type.

9.127.4 Member Function Description

- *int* **int** (*bool* from)

Cast a *bool* value to an integer value, `int(true)` will be equals to 1 and `int(false)` will be equals to 0.

- *int* **int** (*float* from)

Cast a float value to an integer value, this method simply removes the number fractions, so for example `int(2.7)` will be equals to 2, `int(.1)` will be equals to 0 and `int(-2.7)` will be equals to -2.

- *int* **int** (*String* from)

Cast a *String* value to an integer value, this method is an integer parser from a string, so calling this method with an invalid integer string will return 0, a valid string will be something like `'1.7'`. This method will ignore all non-number characters, so calling `int('1e3')` will return 13.

9.128 IntArray

Category: Built-In Types

9.128.1 Brief Description

Integer Array.

9.128.2 Member Functions

void	*push_back* (*int* integer)
void	*resize* (*int* idx)
void	*set* (*int* idx, *int* integer)
int	*size* ( )
IntArray	*IntArray* (*Array* from)

9.128.3 Description

Integer Array. Array of integers. Can only contain integers. Optimized for memory usage, can't fragment the memory.

9.128.4 Member Function Description

- void **push_back** (*int* integer)

Append a value to the array.

- void **resize** (*int* idx)

Resize the array.

- void **set** (*int* idx, *int* integer)

Set an index in the array.

- *int* **size** ( )

Return the array size.

- *IntArray* **IntArray** (*Array* from)

Create from a generic array.

9.129 InterpolatedCamera

Inherits: *Camera* < *Spatial* < *Node* < *Object*

Category: Core

9.129.1 Brief Description

9.129.2 Member Functions

void	*set_target_path* (*NodePath* target_path)
NodePath	*get_target_path* ( ) const
void	*set_target* (*Camera* target)
void	*set_speed* (*float* speed)
float	*get_speed* ( ) const
void	*set_interpolation_enabled* (*bool* target_path)
bool	*is_interpolation_enabled* ( ) const

9.129.3 Member Function Description

- void **set_target_path** (*NodePath* target_path)
- *NodePath* **get_target_path** ( ) const
- void **set_target** (*Camera* target)
- void **set_speed** (*float* speed)
- *float* **get_speed** ( ) const
- void **set_interpolation_enabled** (*bool* target_path)
- *bool* **is_interpolation_enabled** ( ) const

9.130 IP

Inherits: *Object*

Inherited By: *IP_Unix*

Category: Core

9.130.1 Brief Description

IP Protocol support functions.

9.130.2 Member Functions

String	*resolve_hostname* (*String* host)
int	*resolve_hostname_queue_item* (*String* host)
int	*get_resolve_item_status* (*int* id) const
String	*get_resolve_item_address* (*int* id) const
void	*erase_resolve_item* (*int* id)
Array	*get_local_addresses* ( ) const

9.130.3 Numeric Constants

- **RESOLVER_STATUS_NONE = 0**
- **RESOLVER_STATUS_WAITING = 1**
- **RESOLVER_STATUS_DONE = 2**
- **RESOLVER_STATUS_ERROR = 3**
- **RESOLVER_MAX_QUERIES = 32**
- **RESOLVER_INVALID_ID = -1**

9.130.4 Description

IP contains some support functions for the IPv4 protocol. TCP/IP support is in different classes (see *StreamPeerTCP* and *TCP_Server*). IP provides hostname resolution support, both blocking and threaded.

9.130.5 Member Function Description

- *String* **resolve_hostname** (*String* host)

Resolve a given hostname, blocking. Resolved hostname is returned as an IP.

- *int* **resolve_hostname_queue_item** (*String* host)

Create a queue item for resolving a given hostname. The queue ID is returned, or RESOLVER_INVALID_ID on error.

- *int* **get_resolve_item_status** (*int* id) const

Return the status of hostname queued for resolving, given it's queue ID. Returned status can be any of the RESOLVER_STATUS_* enumeration.

- *String* **get_resolve_item_address** (*int* id) const

Return a resolved item address, or an empty string if an error happened or resolution didn't happen yet (see *get_resolve_item_status*).

- void **erase_resolve_item** (*int* id)

Erase a queue ID, removing it from the queue if needed. This should be used after a queue is completed to free it and enable more queries to happen.

- *Array* **get_local_addresses** ( ) const

9.131 IP_Unix

Inherits: *IP* < *Object*

Category: Core

9.131.1 Brief Description

9.132 ItemList

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.132.1 Brief Description

9.132.2 Member Functions

void	*add_item* (*String* text, *Texture* icon=NULL, *bool* selectable=true)
void	*add_icon_item* (*Texture* icon, *bool* selectable=true)
void	*set_item_text* (*int* idx, *String* text)
String	*get_item_text* (*int* idx) const
void	*set_item_icon* (*int* idx, *Texture* icon)
Texture	*get_item_icon* (*int* idx) const
void	*set_item_selectable* (*int* idx, *bool* selectable)
bool	*is_item_selectable* (*int* idx) const
void	*set_item_disabled* (*int* idx, *bool* disabled)
bool	*is_item_disabled* (*int* idx) const
void	*set_item_metadata* (*int* idx, var metadata)
void	*get_item_metadata* (*int* idx) const
void	*set_item_custom_bg_color* (*int* idx, *Color* custom_bg_color)
Color	*get_item_custom_bg_color* (*int* idx) const
void	*set_item_tooltip* (*int* idx, *String* tooltip)
String	*get_item_tooltip* (*int* idx) const
void	*select* (*int* idx, *bool* single=true)
void	*unselect* (*int* idx)
bool	*is_selected* (*int* idx) const
int	*get_item_count* ( ) const

Continúa en la página siguiente

Tabla 9.12 – proviene de la página anterior

void	*remove_item* (*int* idx)
void	*clear* ( )
void	*sort_items_by_text* ( )
void	*set_fixed_column_width* (*int* width)
int	*get_fixed_column_width* ( ) const
void	*set_max_text_lines* (*int* lines)
int	*get_max_text_lines* ( ) const
void	*set_max_columns* (*int* amount)
int	*get_max_columns* ( ) const
void	*set_select_mode* (*int* mode)
int	*get_select_mode* ( ) const
void	*set_icon_mode* (*int* mode)
int	*get_icon_mode* ( ) const
void	*set_min_icon_size* (*Vector2* size)
Vector2	*get_min_icon_size* ( ) const
void	*ensure_current_is_visible* ( )

9.132.3 Signals

- **item_activated** (*int* index)
- **multi_selected** (*int* index, *bool* selected)
- **item_selected** (*int* index)

9.132.4 Numeric Constants

- **ICON_MODE_TOP = 0**
- **ICON_MODE_LEFT = 1**
- **SELECT_SINGLE = 0**
- **SELECT_MULTI = 1**

9.132.5 Member Function Description

- void **add_item** (*String* text, *Texture* icon=NULL, *bool* selectable=true)
- void **add_icon_item** (*Texture* icon, *bool* selectable=true)
- void **set_item_text** (*int* idx, *String* text)
- *String* **get_item_text** (*int* idx) const
- void **set_item_icon** (*int* idx, *Texture* icon)
- *Texture* **get_item_icon** (*int* idx) const
- void **set_item_selectable** (*int* idx, *bool* selectable)
- *bool* **is_item_selectable** (*int* idx) const
- void **set_item_disabled** (*int* idx, *bool* disabled)
- *bool* **is_item_disabled** (*int* idx) const

- void **set_item_metadata** (*int* idx, var metadata)
- void **get_item_metadata** (*int* idx) const
- void **set_item_custom_bg_color** (*int* idx, *Color* custom_bg_color)
- *Color* **get_item_custom_bg_color** (*int* idx) const
- void **set_item_tooltip** (*int* idx, *String* tooltip)
- *String* **get_item_tooltip** (*int* idx) const
- void **select** (*int* idx, *bool* single=true)
- void **unselect** (*int* idx)
- *bool* **is_selected** (*int* idx) const
- *int* **get_item_count** ( ) const
- void **remove_item** (*int* idx)
- void **clear** ( )
- void **sort_items_by_text** ( )
- void **set_fixed_column_width** (*int* width)
- *int* **get_fixed_column_width** ( ) const
- void **set_max_text_lines** (*int* lines)
- *int* **get_max_text_lines** ( ) const
- void **set_max_columns** (*int* amount)
- *int* **get_max_columns** ( ) const
- void **set_select_mode** (*int* mode)
- *int* **get_select_mode** ( ) const
- void **set_icon_mode** (*int* mode)
- *int* **get_icon_mode** ( ) const
- void **set_min_icon_size** (*Vector2* size)
- *Vector2* **get_min_icon_size** ( ) const
- void **ensure_current_is_visible** ( )

9.133 Joint

Inherits: *Spatial* < *Node* < *Object*

Inherited By: *ConeTwistJoint*, *SliderJoint*, *Generic6DOFJoint*, *HingeJoint*, *PinJoint*

Category: Core

9.133.1 Brief Description

9.133.2 Member Functions

void	*set_node_a* (*NodePath* node)
NodePath	*get_node_a* ( ) const
void	*set_node_b* (*NodePath* node)
NodePath	*get_node_b* ( ) const
void	*set_solver_priority* (*int* priority)
int	*get_solver_priority* ( ) const
void	*set_exclude_nodes_from_collision* (*bool* enable)
bool	*get_exclude_nodes_from_collision* ( ) const

9.133.3 Member Function Description

- void **set_node_a** (*NodePath* node)
- *NodePath* **get_node_a** ( ) const
- void **set_node_b** (*NodePath* node)
- *NodePath* **get_node_b** ( ) const
- void **set_solver_priority** (*int* priority)
- *int* **get_solver_priority** ( ) const
- void **set_exclude_nodes_from_collision** (*bool* enable)
- *bool* **get_exclude_nodes_from_collision** ( ) const

9.134 Joint2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Inherited By: *PinJoint2D*, *DampedSpringJoint2D*, *GrooveJoint2D*

Category: Core

9.134.1 Brief Description

Base node for all joint constraints in 2D physics.

9.134.2 Member Functions

void	*set_node_a* (*NodePath* node)
NodePath	*get_node_a* ( ) const
void	*set_node_b* (*NodePath* node)
NodePath	*get_node_b* ( ) const
void	*set_bias* (*float* bias)
float	*get_bias* ( ) const
void	*set_exclude_nodes_from_collision* (*bool* enable)
bool	*get_exclude_nodes_from_collision* ( ) const

9.134.3 Description

Base node for all joint constraints in 2D physics. Joints take 2 bodies and apply a custom constraint.

9.134.4 Member Function Description

- void **set_node_a** (*NodePath* node)

Set the path to the A node for the joint. Must be of type *PhysicsBody2D*.

- *NodePath* **get_node_a** ( ) const

Return the path to the A node for the joint.

- void **set_node_b** (*NodePath* node)

Set the path to the B node for the joint. Must be of type *PhysicsBody2D*.

- *NodePath* **get_node_b** ( ) const

Return the path to the B node for the joint.

- void **set_bias** (*float* bias)
- *float* **get_bias** ( ) const
- void **set_exclude_nodes_from_collision** (*bool* enable)
- *bool* **get_exclude_nodes_from_collision** ( ) const

9.135 KinematicBody

Inherits: *PhysicsBody* < *CollisionObject* < *Spatial* < *Node* < *Object*

Category: Core

9.135.1 Brief Description

9.135.2 Member Functions

Vector3	*move* (*Vector3* rel_vec)
Vector3	*move_to* (*Vector3* position)
bool	*can_teleport_to* (*Vector3* position)
bool	*is_colliding* ( ) const
Vector3	*get_collision_pos* ( ) const
Vector3	*get_collision_normal* ( ) const
Vector3	*get_collider_velocity* ( ) const
Object	*get_collider* ( ) const
int	*get_collider_shape* ( ) const
void	*set_collide_with_static_bodies* (*bool* enable)
bool	*can_collide_with_static_bodies* ( ) const
void	*set_collide_with_kinematic_bodies* (*bool* enable)
bool	*can_collide_with_kinematic_bodies* ( ) const
void	*set_collide_with_rigid_bodies* (*bool* enable)
bool	*can_collide_with_rigid_bodies* ( ) const
void	*set_collide_with_character_bodies* (*bool* enable)
bool	*can_collide_with_character_bodies* ( ) const
void	*set_collision_margin* (*float* pixels)
float	*get_collision_margin* ( ) const

9.135.3 Member Function Description

- *Vector3* **move** (*Vector3* rel_vec)
- *Vector3* **move_to** (*Vector3* position)
- *bool* **can_teleport_to** (*Vector3* position)

Returns whether the KinematicBody can be teleported to the destination given as an argument, checking all collision shapes of the body against potential colliders at the destination.

- *bool* **is_colliding** ( ) const
- *Vector3* **get_collision_pos** ( ) const
- *Vector3* **get_collision_normal** ( ) const
- *Vector3* **get_collider_velocity** ( ) const
- *Object* **get_collider** ( ) const
- *int* **get_collider_shape** ( ) const
- void **set_collide_with_static_bodies** (*bool* enable)
- *bool* **can_collide_with_static_bodies** ( ) const
- void **set_collide_with_kinematic_bodies** (*bool* enable)
- *bool* **can_collide_with_kinematic_bodies** ( ) const
- void **set_collide_with_rigid_bodies** (*bool* enable)
- *bool* **can_collide_with_rigid_bodies** ( ) const
- void **set_collide_with_character_bodies** (*bool* enable)

- *bool* **can_collide_with_character_bodies** ( ) const
- void **set_collision_margin** (*float* pixels)
- *float* **get_collision_margin** ( ) const

9.136 KinematicBody2D

Inherits: *PhysicsBody2D* < *CollisionObject2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.136.1 Brief Description

Kinematic body 2D node.

9.136.2 Member Functions

Vector2	*move* (*Vector2* rel_vec)
Vector2	*move_to* (*Vector2* position)
bool	*test_move* (*Vector2* rel_vec)
Vector2	*get_travel* ( ) const
void	*revert_motion* ( )
bool	*is_colliding* ( ) const
Vector2	*get_collision_pos* ( ) const
Vector2	*get_collision_normal* ( ) const
Vector2	*get_collider_velocity* ( ) const
Object	*get_collider* ( ) const
int	*get_collider_shape* ( ) const
Variant	*get_collider_metadata* ( ) const
void	*set_collision_margin* (*float* pixels)
float	*get_collision_margin* ( ) const

9.136.3 Description

Kinematic bodies are special types of bodies that are meant to be user-controlled. They are not affected by physics at all (to other types of bodies, such a character or a rigid body, these are the same as a static body). They have however, two main uses:

Simulated Motion: When these bodies are moved manually, either from code or from an AnimationPlayer (with process mode set to fixed), the physics will automatically compute an estimate of their linear and angular velocity. This makes them very useful for moving platforms or other AnimationPlayer-controlled objects (like a door, a bridge that opens, etc).

Kinematic Characters: KinematicBody2D also has an api for moving objects (the *move* method) while performing collision tests. This makes them really useful to implement characters that collide against a world, but that don't require advanced physics.

9.136.4 Member Function Description

- *Vector2* **move** (*Vector2* rel_vec)

Move the body in the given direction, stopping if there is an obstacle.

- *Vector2* **move_to** (*Vector2* position)

Move the body to the given position. This is not a teleport, and the body will stop if there is an obstacle.

- *bool* **test_move** (*Vector2* rel_vec)

Return true if there would be a collision if the body moved in the given direction.

- *Vector2* **get_travel** ( ) const

Return the last movement done by the body.

- void **revert_motion** ( )

Undo the last movement done by the body.

- *bool* **is_colliding** ( ) const

Return whether the body is colliding with another.

- *Vector2* **get_collision_pos** ( ) const

Return the point in space where the body is touching another. If there is no collision, this method will return (0,0), so collisions must be checked first with *is_colliding*.

- *Vector2* **get_collision_normal** ( ) const

Return the normal of the surface the body collided with. This is useful to implement sliding along a surface.

- *Vector2* **get_collider_velocity** ( ) const

Return the velocity of the body that collided with this one.

- *Object* **get_collider** ( ) const

Return the body that collided with this one.

- *int* **get_collider_shape** ( ) const

Return the shape index from the body that collided with this one. If there is no collision, this method will return 0, so collisions must be checked first with *is_colliding*.

- Variant **get_collider_metadata** ( ) const

Return the metadata of the shape that collided with this body. If there is no collision, it will return 0, so collisions must be checked first with *is_colliding*. Additionally, this metadata can not be set with *Object.set_meta*, it must be set with *Physics2DServer.body_set_shape_metadata*.

- void **set_collision_margin** (*float* pixels)

Set the collision margin for this object. A collision margin is an amount (in pixels) that all shapes will grow when computing collisions, to account for numerical imprecision.

- *float* **get_collision_margin** ( ) const

Return the collision margin for this object.

9.137 Label

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.137.1 Brief Description

Control that displays formatted text.

9.137.2 Member Functions

void	*set_align* (*int* align)
int	*get_align* ( ) const
void	*set_valign* (*int* valign)
int	*get_valign* ( ) const
void	*set_text* (*String* text)
String	*get_text* ( ) const
void	*set_autowrap* (*bool* enable)
bool	*has_autowrap* ( ) const
void	*set_clip_text* (*bool* enable)
bool	*is_clipping_text* ( ) const
void	*set_uppercase* (*bool* enable)
bool	*is_uppercase* ( ) const
int	*get_line_height* ( ) const
int	*get_line_count* ( ) const
int	*get_total_character_count* ( ) const
void	*set_visible_characters* (*int* amount)
int	*get_visible_characters* ( ) const
void	*set_percent_visible* (*float* percent_visible)
float	*get_percent_visible* ( ) const
void	*set_lines_skipped* (*int* lines_skipped)
int	*get_lines_skipped* ( ) const
void	*set_max_lines_visible* (*int* lines_visible)
int	*get_max_lines_visible* ( ) const

9.137.3 Numeric Constants

- **ALIGN_LEFT = 0** — Align rows to the left (default).
- **ALIGN_CENTER = 1** — Align rows centered.
- **ALIGN_RIGHT = 2** — Align rows to the right (default).
- **ALIGN_FILL = 3** — Expand row whitespaces to fit the width.
- **VALIGN_TOP = 0** — Align the whole text to the top.
- **VALIGN_CENTER = 1** — Align the whole text to the center.
- **VALIGN_BOTTOM = 2** — Align the whole text to the bottom.
- **VALIGN_FILL = 3** — Align the whole text by spreading the rows.

9.137.4 Description

Label is a control that displays formatted text, optionally autowrapping it to the *Control* area. It inherits from range to be able to scroll wrapped text vertically.

9.137.5 Member Function Description

- void **set_align** (*int* align)

Sets the alignment mode to any of the ALIGN_* enumeration values.

- *int* **get_align** ( ) const

Return the alignment mode (any of the ALIGN_* enumeration values).

- void **set_valign** (*int* valign)

Sets the vertical alignment mode to any of the VALIGN_* enumeration values.

- *int* **get_valign** ( ) const

Return the vertical alignment mode (any of the VALIGN_* enumeration values).

- void **set_text** (*String* text)

Set the label text. Text can contain newlines.

- *String* **get_text** ( ) const

Return the label text. Text can contain newlines.

- void **set_autowrap** (*bool* enable)

Set *autowrap* mode. When enabled, autowrap will fit text to the control width, breaking sentences when they exceed the available horizontal space. When disabled, the label minimum width becomes the width of the longest row, and the minimum height large enough to fit all rows.

- *bool* **has_autowrap** ( ) const

Return the state of the *autowrap* mode (see *set_autowrap*).

- void **set_clip_text** (*bool* enable)

Cuts off the rest of the text if it is too wide.

- *bool* **is_clipping_text** ( ) const

Return true if text would be cut off if it is too wide.

- void **set_uppercase** (*bool* enable)

Display text in all capitals.

- *bool* **is_uppercase** ( ) const

Return true if text is displayed in all capitals.

- *int* **get_line_height** ( ) const

Return the height of a line.

- *int* **get_line_count** ( ) const

Return the amount of lines.

- *int* **get_total_character_count** ( ) const

Return the total length of the text.

- void **set_visible_characters** (*int* amount)

Restricts the number of characters to display. Set to -1 to disable.

- *int* **get_visible_characters** ( ) const

Return the restricted number of characters to display. Returns -1 if unrestricted.

- void **set_percent_visible** (*float* percent_visible)

Restricts the number of characters to display (as a percentage of the total text).

- *float* **get_percent_visible** ( ) const

Return the restricted number of characters to display (as a percentage of the total text).

- void **set_lines_skipped** (*int* lines_skipped)

Sets the number of lines to skip before displaying. Useful for scrolling text.

- *int* **get_lines_skipped** ( ) const

Return the the number of lines to skipped before displaying.

- void **set_max_lines_visible** (*int* lines_visible)

Restricts the number of lines to display. Set to -1 to disable.

- *int* **get_max_lines_visible** ( ) const

Return the restricted number of lines to display. Returns -1 if unrestricted.

9.138 LargeTexture

Inherits: *Texture* < *Resource* < *Reference* < *Object*

Category: Core

9.138.1 Brief Description

9.138.2 Member Functions

int	*add_piece* (*Vector2* ofs, *Texture* texture)
void	*set_piece_offset* (*int* idx, *Vector2* ofs)
void	*set_piece_texture* (*int* idx, *Texture* texture)
void	*set_size* (*Vector2* size)
void	*clear* ( )
int	*get_piece_count* ( ) const
Vector2	*get_piece_offset* (*int* idx) const
Texture	*get_piece_texture* (*int* idx) const

9.138.3 Member Function Description

- *int* **add_piece** (*Vector2* ofs, *Texture* texture)
- void **set_piece_offset** (*int* idx, *Vector2* ofs)

- void **set_piece_texture** (*int* idx, *Texture* texture)
- void **set_size** (*Vector2* size)
- void **clear** ( )
- *int* **get_piece_count** ( ) const
- *Vector2* **get_piece_offset** (*int* idx) const
- *Texture* **get_piece_texture** (*int* idx) const

9.139 Light

Inherits: *VisualInstance* < *Spatial* < *Node* < *Object*

Inherited By: *SpotLight*, *OmniLight*, *DirectionalLight*

Category: Core

9.139.1 Brief Description

Provides a base class for different kinds of light nodes.

9.139.2 Member Functions

void	*set_parameter* (*int* variable, *float* value)
float	*get_parameter* (*int* variable) const
void	*set_color* (*int* color, *Color* value)
Color	*get_color* (*int* color) const
void	*set_project_shadows* (*bool* enable)
bool	*has_project_shadows* ( ) const
void	*set_projector* (*Texture* projector)
Texture	*get_projector* ( ) const
void	*set_operator* (*int* operator)
int	*get_operator* ( ) const
void	*set_bake_mode* (*int* bake_mode)
int	*get_bake_mode* ( ) const
void	*set_enabled* (*bool* enabled)
bool	*is_enabled* ( ) const
void	*set_editor_only* (*bool* editor_only)
bool	*is_editor_only* ( ) const

9.139.3 Numeric Constants

- **PARAM_RADIUS = 2**
- **PARAM_ENERGY = 3**
- **PARAM_ATTENUATION = 4**
- **PARAM_SPOT_ANGLE = 1**
- **PARAM_SPOT_ATTENUATION = 0**

- **PARAM_SHADOW_DARKENING = 5**
- **PARAM_SHADOW_Z_OFFSET = 6**
- **COLOR_DIFFUSE = 0**
- **COLOR_SPECULAR = 1**
- **BAKE_MODE_DISABLED = 0**
- **BAKE_MODE_INDIRECT = 1**
- **BAKE_MODE_INDIRECT_AND_SHADOWS = 2**
- **BAKE_MODE_FULL = 3**

9.139.4 Description

Light is the abstract base class for light nodes, so it shouldn't be used directly (It can't be instanced). Other types of light nodes inherit from it. Light contains the common variables and parameters used for lighting.

9.139.5 Member Function Description

- void **set_parameter** (*int* variable, *float* value)
- *float* **get_parameter** (*int* variable) const
- void **set_color** (*int* color, *Color* value)
- *Color* **get_color** (*int* color) const
- void **set_project_shadows** (*bool* enable)
- *bool* **has_project_shadows** ( ) const
- void **set_projector** (*Texture* projector)
- *Texture* **get_projector** ( ) const
- void **set_operator** (*int* operator)
- *int* **get_operator** ( ) const
- void **set_bake_mode** (*int* bake_mode)
- *int* **get_bake_mode** ( ) const
- void **set_enabled** (*bool* enabled)
- *bool* **is_enabled** ( ) const
- void **set_editor_only** (*bool* editor_only)
- *bool* **is_editor_only** ( ) const

9.140 Light2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.140.1 Brief Description

Node that casts light in a 2D environment.

9.140.2 Member Functions

void	*set_enabled* (*bool* enabled)
bool	*is_enabled* ( ) const
void	*set_texture* (*Object* texture)
Object	*get_texture* ( ) const
void	*set_texture_offset* (*Vector2* texture_offset)
Vector2	*get_texture_offset* ( ) const
void	*set_color* (*Color* color)
Color	*get_color* ( ) const
void	*set_height* (*float* height)
float	*get_height* ( ) const
void	*set_energy* (*float* energy)
float	*get_energy* ( ) const
void	*set_texture_scale* (*float* texture_scale)
float	*get_texture_scale* ( ) const
void	*set_z_range_min* (*int* z)
int	*get_z_range_min* ( ) const
void	*set_z_range_max* (*int* z)
int	*get_z_range_max* ( ) const
void	*set_layer_range_min* (*int* layer)
int	*get_layer_range_min* ( ) const
void	*set_layer_range_max* (*int* layer)
int	*get_layer_range_max* ( ) const
void	*set_item_mask* (*int* item_mask)
int	*get_item_mask* ( ) const
void	*set_item_shadow_mask* (*int* item_shadow_mask)
int	*get_item_shadow_mask* ( ) const
void	*set_mode* (*int* mode)
int	*get_mode* ( ) const
void	*set_shadow_enabled* (*bool* enabled)
bool	*is_shadow_enabled* ( ) const
void	*set_shadow_buffer_size* (*int* size)
int	*get_shadow_buffer_size* ( ) const
void	*set_shadow_esm_multiplier* (*float* multiplier)
float	*get_shadow_esm_multiplier* ( ) const
void	*set_shadow_color* (*Color* shadow_color)
Color	*get_shadow_color* ( ) const

9.140.3 Numeric Constants

- **MODE_ADD = 0** — Adds the value of pixels corresponding to the Light2D to the values of pixels under it. This is the common behaviour of a light.
- **MODE_SUB = 1** — Substract the value of pixels corresponding to the Light2D to the values of pixels under it, resulting in inversed light effect.

- **MODE_MIX = 2** — Mix the value of pixels corresponding to the Light2D to the values of pixels under it by linear interpolation.
- **MODE_MASK = 3** — The light texture of the Light2D is used as a mask, hiding or revealing parts of the screen underneath depending on the value of each pixel of the light (mask) texture.

9.140.4 Description

Node that casts light in a 2D environment. Light is defined by a (usually grayscale) texture, a color, an energy value, a mode (see constants), and various other parameters (range and shadows-related). Note that Light2D can be used as a mask.

9.140.5 Member Function Description

- void **set_enabled** (*bool* enabled)

Switches the Light2D on or off, depending on the 'enabled' parameter.

- *bool* **is_enabled** ( ) const

Return true if the Light2D is enabled, false if it is not.

- void **set_texture** (*Object* texture)

Set the texture of the Light2D.

- *Object* **get_texture** ( ) const

Return the texture of the Light2D.

- void **set_texture_offset** (*Vector2* texture_offset)

Set the offset of the light texture.

- *Vector2* **get_texture_offset** ( ) const

Return the offset of the light texture.

- void **set_color** (*Color* color)

Set the color of the Light2D.

- *Color* **get_color** ( ) const

Return the color of the Light2D.

- void **set_height** (*float* height)

Set the height of the Light2D. Used with 2D normalmapping.

- *float* **get_height** ( ) const

Return the height of the Light2D. Used with 2D normalmapping.

- void **set_energy** (*float* energy)

Set the energy value of the Light2D. The bigger the value, the stronger the light.

- *float* **get_energy** ( ) const

Return the energy value of the Light2D.

- void **set_texture_scale** (*float* texture_scale)

Set the scale value of the light texture.

- *float* **get_texture_scale** ( ) const

Return the scale value of the light texture.

- void **set_z_range_min** (*int* z)

Set the minimum Z value that objects of the scene have to be in order to be affected by the Light2D.

- *int* **get_z_range_min** ( ) const

Get the minimum Z value that objects of the scene have to be in order to be affected by the Light2D.

- void **set_z_range_max** (*int* z)

Set the maximum Z value that objects of the scene can be in order to be affected by the Light2D.

- *int* **get_z_range_max** ( ) const

Get the maximum Z value that objects of the scene can be in order to be affected by the Light2D.

- void **set_layer_range_min** (*int* layer)

Set the minimum layer value of objects of the scene that are affected by the Light2D.

- *int* **get_layer_range_min** ( ) const

Get the minimum layer value of objects of the scene that are affected by the Light2D.

- void **set_layer_range_max** (*int* layer)

Set the maximum layer value of objects of the scene that are affected by the Light2D.

- *int* **get_layer_range_max** ( ) const

Set the maximum layer value of objects of the scene that are affected by the Light2D.

- void **set_item_mask** (*int* item_mask)

Set the item mask of the Light2D to 'item_mask' value.

- *int* **get_item_mask** ( ) const

Return the item mask of the Light2D.

- void **set_item_shadow_mask** (*int* item_shadow_mask)

Set the item shadow mask to 'item_shadow_mask' value.

- *int* **get_item_shadow_mask** ( ) const

Return the item shadow mask of the Light2D.

- void **set_mode** (*int* mode)

Set the behaviour mode of the Light2D. Use constants defined in the constants section.

- *int* **get_mode** ( ) const

Return the current mode set to the Light2D.

- void **set_shadow_enabled** (*bool* enabled)

Enable or disable shadows casting from this Light2D according to the 'enabled' parameter.

- *bool* **is_shadow_enabled** ( ) const

Return true if shadow casting is enabled for this Light2D, else return false.

- void **set_shadow_buffer_size** (*int* size)

Set the shadow buffer size.

- *int* **get_shadow_buffer_size** ( ) const

Return the shadow buffer size.

- void **set_shadow_esm_multiplier** (*float* multiplier)

Set the Exponential Shadow Multiplier (ESM) value of the Light2D.

- *float* **get_shadow_esm_multiplier** ( ) const

Return the Exponential Shadow Multiplier (ESM) value of the Light2D.

- void **set_shadow_color** (*Color* shadow_color)

Set the color of casted shadows for this Light2D.

- *Color* **get_shadow_color** ( ) const

Return the color of casted shadows for this Light2D.

9.141 LightOccluder2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.141.1 Brief Description

Occludes light cast by a Light2D, thus casting shadows.

9.141.2 Member Functions

void	*set_occluder_polygon* (*OccluderPolygon2D* polygon)
OccluderPolygon2D	*get_occluder_polygon* ( ) const
void	*set_occluder_light_mask* (*int* mask)
int	*get_occluder_light_mask* ( ) const

9.141.3 Description

Occludes light cast by a Light2D, thus casting shadows. The LightOccluder2D must be provided with a shape (see OccluderPolygon2D) that allows the shadow to be computed. This shape affects the resulting shadow, while the shape of the representating asset shadowed does not actually affect shadows.

9.141.4 Member Function Description

- void **set_occluder_polygon** (*OccluderPolygon2D* polygon)

Set the OccluderPolygon2D that defines the LightOccluder2D.

- *OccluderPolygon2D* **get_occluder_polygon** ( ) const

Return the OccluderPolygon2D that defines the LightOccluder2D.

- void **set_occluder_light_mask** (*int* mask)

Set the LightOccluder2D light mask. The LightOccluder2D will cast shadows only from Light2Ds that belong to the same light mask(s).

- *int* **get_occluder_light_mask** ( ) const

Return the light mask of the LightOccluder2D.

9.142 LineEdit

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.142.1 Brief Description

Control that provides single line string editing.

9.142.2 Member Functions

void	*set_align* (*int* align)
int	*get_align* ( ) const
void	*clear* ( )
void	*select_all* ( )
void	*set_text* (*String* text)
String	*get_text* ( ) const
void	*set_cursor_pos* (*int* pos)
int	*get_cursor_pos* ( ) const
void	*set_max_length* (*int* chars)
int	*get_max_length* ( ) const
void	*append_at_cursor* (*String* text)
void	*set_editable* (*bool* enabled)
bool	*is_editable* ( ) const
void	*set_secret* (*bool* enabled)
bool	*is_secret* ( ) const
void	*select* (*int* from=0, *int* to=-1)

9.142.3 Signals

- **text_entered** (*String* text)
- **text_changed** (*String* text)

9.142.4 Numeric Constants

- **ALIGN_LEFT = 0**
- **ALIGN_CENTER = 1**
- **ALIGN_RIGHT = 2**
- **ALIGN_FILL = 3**

9.142.5 Description

LineEdit provides a single line string editor, used for text fields.

9.142.6 Member Function Description

- void **set_align** (*int* align)
- *int* **get_align** ( ) const
- void **clear** ( )

Clear the *LineEdit* text.

- void **select_all** ( )

Select the whole string.

- void **set_text** (*String* text)

Set the text in the *LineEdit*, clearing the existing one and the selection.

- *String* **get_text** ( ) const

Return the text in the *LineEdit*.

- void **set_cursor_pos** (*int* pos)

Set the cursor position inside the *LineEdit*, causing it to scroll if needed.

- *int* **get_cursor_pos** ( ) const

Return the cursor position inside the *LineEdit*.

- void **set_max_length** (*int* chars)

Set the maximum amount of characters the *LineEdit* can edit, and cropping existing text in case it exceeds that limit. Setting 0 removes the limit.

- *int* **get_max_length** ( ) const

Return the maximum amount of characters the *LineEdit* can edit. If 0 is returned, no limit exists.

- void **append_at_cursor** (*String* text)

Append text at cursor, scrolling the *LineEdit* when needed.

- void **set_editable** (*bool* enabled)

Set the *editable* status of the *LineEdit*. When disabled, existing text can't be modified and new text can't be added.

- *bool* **is_editable** ( ) const

Return the *editable* status of the *LineEdit* (see *set_editable*).

- void **set_secret** (*bool* enabled)

Set the *secret* status of the *LineEdit*. When enabled, every character is displayed as "*".

- *bool* **is_secret** ( ) const

Return the *secret* status of the *LineEdit* (see *set_secret*).

- void **select** (*int* from=0, *int* to=-1)

9.143 LineShape2D

Inherits: *Shape2D* < *Resource* < *Reference* < *Object*

Category: Core

9.143.1 Brief Description

Line shape for 2D collision objects.

9.143.2 Member Functions

void	*set_normal* (*Vector2* normal)
Vector2	*get_normal* ( ) const
void	*set_d* (*float* d)
float	*get_d* ( ) const

9.143.3 Description

Line shape for 2D collision objects. It works like a 2D plane and will not allow any body to go to the negative side. Not recommended for rigid bodies, and usually not recommended for static bodies either because it forces checks against it on every frame.

9.143.4 Member Function Description

- void **set_normal** (*Vector2* normal)

Set the line normal.

- *Vector2* **get_normal** ( ) const

Return the line normal.

- void **set_d** (*float* d)

Set the line distance from the origin.

- *float* **get_d** ( ) const

Return the line distance from the origin.

9.144 MainLoop

Inherits: *Object*

Inherited By: *SceneTree*

Category: Core

9.144.1 Brief Description

Main loop is the abstract main loop base class.

9.144.2 Member Functions

void	*_finalize* ( ) virtual
void	*_idle* (*float* delta) virtual
void	*_initialize* ( ) virtual
void	*_input_event* (*InputEvent* ev) virtual
void	*_input_text* (*String* text) virtual
void	*_iteration* (*float* delta) virtual
void	*input_event* (*InputEvent* ev)
void	*input_text* (*String* text)
void	*init* ( )
bool	*iteration* (*float* delta)
bool	*idle* (*float* delta)
void	*finish* ( )

9.144.3 Numeric Constants

- **NOTIFICATION_WM_MOUSE_ENTER = 3**
- **NOTIFICATION_WM_MOUSE_EXIT = 4**
- **NOTIFICATION_WM_FOCUS_IN = 5**
- **NOTIFICATION_WM_FOCUS_OUT = 6**
- **NOTIFICATION_WM_QUIT_REQUEST = 7**
- **NOTIFICATION_WM_UNFOCUS_REQUEST = 8**
- **NOTIFICATION_OS_MEMORY_WARNING = 9**

9.144.4 Description

Main loop is the abstract main loop base class. All other main loop classes are derived from it. Upon application start, a *MainLoop* has to be provided to OS, else the application will exit. This happens automatically (and a *SceneTree* is created), unless a main *Script* is supplied, which may or not create and return a *MainLoop*.

9.144.5 Member Function Description

- void **_finalize** ( ) virtual
- void **_idle** (*float* delta) virtual
- void **_initialize** ( ) virtual
- void **_input_event** (*InputEvent* ev) virtual
- void **_input_text** (*String* text) virtual
- void **_iteration** (*float* delta) virtual
- void **input_event** (*InputEvent* ev)
- void **input_text** (*String* text)
- void **init** ( )
- *bool* **iteration** (*float* delta)

- *bool* **idle** (*float* delta)
- void **finish** ( )

9.145 MarginContainer

Inherits: *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.145.1 Brief Description

Simple margin container.

9.145.2 Description

Simple margin container. Adds a left margin to anything contained.

9.146 Marshalls

Inherits: *Reference* < *Object*

Category: Core

9.146.1 Brief Description

9.146.2 Member Functions

String	*variant_to_base64* (var variant)
Variant	*base64_to_variant* (*String* base64_str)
String	*raw_to_base64* (*RawArray* array)
RawArray	*base64_to_raw* (*String* base64_str)
String	*utf8_to_base64* (*String* utf8_str)
String	*base64_to_utf8* (*String* base64_str)

9.146.3 Member Function Description

- *String* **variant_to_base64** (var variant)
- Variant **base64_to_variant** (*String* base64_str)
- *String* **raw_to_base64** (*RawArray* array)
- *RawArray* **base64_to_raw** (*String* base64_str)
- *String* **utf8_to_base64** (*String* utf8_str)
- *String* **base64_to_utf8** (*String* base64_str)

9.147 Material

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *ShaderMaterial*, *FixedMaterial*

Category: Core

9.147.1 Brief Description

Abstract base *Resource* for coloring and shading geometry.

9.147.2 Member Functions

void	*set_flag* (*int* flag, *bool* enable)
bool	*get_flag* (*int* flag) const
void	*set_blend_mode* (*int* mode)
int	*get_blend_mode* ( ) const
void	*set_line_width* (*float* width)
float	*get_line_width* ( ) const
void	*set_depth_draw_mode* (*int* mode)
int	*get_depth_draw_mode* ( ) const

9.147.3 Numeric Constants

- **FLAG_VISIBLE = 0** — Geometry is visible when this flag is enabled (default).
- **FLAG_DOUBLE_SIDED = 1** — Both front facing and back facing triangles are rendered when this flag is enabled.
- **FLAG_INVERT_FACES = 2** — Front facing and back facing order is swapped when this flag is enabled.
- **FLAG_UNSHADED = 3** — Shading (lighting) is disabled when this flag is enabled.
- **FLAG_ONTOP = 4**
- **FLAG_LIGHTMAP_ON_UV2 = 5**
- **FLAG_COLOR_ARRAY_SRGB = 6**
- **FLAG_MAX = 7** — Maximum amount of flags.
- **DEPTH_DRAW_ALWAYS = 0**
- **DEPTH_DRAW_OPAQUE_ONLY = 1**
- **DEPTH_DRAW_OPAQUE_PRE_PASS_ALPHA = 2**
- **DEPTH_DRAW_NEVER = 3**
- **BLEND_MODE_MIX = 0** — Use the regular alpha blending equation (source and dest colors are faded) (default).
- **BLEND_MODE_ADD = 1** — Use additive blending equation, often used for particle effects such as fire or light decals.
- **BLEND_MODE_SUB = 2** — Use subtractive blending equation, often used for some smoke effects or types of glass.

- **BLEND_MODE_MUL = 3**
- **BLEND_MODE_PREMULT_ALPHA = 4**

9.147.4 Description

Material is a base *Resource* used for coloring and shading geometry. All materials inherit from it and almost all *VisualInstance* derived nodes carry a Material. A few flags and parameters are shared between all material types and are configured here.

9.147.5 Member Function Description

- void **set_flag** (*int* flag, *bool* enable)

Set a *Material* flag, which toggles on or off a behavior when rendering. See enumeration FLAG_* for a list.

- *bool* **get_flag** (*int* flag) const

Return a *Material* flag, which toggles on or off a behavior when rendering. See enumeration FLAG_* for a list.

- void **set_blend_mode** (*int* mode)

Set blend mode for the material, which can be one of BLEND_MODE_MIX (default), BLEND_MODE_ADD, BLEND_MODE_SUB. Keep in mind that only BLEND_MODE_MIX ensures that the material *may* be opaque, any other blend mode will render with alpha blending enabled in raster-based *VisualServer* implementations.

- *int* **get_blend_mode** ( ) const

Return blend mode for the material, which can be one of BLEND_MODE_MIX (default), BLEND_MODE_ADD, BLEND_MODE_SUB. Keep in mind that only BLEND_MODE_MIX ensures that the material *may* be opaque, any other blend mode will render with alpha blending enabled in raster-based *VisualServer* implementations.

- void **set_line_width** (*float* width)

Set the line width for geometry drawn with FLAG_WIREFRAME enabled, or LINE primitives. Note that not all hardware or VisualServer backends support this (like DirectX).

- *float* **get_line_width** ( ) const

Return the line width for geometry drawn with FLAG_WIREFRAME enabled, or LINE primitives. Note that not all hardware or VisualServer backends support this (like DirectX).

- void **set_depth_draw_mode** (*int* mode)
- *int* **get_depth_draw_mode** ( ) const

9.148 MaterialShader

Inherits: *Shader* < *Resource* < *Reference* < *Object*

Category: Core

9.148.1 Brief Description

9.149 MaterialShaderGraph

Inherits: *ShaderGraph* < *Shader* < *Resource* < *Reference* < *Object*

Category: Core

9.149.1 Brief Description

9.150 Matrix3

Category: Built-In Types

9.150.1 Brief Description

3x3 matrix datatype.

9.150.2 Member Functions

float	*determinant* ( )
Vector3	*get_euler* ( )
int	*get_orthogonal_index* ( )
Vector3	*get_scale* ( )
Matrix3	*inverse* ( )
Matrix3	*orthonormalized* ( )
Matrix3	*rotated* (*Vector3* axis, *float* phi)
Matrix3	*scaled* (*Vector3* scale)
float	*tdotx* (*Vector3* with)
float	*tdoty* (*Vector3* with)
float	*tdotz* (*Vector3* with)
Matrix3	*transposed* ( )
Vector3	*xform* (*Vector3* v)
Vector3	*xform_inv* (*Vector3* v)
Matrix3	Matrix3 (*Vector3* x_axis, *Vector3* y_axis, *Vector3* z_axis)
Matrix3	Matrix3 (*Vector3* axis, *float* phi)
Matrix3	Matrix3 (*Quat* from)

9.150.3 Member Variables

- *Vector3* **x**
- *Vector3* **y**
- *Vector3* **z**

9.150.4 Description

3x3 matrix used for 3D rotation and scale. Contains 3 vector fields x,y and z. Can also be accessed as array of 3D vectors. Almost always used as orthogonal basis for a *Transform*.

9.150.5 Member Function Description

- *float* **determinant** ( )

Return the determinant of the matrix.

- *Vector3* **get_euler** ( )

Return euler angles from the matrix.

- *int* **get_orthogonal_index** ( )
- *Vector3* **get_scale** ( )
- *Matrix3* **inverse** ( )

Return the affine inverse of the matrix.

- *Matrix3* **orthonormalized** ( )

Return the orthonormalized version of the matrix (useful to call from time to time to avoid rounding error).

- *Matrix3* **rotated** (*Vector3* axis, *float* phi)

Return the rotated version of the matrix, by a given axis and angle.

- *Matrix3* **scaled** (*Vector3* scale)

Return the scaled version of the matrix, by a 3D scale.

- *float* **tdotx** (*Vector3* with)

Transposed dot product with the x axis of the matrix.

- *float* **tdoty** (*Vector3* with)

Transposed dot product with the y axis of the matrix.

- *float* **tdotz** (*Vector3* with)

Transposed dot product with the z axis of the matrix.

- *Matrix3* **transposed** ( )

Return the transposed version of the matrix.

- *Vector3* **xform** (*Vector3* v)

Return a vector transformed by the matrix and return it.

- *Vector3* **xform_inv** (*Vector3* v)

Return a vector transformed by the transposed matrix and return it.

- *Matrix3* **Matrix3** (*Vector3* x_axis, *Vector3* y_axis, *Vector3* z_axis)

Create a matrix from 3 axis vectors.

- *Matrix3* **Matrix3** (*Vector3* axis, *float* phi)

Create a matrix from an axis vector and an angle.

- *Matrix3* **Matrix3** (*Quat* from)

Create a matrix from a quaternion.

9.151 Matrix32

Category: Built-In Types

9.151.1 Brief Description

3x2 Matrix for 2D transforms.

9.151.2 Member Functions

Matrix32	*affine_inverse* ( )
Matrix32	*basis_xform* (var v)
Matrix32	*basis_xform_inv* (var v)
Vector2	*get_origin* ( )
float	*get_rotation* ( )
Vector2	*get_scale* ( )
Matrix32	*interpolate_with* (*Matrix32* m, *float* c)
Matrix32	*inverse* ( )
Matrix32	*orthonormalized* ( )
Matrix32	*rotated* (*float* phi)
Matrix32	*scaled* (*Vector2* scale)
Matrix32	*translated* (*Vector2* offset)
Matrix32	*xform* (var v)
Matrix32	*xform_inv* (var v)
Matrix32	Matrix32 (*float* rot, *Vector2* pos)
Matrix32	Matrix32 (*Vector2* x_axis, *Vector2* y_axis, *Vector2* origin)
Matrix32	Matrix32 (*Transform* from)

9.151.3 Member Variables

- *Vector2* **x**
- *Vector2* **y**
- *Vector2* **o**

9.151.4 Description

3x2 Matrix for 2D transforms.

9.151.5 Member Function Description

- *Matrix32* **affine_inverse** ( )
- *Matrix32* **basis_xform** (var v)
- *Matrix32* **basis_xform_inv** (var v)

- *Vector2* **get_origin** ( )
- *float* **get_rotation** ( )
- *Vector2* **get_scale** ( )
- *Matrix32* **interpolate_with** (*Matrix32* m, *float* c)
- *Matrix32* **inverse** ( )
- *Matrix32* **orthonormalized** ( )
- *Matrix32* **rotated** (*float* phi)
- *Matrix32* **scaled** (*Vector2* scale)
- *Matrix32* **translated** (*Vector2* offset)
- *Matrix32* **xform** (var v)
- *Matrix32* **xform_inv** (var v)
- *Matrix32* **Matrix32** (*float* rot, *Vector2* pos)
- *Matrix32* **Matrix32** (*Vector2* x_axis, *Vector2* y_axis, *Vector2* origin)
- *Matrix32* **Matrix32** (*Transform* from)

9.152 MenuButton

Inherits: *Button* < *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.152.1 Brief Description

Special button that brings up a *PopupMenu* when clicked.

9.152.2 Member Functions

PopupMenu	*get_popup* ( )

9.152.3 Signals

- **about_to_show** ( )

9.152.4 Description

Special button that brings up a *PopupMenu* when clicked. That's pretty much all it does, as it's just a helper class when building GUIs.

9.152.5 Member Function Description

- *PopupMenu* **get_popup** ( )

Return the *PopupMenu* contained in this button.

9.153 Mesh

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.153.1 Brief Description

A *Resource* that contains vertex-array based geometry.

9.153.2 Member Functions

void	*add_morph_target* (*String* name)
int	*get_morph_target_count* ( ) const
String	*get_morph_target_name* (*int* index) const
void	*clear_morph_targets* ( )
void	*set_morph_target_mode* (*int* mode)
int	*get_morph_target_mode* ( ) const
void	*add_surface* (*int* primitive, *Array* arrays, *Array* morph_arrays=Array(), *bool* alphasort=false)
int	*get_surface_count* ( ) const
void	*surface_remove* (*int* surf_idx)
int	*surface_get_array_len* (*int* surf_idx) const
int	*surface_get_array_index_len* (*int* surf_idx) const
int	*surface_get_format* (*int* surf_idx) const
int	*surface_get_primitive_type* (*int* surf_idx) const
void	*surface_set_material* (*int* surf_idx, *Material* material)
Material	*surface_get_material* (*int* surf_idx) const
void	*surface_set_name* (*int* surf_idx, *String* name)
String	*surface_get_name* (*int* surf_idx) const
void	*center_geometry* ( )
void	*regen_normalmaps* ( )
void	*set_custom_aabb* (*AABB* aabb)
AABB	*get_custom_aabb* ( ) const

9.153.3 Numeric Constants

- **NO_INDEX_ARRAY = -1** — Default value used for index_array_len when no indices are present.
- **ARRAY_WEIGHTS_SIZE = 4** — Amount of weights/bone indices per vertex (always 4).
- **ARRAY_VERTEX = 0** — Vertex array (array of *Vector3* vertices).
- **ARRAY_NORMAL = 1** — Normal array (array of *Vector3* normals).
- **ARRAY_TANGENT = 2** — Tangent array, array of groups of 4 floats. first 3 floats determine the tangent, and the last the binormal direction as -1 or 1.
- **ARRAY_COLOR = 3** — Vertex array (array of *Color* colors).
- **ARRAY_TEX_UV = 4** — UV array (array of *Vector3* UVs or float array of groups of 2 floats (u,v)).
- **ARRAY_TEX_UV2 = 5** — Second UV array (array of *Vector3* UVs or float array of groups of 2 floats (u,v)).
- **ARRAY_BONES = 6** — Array of bone indices, as a float array. Each element in groups of 4 floats.

- **ARRAY_WEIGHTS = 7** — Array of bone weights, as a float array. Each element in groups of 4 floats.
- **ARRAY_INDEX = 8** — Array of integers, used as indices referencing vertices. No index can be beyond the vertex array size.
- **ARRAY_FORMAT_VERTEX = 1** — Array format will include vertices (mandatory).
- **ARRAY_FORMAT_NORMAL = 2** — Array format will include normals
- **ARRAY_FORMAT_TANGENT = 4** — Array format will include tangents
- **ARRAY_FORMAT_COLOR = 8** — Array format will include a color array.
- **ARRAY_FORMAT_TEX_UV = 16** — Array format will include UVs.
- **ARRAY_FORMAT_TEX_UV2 = 32** — Array format will include another set of UVs.
- **ARRAY_FORMAT_BONES = 64** — Array format will include bone indices.
- **ARRAY_FORMAT_WEIGHTS = 128** — Array format will include bone weights.
- **ARRAY_FORMAT_INDEX = 256** — Index array will be used.
- **PRIMITIVE_POINTS = 0** — Render array as points (one vertex equals one point).
- **PRIMITIVE_LINES = 1** — Render array as lines (every two vertices a line is created).
- **PRIMITIVE_LINE_STRIP = 2** — Render array as line strip.
- **PRIMITIVE_LINE_LOOP = 3** — Render array as line loop (like line strip, but closed).
- **PRIMITIVE_TRIANGLES = 4** — Render array as triangles (every three vertices a triangle is created).
- **PRIMITIVE_TRIANGLE_STRIP = 5** — Render array as triangle strips.
- **PRIMITIVE_TRIANGLE_FAN = 6** — Render array as triangle fans.

9.153.4 Description

Mesh is a type of *Resource* that contains vertex-array based geometry, divided in *surfaces*. Each surface contains a completely separate array and a material used to draw it. Design wise, a mesh with multiple surfaces is preferred to a single surface, because objects created in 3D editing software commonly contain multiple materials.

9.153.5 Member Function Description

- void **add_morph_target** (*String* name)
- *int* **get_morph_target_count** ( ) const
- *String* **get_morph_target_name** (*int* index) const
- void **clear_morph_targets** ( )
- void **set_morph_target_mode** (*int* mode)
- *int* **get_morph_target_mode** ( ) const
- void **add_surface** (*int* primitive, *Array* arrays, *Array* morph_arrays=Array(), *bool* alphasort=false)

Create a new surface (*get_surface_count* that will become surf_idx for this.

Surfaces are created to be rendered using a "primitive", which may be PRIMITIVE_POINTS, PRIMITIVE_LINES, PRIMITIVE_LINE_STRIP, PRIMITIVE_LINE_LOOP, PRIMITIVE_TRIANGLES, PRIMITIVE_TRIANGLE_STRIP, PRIMITIVE_TRIANGLE_FAN. (As a note, when using indices, it is recommended to only use just points, lines or triangles).

The format of a surface determines which arrays it will allocate and hold, so "format" is a combination of ARRAY_FORMAT_* mask constants ORed together. ARRAY_FORMAT_VERTEX must be always present. "array_len" determines the amount of vertices in the array (not primitives!). if ARRAY_FORMAT_INDEX is in the format mask, then it means that an index array will be allocated and "index_array_len" must be passed.

- *int* **get_surface_count** **(** **)** const

Return the amount of surfaces that the *Mesh* holds.

- void **surface_remove** **(** *int* surf_idx **)**

Remove a surface at position surf_idx, shifting greater surfaces one surf_idx slot down.

- *int* **surface_get_array_len** **(** *int* surf_idx **)** const

Return the length in vertices of the vertex array in the requested surface (see *add_surface*).

- *int* **surface_get_array_index_len** **(** *int* surf_idx **)** const

Return the length in indices of the index array in the requested surface (see *add_surface*).

- *int* **surface_get_format** **(** *int* surf_idx **)** const

Return the format mask of the requested surface (see *add_surface*).

- *int* **surface_get_primitive_type** **(** *int* surf_idx **)** const

Return the primitive type of the requested surface (see *add_surface*).

- void **surface_set_material** **(** *int* surf_idx, *Material* material **)**

Set a *Material* for a given surface. Surface will be rendered using this material.

- *Material* **surface_get_material** **(** *int* surf_idx **)** const

Return a *Material* in a given surface. Surface is rendered using this material.

- void **surface_set_name** **(** *int* surf_idx, *String* name **)**
- *String* **surface_get_name** **(** *int* surf_idx **)** const
- void **center_geometry** **(** **)**
- void **regen_normalmaps** **(** **)**
- void **set_custom_aabb** **(** *AABB* aabb **)**
- *AABB* **get_custom_aabb** **(** **)** const

9.154 MeshDataTool

Inherits: *Reference* **<** *Object*

Category: Core

9.154.1 Brief Description

9.154.2 Member Functions

void	*clear* **(** **)**
int	*create_from_surface* **(** *Object* mesh, *int* surface **)**

Continúa en la página siguiente

Tabla 9.14 – proviene de la página anterior

int	*commit_to_surface* (*Object* mesh)
int	*get_format* ( ) const
int	*get_vertex_count* ( ) const
int	*get_edge_count* ( ) const
int	*get_face_count* ( ) const
void	*set_vertex* (*int* idx, *Vector3* vertex)
Vector3	*get_vertex* (*int* idx) const
void	*set_vertex_normal* (*int* idx, *Vector3* normal)
Vector3	*get_vertex_normal* (*int* idx) const
void	*set_vertex_tangent* (*int* idx, *Plane* tangent)
Plane	*get_vertex_tangent* (*int* idx) const
void	*set_vertex_uv* (*int* idx, *Vector2* uv)
Vector2	*get_vertex_uv* (*int* idx) const
void	*set_vertex_uv2* (*int* idx, *Vector2* uv2)
Vector2	*get_vertex_uv2* (*int* idx) const
void	*set_vertex_color* (*int* idx, *Color* color)
Color	*get_vertex_color* (*int* idx) const
void	*set_vertex_bones* (*int* idx, *IntArray* bones)
IntArray	*get_vertex_bones* (*int* idx) const
void	*set_vertex_weights* (*int* idx, *RealArray* weights)
RealArray	*get_vertex_weights* (*int* idx) const
void	*set_vertex_meta* (*int* idx, var meta)
void	*get_vertex_meta* (*int* idx) const
IntArray	*get_vertex_edges* (*int* idx) const
IntArray	*get_vertex_faces* (*int* idx) const
int	*get_edge_vertex* (*int* idx, *int* vertex) const
IntArray	*get_edge_faces* (*int* idx) const
void	*set_edge_meta* (*int* idx, var meta)
void	*get_edge_meta* (*int* idx) const
int	*get_face_vertex* (*int* idx, *int* vertex) const
int	*get_face_edge* (*int* idx, *int* edge) const
void	*set_face_meta* (*int* idx, var meta)
void	*get_face_meta* (*int* idx) const
Vector3	*get_face_normal* (*int* idx) const
void	*set_material* (*Material* material)
Object	*get_material* ( ) const

9.154.3 Member Function Description

- void **clear** ( )
- *int* **create_from_surface** (*Object* mesh, *int* surface)
- *int* **commit_to_surface** (*Object* mesh)
- *int* **get_format** ( ) const
- *int* **get_vertex_count** ( ) const
- *int* **get_edge_count** ( ) const
- *int* **get_face_count** ( ) const
- void **set_vertex** (*int* idx, *Vector3* vertex)

- *Vector3* **get_vertex** (*int* idx) const
- void **set_vertex_normal** (*int* idx, *Vector3* normal)
- *Vector3* **get_vertex_normal** (*int* idx) const
- void **set_vertex_tangent** (*int* idx, *Plane* tangent)
- *Plane* **get_vertex_tangent** (*int* idx) const
- void **set_vertex_uv** (*int* idx, *Vector2* uv)
- *Vector2* **get_vertex_uv** (*int* idx) const
- void **set_vertex_uv2** (*int* idx, *Vector2* uv2)
- *Vector2* **get_vertex_uv2** (*int* idx) const
- void **set_vertex_color** (*int* idx, *Color* color)
- *Color* **get_vertex_color** (*int* idx) const
- void **set_vertex_bones** (*int* idx, *IntArray* bones)
- *IntArray* **get_vertex_bones** (*int* idx) const
- void **set_vertex_weights** (*int* idx, *RealArray* weights)
- *RealArray* **get_vertex_weights** (*int* idx) const
- void **set_vertex_meta** (*int* idx, var meta)
- void **get_vertex_meta** (*int* idx) const
- *IntArray* **get_vertex_edges** (*int* idx) const
- *IntArray* **get_vertex_faces** (*int* idx) const
- *int* **get_edge_vertex** (*int* idx, *int* vertex) const
- *IntArray* **get_edge_faces** (*int* idx) const
- void **set_edge_meta** (*int* idx, var meta)
- void **get_edge_meta** (*int* idx) const
- *int* **get_face_vertex** (*int* idx, *int* vertex) const
- *int* **get_face_edge** (*int* idx, *int* edge) const
- void **set_face_meta** (*int* idx, var meta)
- void **get_face_meta** (*int* idx) const
- *Vector3* **get_face_normal** (*int* idx) const
- void **set_material** (*Material* material)
- *Object* **get_material** ( ) const

9.155 MeshInstance

Inherits: *GeometryInstance* **<** *VisualInstance* **<** *Spatial* **<** *Node* **<** *Object*

Category: Core

9.155.1 Brief Description

Node that instances meshes into a scenario.

9.155.2 Member Functions

void	*set_mesh* (*Mesh* mesh)
Mesh	*get_mesh* ( ) const
void	*set_skeleton_path* (*NodePath* skeleton_path)
NodePath	*get_skeleton_path* ( )
AABB	*get_aabb* ( ) const
void	*create_trimesh_collision* ( )
void	*create_convex_collision* ( )

9.155.3 Description

MeshInstance is a *Node* that takes a *Mesh* resource and adds it to the current scenario by creating an instance of it. This is the class most often used to get 3D geometry rendered and can be used to instance a single *Mesh* in many places. This allows to reuse geometry and save on resources. When a *Mesh* has to be instanced more than thousands of times at close proximity, consider using a *MultiMesh* in a *MultiMeshInstance* instead.

9.155.4 Member Function Description

- void **set_mesh** (*Mesh* mesh)

Set the *Mesh* resource for the instance.

- *Mesh* **get_mesh** ( ) const

Return the current *Mesh* resource for the instance.

- void **set_skeleton_path** (*NodePath* skeleton_path)
- *NodePath* **get_skeleton_path** ( )
- *AABB* **get_aabb** ( ) const

Return the AABB of the mesh, in local coordinates.

- void **create_trimesh_collision** ( )

This helper creates a *StaticBody* child *Node* using the mesh geometry as collision. It's mainly used for testing.

- void **create_convex_collision** ( )

9.156 MeshLibrary

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.156.1 Brief Description

Library of meshes.

9.156.2 Member Functions

void	*create_item* (*int* id)
void	*set_item_name* (*int* id, *String* name)
void	*set_item_mesh* (*int* id, *Mesh* mesh)
void	*set_item_shape* (*int* id, *Shape* shape)
String	*get_item_name* (*int* id) const
Mesh	*get_item_mesh* (*int* id) const
Shape	*get_item_shape* (*int* id) const
void	*remove_item* (*int* id)
void	*clear* ( )
IntArray	*get_item_list* ( ) const
int	*get_last_unused_item_id* ( ) const

9.156.3 Description

Library of meshes. Contains a list of *Mesh* resources, each with name and ID. Useful for GridMap or painting Terrain.

9.156.4 Member Function Description

- void **create_item** (*int* id)

Create a new item in the library, supplied an id.

- void **set_item_name** (*int* id, *String* name)

Set the name of the item.

- void **set_item_mesh** (*int* id, *Mesh* mesh)

Set the mesh of the item.

- void **set_item_shape** (*int* id, *Shape* shape)
- *String* **get_item_name** (*int* id) const

Return the name of the item.

- *Mesh* **get_item_mesh** (*int* id) const

Return the mesh of the item.

- *Shape* **get_item_shape** (*int* id) const
- void **remove_item** (*int* id)

Remove the item.

- void **clear** ( )

Clear the library.

- *IntArray* **get_item_list** ( ) const

Return the list of items.

- *int* **get_last_unused_item_id** ( ) const

Get an unused id for a new item.

9.157 MultiMesh

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.157.1 Brief Description

Provides high performance mesh instancing.

9.157.2 Member Functions

void	*set_mesh* (*Mesh* mesh)
Mesh	*get_mesh* ( ) const
void	*set_instance_count* (*int* count)
int	*get_instance_count* ( ) const
void	*set_instance_transform* (*int* instance, *Transform* transform)
Transform	*get_instance_transform* (*int* instance) const
void	*set_instance_color* (*int* instance, *Color* color)
Color	*get_instance_color* (*int* instance) const
void	*set_aabb* (*AABB* visibility_aabb)
AABB	*get_aabb* ( ) const
void	*generate_aabb* ( )

9.157.3 Description

MultiMesh provides low level mesh instancing. If the amount of *Mesh* instances needed goes from hundreds to thousands (and most need to be visible at close proximity) creating such a large amount of *MeshInstance* nodes may affect performance by using too much CPU or video memory.

For this case a MultiMesh becomes very useful, as it can draw thousands of instances with little API overhead.

As a drawback, if the instances are too far away of each other, performance may be reduced as every single instance will always rendered (they are spatially indexed as one, for the whole object).

Since instances may have any behavior, the AABB used for visibility must be provided by the user, or generated with *generate_aabb*.

9.157.4 Member Function Description

- void **set_mesh** (*Mesh* mesh)

Set the *Mesh* resource to be drawn in multiple instances.

- *Mesh* **get_mesh** ( ) const

Return the *Mesh* resource drawn as multiple instances.

- void **set_instance_count** (*int* count)

Set the amount of instances that is going to be drawn. Changing this number will erase all the existing instance transform and color data.

- *int* **get_instance_count** ( ) const

Return the amount of instances that is going to be drawn.

- void **set_instance_transform** (*int* instance, *Transform* transform)

Set the transform for a specific instance.

- *Transform* **get_instance_transform** (*int* instance) const

Return the transform of a specific instance.

- void **set_instance_color** (*int* instance, *Color* color)

Set the color of a specific instance.

- *Color* **get_instance_color** (*int* instance) const

Get the color of a specific instance.

- void **set_aabb** (*AABB* visibility_aabb)

Set the visibility AABB. If not provided, MultiMesh will not be visible.

- *AABB* **get_aabb** ( ) const

Return the visibility AABB.

- void **generate_aabb** ( )

Generate a new visibility AABB, using mesh AABB and instance transforms. Since instance information is stored in the *VisualServer*, this function is VERY SLOW and must NOT be used often.

9.158 MultiMeshInstance

Inherits: *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.158.1 Brief Description

Node that instances a *MultiMesh*.

9.158.2 Member Functions

void	*set_multimesh* (*Object* multimesh)
Object	*get_multimesh* ( ) const

9.158.3 Description

MultiMeshInstance is a *Node* that takes a *MultiMesh* resource and adds it to the current scenario by creating an instance of it (yes, this is an instance of instances).

9.158.4 Member Function Description

- void **set_multimesh** (*Object* multimesh)

Set the *MultiMesh* to be instance.

- *Object* **get_multimesh** ( ) const

Return the *MultiMesh* that is used for instancing.

9.159 Mutex

Inherits: *Reference* < *Object*

Category: Core

9.159.1 Brief Description

9.159.2 Member Functions

void	*lock* ( )
Error	*try_lock* ( )
void	*unlock* ( )

9.159.3 Member Function Description

- void **lock** ( )
- Error **try_lock** ( )
- void **unlock** ( )

9.160 Navigation

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.160.1 Brief Description

9.160.2 Member Functions

int	*navmesh_create* (*NavigationMesh* mesh, *Transform* xform, *Object* owner=NULL)
void	*navmesh_set_transform* (*int* id, *Transform* xform)
void	*navmesh_remove* (*int* id)
Vector3Array	*get_simple_path* (*Vector3* start, *Vector3* end, *bool* optimize=true)
Vector3	*get_closest_point_to_segment* (*Vector3* start, *Vector3* end, *bool* use_collision=false)
Vector3	*get_closest_point* (*Vector3* to_point)
Vector3	*get_closest_point_normal* (*Vector3* to_point)
Object	*get_closest_point_owner* (*Vector3* to_point)
void	*set_up_vector* (*Vector3* up)
Vector3	*get_up_vector* ( ) const

9.160.3 Member Function Description

- *int* **navmesh_create** (*NavigationMesh* mesh, *Transform* xform, *Object* owner=NULL)
- void **navmesh_set_transform** (*int* id, *Transform* xform)
- void **navmesh_remove** (*int* id)
- *Vector3Array* **get_simple_path** (*Vector3* start, *Vector3* end, *bool* optimize=true)
- *Vector3* **get_closest_point_to_segment** (*Vector3* start, *Vector3* end, *bool* use_collision=false)
- *Vector3* **get_closest_point** (*Vector3* to_point)
- *Vector3* **get_closest_point_normal** (*Vector3* to_point)
- *Object* **get_closest_point_owner** (*Vector3* to_point)
- void **set_up_vector** (*Vector3* up)
- *Vector3* **get_up_vector** ( ) const

9.161 Navigation2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.161.1 Brief Description

9.161.2 Member Functions

int	*navpoly_create* (*NavigationPolygon* mesh, *Matrix32* xform, *Object* owner=NULL)
void	*navpoly_set_transform* (*int* id, *Matrix32* xform)
void	*navpoly_remove* (*int* id)
Vector2Array	*get_simple_path* (*Vector2* start, *Vector2* end, *bool* optimize=true)
Vector2	*get_closest_point* (*Vector2* to_point)
Object	*get_closest_point_owner* (*Vector2* to_point)

9.161.3 Member Function Description

- *int* **navpoly_create** (*NavigationPolygon* mesh, *Matrix32* xform, *Object* owner=NULL)
- void **navpoly_set_transform** (*int* id, *Matrix32* xform)
- void **navpoly_remove** (*int* id)
- *Vector2Array* **get_simple_path** (*Vector2* start, *Vector2* end, *bool* optimize=true)
- *Vector2* **get_closest_point** (*Vector2* to_point)
- *Object* **get_closest_point_owner** (*Vector2* to_point)

9.162 NavigationMesh

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.162.1 Brief Description

9.162.2 Member Functions

void	*set_vertices* (*Vector3Array* vertices)
Vector3Array	*get_vertices* ( ) const
void	*add_polygon* (*IntArray* polygon)
int	*get_polygon_count* ( ) const
IntArray	*get_polygon* (*int* idx)
void	*clear_polygons* ( )

9.162.3 Member Function Description

- void **set_vertices** (*Vector3Array* vertices)
- *Vector3Array* **get_vertices** ( ) const
- void **add_polygon** (*IntArray* polygon)
- *int* **get_polygon_count** ( ) const
- *IntArray* **get_polygon** (*int* idx)
- void **clear_polygons** ( )

9.163 NavigationMeshInstance

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.163.1 Brief Description

9.163.2 Member Functions

void	*set_navigation_mesh* (*Object* navmesh)
Object	*get_navigation_mesh* ( ) const
void	*set_enabled* (*bool* enabled)
bool	*is_enabled* ( ) const

9.163.3 Member Function Description

- void **set_navigation_mesh** (*Object* navmesh)
- *Object* **get_navigation_mesh** ( ) const
- void **set_enabled** (*bool* enabled)
- *bool* **is_enabled** ( ) const

9.164 NavigationPolygon

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.164.1 Brief Description

9.164.2 Member Functions

void	*set_vertices* (*Vector2Array* vertices)
Vector2Array	*get_vertices* ( ) const
void	*add_polygon* (*IntArray* polygon)
int	*get_polygon_count* ( ) const
IntArray	*get_polygon* (*int* idx)
void	*clear_polygons* ( )
void	*add_outline* (*Vector2Array* outline)
void	*add_outline_at_index* (*Vector2Array* outline, *int* index)
int	*get_outline_count* ( ) const
void	*set_outline* (*int* idx, *Vector2Array* outline)
Vector2Array	*get_outline* (*int* idx) const
void	*remove_outline* (*int* idx)
void	*clear_outlines* ( )
void	*make_polygons_from_outlines* ( )

9.164.3 Member Function Description

- void **set_vertices** (*Vector2Array* vertices)
- *Vector2Array* **get_vertices** ( ) const
- void **add_polygon** (*IntArray* polygon)

- *int* **get_polygon_count** ( ) const
- *IntArray* **get_polygon** (*int* idx)
- void **clear_polygons** ( )
- void **add_outline** (*Vector2Array* outline)
- void **add_outline_at_index** (*Vector2Array* outline, *int* index)
- *int* **get_outline_count** ( ) const
- void **set_outline** (*int* idx, *Vector2Array* outline)
- *Vector2Array* **get_outline** (*int* idx) const
- void **remove_outline** (*int* idx)
- void **clear_outlines** ( )
- void **make_polygons_from_outlines** ( )

9.165 NavigationPolygonInstance

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.165.1 Brief Description

9.165.2 Member Functions

void	*set_navigation_polygon* (*NavigationPolygon* navpoly)
NavigationPolygon	*get_navigation_polygon* ( ) const
void	*set_enabled* (*bool* enabled)
bool	*is_enabled* ( ) const

9.165.3 Member Function Description

- void **set_navigation_polygon** (*NavigationPolygon* navpoly)
- *NavigationPolygon* **get_navigation_polygon** ( ) const
- void **set_enabled** (*bool* enabled)
- *bool* **is_enabled** ( ) const

9.166 Nil

Category: Built-In Types

9.166.1 Brief Description

9.166.2 Member Functions

void	Nil (*bool* from)
void	Nil (*int* from)
void	Nil (*float* from)
void	Nil (*String* from)
void	Nil (*Vector2* from)
void	Nil (*Rect2* from)
void	Nil (*Vector3* from)
void	Nil (*Matrix32* from)
void	Nil (*Plane* from)
void	Nil (*Quat* from)
void	Nil (*AABB* from)
void	Nil (*Matrix3* from)
void	Nil (*Transform* from)
void	Nil (*Color* from)
void	Nil (*Image* from)
void	Nil (*NodePath* from)
void	Nil (*RID* from)
void	Nil (*Object* from)
void	Nil (*InputEvent* from)
void	Nil (*Dictionary* from)
void	Nil (*Array* from)
void	Nil (*RawArray* from)
void	Nil (*IntArray* from)
void	Nil (*RealArray* from)
void	Nil (*StringArray* from)
void	Nil (*Vector2Array* from)
void	Nil (*Vector3Array* from)
void	Nil (*ColorArray* from)

9.166.3 Member Function Description

- void **Nil** (*bool* from)
- void **Nil** (*int* from)
- void **Nil** (*float* from)
- void **Nil** (*String* from)
- void **Nil** (*Vector2* from)
- void **Nil** (*Rect2* from)
- void **Nil** (*Vector3* from)
- void **Nil** (*Matrix32* from)
- void **Nil** (*Plane* from)
- void **Nil** (*Quat* from)
- void **Nil** (*AABB* from)

- void **Nil** (*Matrix3* from)
- void **Nil** (*Transform* from)
- void **Nil** (*Color* from)
- void **Nil** (*Image* from)
- void **Nil** (*NodePath* from)
- void **Nil** (*RID* from)
- void **Nil** (*Object* from)
- void **Nil** (*InputEvent* from)
- void **Nil** (*Dictionary* from)
- void **Nil** (*Array* from)
- void **Nil** (*RawArray* from)
- void **Nil** (*IntArray* from)
- void **Nil** (*RealArray* from)
- void **Nil** (*StringArray* from)
- void **Nil** (*Vector2Array* from)
- void **Nil** (*Vector3Array* from)
- void **Nil** (*ColorArray* from)

9.167 Node

Inherits: *Object*

Inherited By: *Viewport*, *Timer*, *CanvasLayer*, *EventPlayer*, *SoundRoomParams*, *Spatial*, *AnimationPlayer*, *EditorPlugin*, *ResourcePreloader*, *AnimationTreePlayer*, *SamplePlayer*, *InstancePlaceholder*, *StreamPlayer*, *CanvasItem*, *Tween*

Category: Core

9.167.1 Brief Description

Base class for all the "Scene" elements.

9.167.2 Member Functions

void	*_enter_tree* ( ) virtual
void	*_exit_tree* ( ) virtual
void	*_fixed_process* (*float* delta) virtual
void	*_input* (*InputEvent* event) virtual
void	*_process* (*float* delta) virtual
void	*_ready* ( ) virtual
void	*_unhandled_input* (*InputEvent* event) virtual

Continúa en la página siguiente

Tabla 9.15 – proviene de la página anterior

void	*_unhandled_key_input* (*InputEvent* key_event) virtual
void	*set_name* (*String* name)
String	*get_name* ( ) const
void	*add_child* (*Node* node, *bool* legible_unique_name=false)
void	*remove_child* (*Node* node)
int	*get_child_count* ( ) const
Array	*get_children* ( ) const
Node	*get_child* (*int* idx) const
bool	*has_node* (*NodePath* path) const
Node	*get_node* (*NodePath* path) const
Node	*get_parent* ( ) const
Node	*find_node* (*String* mask, *bool* recursive=true, *bool* owned=true) const
bool	*has_node_and_resource* (*NodePath* path) const
Array	*get_node_and_resource* (*NodePath* path)
bool	*is_inside_tree* ( ) const
bool	*is_a_parent_of* (*Node* node) const
bool	*is_greater_than* (*Node* node) const
NodePath	*get_path* ( ) const
NodePath	*get_path_to* (*Node* node) const
void	*add_to_group* (*String* group, *bool* persistent=false)
void	*remove_from_group* (*String* group)
bool	*is_in_group* (*String* group) const
void	*move_child* (*Node* child_node, *int* to_pos)
Array	*get_groups* ( ) const
void	*raise* ( )
void	*set_owner* (*Node* owner)
Node	*get_owner* ( ) const
void	*remove_and_skip* ( )
int	*get_index* ( ) const
void	*print_tree* ( )
void	*set_filename* (*String* filename)
String	*get_filename* ( ) const
void	*propagate_notification* (*int* what)
void	*set_fixed_process* (*bool* enable)
float	*get_fixed_process_delta_time* ( ) const
bool	*is_fixed_processing* ( ) const
void	*set_process* (*bool* enable)
float	*get_process_delta_time* ( ) const
bool	*is_processing* ( ) const
void	*set_process_input* (*bool* enable)
bool	*is_processing_input* ( ) const
void	*set_process_unhandled_input* (*bool* enable)
bool	*is_processing_unhandled_input* ( ) const
void	*set_process_unhandled_key_input* (*bool* enable)
bool	*is_processing_unhandled_key_input* ( ) const
void	*set_pause_mode* (*int* mode)
int	*get_pause_mode* ( ) const
bool	*can_process* ( ) const
void	*print_stray_nodes* ( )
int	*get_position_in_parent* ( ) const

Continúa en la página siguiente

Tabla 9.15 – proviene de la página anterior

SceneTree	*get_tree* ( ) const
Node	*duplicate* (*bool* use_instancing=false) const
void	*replace_by* (*Node* node, *bool* keep_data=false)
void	*set_scene_instance_load_placeholder* (*bool* load_placeholder)
bool	*get_scene_instance_load_placeholder* ( ) const
Object	*get_viewport* ( ) const
void	*queue_free* ( )

9.167.3 Signals

- **renamed** ( )
- **enter_tree** ( )
- **exit_tree** ( )

9.167.4 Numeric Constants

- **NOTIFICATION_ENTER_TREE = 10**
- **NOTIFICATION_EXIT_TREE = 11**
- **NOTIFICATION_MOVED_IN_PARENT = 12**
- **NOTIFICATION_READY = 13**
- **NOTIFICATION_FIXED_PROCESS = 16**
- **NOTIFICATION_PROCESS = 17** — Notification received every frame when the process flag is set (see *set_process*).
- **NOTIFICATION_PARENTED = 18** — Notification received when a node is set as a child of another node. Note that this doesn't mean that a node entered the Scene Tree.
- **NOTIFICATION_UNPARENTED = 19** — Notification received when a node is unparented (parent removed it from the list of children).
- **NOTIFICATION_PAUSED = 14**
- **NOTIFICATION_UNPAUSED = 15**
- **NOTIFICATION_INSTANCED = 20**
- **PAUSE_MODE_INHERIT = 0**
- **PAUSE_MODE_STOP = 1**
- **PAUSE_MODE_PROCESS = 2**

9.167.5 Description

Nodes can be set as children of other nodes, resulting in a tree arrangement. Any tree of nodes is called a "Scene".

Scenes can be saved to disk, and then instanced into other scenes. This allows for very high flexibility in the architecture and data model of the projects.

SceneTree contains the "active" tree of nodes, and a node becomes active (receiving NOTIFICATION_ENTER_SCENE) when added to that tree.

A node can contain any number of nodes as a children (but there is only one tree root) with the requirement that no two children with the same name can exist.

Nodes can, optionally, be added to groups. This makes it easy to reach a number of nodes from the code (for example an "enemies" group).

Nodes can be set to "process" state, so they constantly receive a callback requesting them to process (do anything). Normal processing (*_process*) happens as fast as possible and is dependent on the frame rate, so the processing time delta is variable. Fixed processing (*_fixed_process*) happens a fixed amount of times per second (by default 60) and is useful to link itself to the physics.

Nodes can also process input events. When set, the *_input* function will be called with every input that the program receives. Since this is usually too overkill (unless used for simple projects), an *_unhandled_input* function is called when the input was not handled by anyone else (usually, GUI *Control* nodes).

To keep track of the scene hierarchy (specially when instancing scenes into scenes) an "owner" can be set to a node. This keeps track of who instanced what. This is mostly useful when writing editors and tools, though.

Finally, when a node is freed, it will free all its children nodes too.

9.167.6 Member Function Description

- void **_enter_tree** **(** **)** virtual
- void **_exit_tree** **(** **)** virtual
- void **_fixed_process** **(** *float* delta **)** virtual

Called for fixed processing (synced to the physics).

- void **_input** **(** *InputEvent* event **)** virtual

Called when any input happens (also must enable with *set_process_input* or the property).

- void **_process** **(** *float* delta **)** virtual

Called for processing. This is called every frame, with the delta time from the previous frame.

- void **_ready** **(** **)** virtual

Called when ready (entered scene and children entered too).

- void **_unhandled_input** **(** *InputEvent* event **)** virtual

Called when any input happens that was not handled by something else (also must enable with *set_process_unhandled_input* or the property).

- void **_unhandled_key_input** **(** *InputEvent* key_event **)** virtual

Called when any key input happens that was not handled by something else.

- void **set_name** **(** *String* name **)**

Set the name of the *Node*. Name must be unique within parent, and setting an already existing name will cause for the node to be automatically renamed.

- *String* **get_name** **(** **)** const

Return the name of the *Node*. Name is be unique within parent.

- void **add_child** **(** *Node* node, *bool* legible_unique_name=false **)**

Add a child *Node*. Nodes can have as many children as they want, but every child must have a unique name. Children nodes are automatically deleted when the parent node is deleted, so deleting a whole scene is performed by deleting its topmost node.

The optional boolean argument enforces creating child node with human-readable names, based on the name of node being instanced instead of its type only.

- void **remove_child** (*Node* node)

Remove a child *Node*. Node is NOT deleted and will have to be deleted manually.

- *int* **get_child_count** ( ) const

Return the amount of children nodes.

- *Array* **get_children** ( ) const
- *Node* **get_child** (*int* idx) const

Return a children node by it's index (see *get_child_count*). This method is often used for iterating all children of a node.

- *bool* **has_node** (*NodePath* path) const
- *Node* **get_node** (*NodePath* path) const

Fetch a node. NodePath must be valid (or else error will occur) and can be either the path to child node, a relative path (from the current node to another node), or an absolute path to a node.

Note: fetching absolute paths only works when the node is inside the scene tree (see *is_inside_tree*). Examples. Assume your current node is Character and following tree:

root/

root/Character

root/Character/Sword

root/Character/Backpack/Dagger

root/MyGame

root/Swamp/Alligator

root/Swamp/Mosquito

root/Swamp/Goblin

Possible paths are:

- get_node("Sword")
- get_node("Backpack/Dagger")
- get_node("../Swamp/Alligator")
- get_node("/root/MyGame")

- *Node* **get_parent** ( ) const

Return the parent *Node* of the current *Node*, or an empty Object if the node lacks a parent.

- *Node* **find_node** (*String* mask, *bool* recursive=true, *bool* owned=true) const

Find a descendant of this node whose name matches `mask` as in *String.match* (i.e. case sensitive, but '*' matches zero or more characters and '?' matches any single character except '.'). Note that it does not match against the full path, just against individual node names.

- *bool* **has_node_and_resource** (*NodePath* path) const
- *Array* **get_node_and_resource** (*NodePath* path)
- *bool* **is_inside_tree** ( ) const

- *bool* **is_a_parent_of** (*Node* node) const

Return *true* if the "node" argument is a direct or indirect child of the current node, otherwise return *false*.

- *bool* **is_greater_than** (*Node* node) const

Return *true* if "node" occurs later in the scene hierarchy than the current node, otherwise return *false*.

- *NodePath* **get_path** ( ) const

Return the absolute path of the current node. This only works if the current node is inside the scene tree (see *is_inside_tree*).

- *NodePath* **get_path_to** (*Node* node) const

Return the relative path from the current node to the specified node in "node" argument. Both nodes must be in the same scene, or else the function will fail.

- void **add_to_group** (*String* group, *bool* persistent=false)

Add a node to a group. Groups are helpers to name and organize group of nodes, like for example: "Enemies", "Collectables", etc. A *Node* can be in any number of groups. Nodes can be assigned a group at any time, but will not be added to it until they are inside the scene tree (see *is_inside_tree*).

- void **remove_from_group** (*String* group)

Remove a node from a group.

- *bool* **is_in_group** (*String* group) const
- void **move_child** (*Node* child_node, *int* to_pos)

Move a child node to a different position (order) amongst the other children. Since calls, signals, etc are performed by tree order, changing the order of children nodes may be useful.

- *Array* **get_groups** ( ) const
- void **raise** ( )

Move this node to the top of the array of nodes of the parent node. This is often useful on GUIs (*Control*), because their order of drawing fully depends on their order in the tree.

- void **set_owner** (*Node* owner)

Set the node owner. A node can have any other node as owner (as long as a valid parent, grandparent, etc ascending in the tree). When saving a node (using SceneSaver) all the nodes it owns will be saved with it. This allows to create complex SceneTrees, with instancing and subinstancing.

- *Node* **get_owner** ( ) const

Get the node owner (see *set_owner*).

- void **remove_and_skip** ( )

Remove a node and set all its children as children of the parent node (if exists). All even subscriptions that pass by the removed node will be unsubscribed.

- *int* **get_index** ( ) const

Get the node index in the parent (assuming it has a parent).

- void **print_tree** ( )

Print the scene to stdout. Used mainly for debugging purposes.

- void **set_filename** (*String* filename)

A node can contain a filename. This filename should not be changed by the user, unless writing editors and tools. When a scene is instanced from a file, it topmost node contains the filename from where it was loaded.

- *String* **get_filename** ( ) const

Return a filename that may be containedA node can contained by the node. When a scene is instanced from a file, it topmost node contains the filename from where it was loaded (see *set_filename*).

- void **propagate_notification** (*int* what)

Notify the current node and all its children recursively by calling notification() in all of them.

- void **set_fixed_process** (*bool* enable)

Enables or disables node fixed framerate processing. When a node is being processed, it will receive a NOTIFICATION_PROCESS at a fixed (usually 60 fps, check *OS* to change that) interval (and the *_fixed_process* callback will be called if exists). It is common to check how much time was elapsed since the previous frame by calling *get_fixed_process_delta_time*.

- *float* **get_fixed_process_delta_time** ( ) const

Return the time elapsed since the last fixed frame. This is always the same in fixed processing unless the frames per second is changed in *OS*.

- *bool* **is_fixed_processing** ( ) const

Return true if fixed processing is enabled (see *set_fixed_process*).

- void **set_process** (*bool* enable)

Enables or disables node processing. When a node is being processed, it will receive a NOTIFICATION_PROCESS on every drawn frame (and the *_process* callback will be called if exists). It is common to check how much time was elapsed since the previous frame by calling *get_process_delta_time*.

- *float* **get_process_delta_time** ( ) const

Return the time elapsed (in seconds) since the last process callback. This is almost always different each time.

- *bool* **is_processing** ( ) const

Return whether processing is enabled in the current node (see *set_process*).

- void **set_process_input** (*bool* enable)

Enable input processing for node. This is not required for GUI controls! It hooks up the node to receive all input (see *_input*).

- *bool* **is_processing_input** ( ) const

Return true if the node is processing input (see *set_process_input*).

- void **set_process_unhandled_input** (*bool* enable)

Enable unhandled input processing for node. This is not required for GUI controls! It hooks up the node to receive all input that was not previously handled before (usually by a *Control*). (see *_unhandled_input*).

- *bool* **is_processing_unhandled_input** ( ) const

Return true if the node is processing unhandled input (see *set_process_unhandled_input*).

- void **set_process_unhandled_key_input** (*bool* enable)
- *bool* **is_processing_unhandled_key_input** ( ) const
- void **set_pause_mode** (*int* mode)
- *int* **get_pause_mode** ( ) const

- *bool* **can_process** ( ) const

Return true if the node can process.

- void **print_stray_nodes** ( )
- *int* **get_position_in_parent** ( ) const
- *SceneTree* **get_tree** ( ) const
- *Node* **duplicate** (*bool* use_instancing=false) const
- void **replace_by** (*Node* node, *bool* keep_data=false)

Replace a node in a scene by a given one. Subscriptions that pass through this node will be lost.

- void **set_scene_instance_load_placeholder** (*bool* load_placeholder)
- *bool* **get_scene_instance_load_placeholder** ( ) const
- *Object* **get_viewport** ( ) const
- void **queue_free** ( )

9.168 Node2D

Inherits: *CanvasItem* < *Node* < *Object*

Inherited By: *RemoteTransform2D*, *Joint2D*, *ParticleAttractor2D*, *CollisionObject2D*, *VisibilityNotifier2D*, *TileMap*, *Navigation2D*, *CollisionPolygon2D*, *TouchScreenButton*, *Particles2D*, *AnimatedSprite*, *Light2D*, *SoundPlayer2D*, *ViewportSprite*, *Path2D*, *Sprite*, *RayCast2D*, *CollisionShape2D*, *NavigationPolygonInstance*, *PathFollow2D*, *ParallaxLayer*, *Polygon2D*, *Position2D*, *LightOccluder2D*, *BackBufferCopy*, *CanvasModulate*, *YSort*, *Camera2D*

Category: Core

9.168.1 Brief Description

Base node for 2D system.

9.168.2 Member Functions

void	*set_pos* (*Vector2* pos)
void	*set_rot* (*float* radians)
void	*set_rotd* (*float* degrees)
void	*set_scale* (*Vector2* scale)
Vector2	*get_pos* ( ) const
float	*get_rot* ( ) const
float	*get_rotd* ( ) const
Vector2	*get_scale* ( ) const
void	*rotate* (*float* radians)
void	*move_local_x* (*float* delta, *bool* scaled=false)
void	*move_local_y* (*float* delta, *bool* scaled=false)
void	*translate* (*Vector2* offset)
void	*global_translate* (*Vector2* offset)
void	*scale* (*Vector2* ratio)
void	*set_global_pos* (*Vector2* pos)
Vector2	*get_global_pos* ( ) const
void	*set_transform* (*Matrix32* xform)
void	*set_global_transform* (*Matrix32* xform)
void	*look_at* (*Vector2* point)
float	*get_angle_to* (*Vector2* point) const
void	*set_z* (*int* z)
int	*get_z* ( ) const
void	*set_z_as_relative* (*bool* enable)
bool	*is_z_relative* ( ) const
void	*edit_set_pivot* (*Vector2* pivot)
Matrix32	*get_relative_transform_to_parent* (*Object* parent) const

9.168.3 Description

Base node for 2D system. Node2D contains a position, rotation and scale, which is used to position and animate. It can alternatively be used with a custom 2D transform (*Matrix32*). A tree of Node2Ds allows complex hierarchies for animation and positioning.

9.168.4 Member Function Description

- void **set_pos** (*Vector2* pos)

Set the position of the 2D node.

- void **set_rot** (*float* radians)

Set the rotation of the 2D node.

- void **set_rotd** (*float* degrees)

Set the rotation of the 2D node.

- void **set_scale** (*Vector2* scale)

Set the scale of the 2D node.

- *Vector2* **get_pos** ( ) const

Return the position of the 2D node.

- *float* **get_rot ( )** const

Return the rotation of the 2D node.

- *float* **get_rotd ( )** const
- *Vector2* **get_scale ( )** const

Return the scale of the 2D node.

- void **rotate (** *float* radians **)**

Apply a 'radians' rotation to the 2D node, starting from its current rotation.

- void **move_local_x (** *float* delta, *bool* scaled=false **)**

Apply a local translation on X axis to the 2D node according to the 'delta' of the process. If 'scaled' is false, the movement is normalized.

- void **move_local_y (** *float* delta, *bool* scaled=false **)**

Apply a local translation on Y axis to the 2D node according to the 'delta' of the process. If 'scaled' is false, the movement is normalized.

- void **translate (** *Vector2* offset **)**

Apply a local translation of 'offset' to the 2D node, starting from its current local position.

- void **global_translate (** *Vector2* offset **)**

Apply a global translation of 'offset' to the 2D node, starting from its current global position.

- void **scale (** *Vector2* ratio **)**

Apply the 'ratio' scale to the 2D node, according to its current scale value.

- void **set_global_pos (** *Vector2* pos **)**

Set the global position of the 2D node to 'pos'.

- *Vector2* **get_global_pos ( )** const

Return the global position of the 2D node.

- void **set_transform (** *Matrix32* xform **)**

Set the local transform *Matrix32* of the 2D node.

- void **set_global_transform (** *Matrix32* xform **)**

Set the global transform *Matrix32* of the 2D node.

- void **look_at (** *Vector2* point **)**

Rotate the 2d node so it points at 'point' position.

- *float* **get_angle_to (** *Vector2* point **)** const

Return the rotation angle in radians needed for the 2d node to point at 'point' position.

- void **set_z (** *int* z **)**

Set the Z-index value of the 2D node.

- *int* **get_z ( )** const

Return the Z-index of the 2D node.

- void **set_z_as_relative (** *bool* enable **)**

Set the Z-index value as relative to the parent node of this 2D node. Thus, if this 2D node's Z-index value is 2 and its parent's effective Z-index is 3, then the effective Z-index value of this 2D node would be 3 + 2 = 5.

- *bool* **is_z_relative** ( ) const

Return true if the Z-index value of this 2D node is relative to its parent's. Else, return false.

- void **edit_set_pivot** (*Vector2* pivot)

Set the pivot position of the 2D node to 'pivot' value. This method is implemented only in some nodes that inherit Node2D.

- *Matrix32* **get_relative_transform_to_parent** (*Object* parent) const

Return the transform *Matrix32* calculated relatively to the parent of this 2D node.

9.169 NodePath

Category: Built-In Types

9.169.1 Brief Description

Pre-parsed scene tree path.

9.169.2 Member Functions

String	*get_name* (*int* idx)
int	*get_name_count* ( )
String	*get_property* ( )
String	*get_subname* (*int* idx)
int	*get_subname_count* ( )
bool	*is_absolute* ( )
bool	*is_empty* ( )
NodePath	*NodePath* (*String* from)

9.169.3 Description

A pre-parsed relative or absolute path in a scene tree, for use with *Node.get_node* and similar functions. It can reference a node, a resource within a node, or a property of a node or resource. For instance, `"Path2D/PathFollow2D/Sprite:texture:size"` would refer to the size property of the texture resource on the node named "Sprite" which is a child of the other named nodes in the path. Note that if you want to get a resource, you must end the path with a colon, otherwise the last element will be used as a property name.

You will usually just pass a string to *Node.get_node* and it will be automatically converted, but you may occasionally want to parse a path ahead of time with *NodePath* or the literal syntax `@"path"`. Exporting a *NodePath* variable will give you a node selection widget in the properties panel of the editor, which can often be useful.

A *NodePath* is made up of a list of node names, a list of "subnode" (resource) names, and the name of a property in the final node or resource.

9.169.4 Member Function Description

- *String* **get_name** (*int* idx)

Get the node name indicated by `idx` (0 to *get_name_count*)

- *int* **get_name_count** ( )

Get the number of node names which make up the path.

- *String* **get_property** ( )

Get the path's property name, or an empty string if the path doesn't have a property.

- *String* **get_subname** (*int* idx)

Get the resource name indicated by `idx` (0 to *get_subname_count*)

- *int* **get_subname_count** ( )

Get the number of resource names in the path.

- *bool* **is_absolute** ( )

Return true if the node path is absolute (not relative).

- *bool* **is_empty** ( )

Return true if the node path is empty.

- *NodePath* **NodePath** (*String* from)

Create a NodePath from a string, e.g. "Path2D/PathFollow2D/Sprite:texture:size". A path is absolute if it starts with a slash. Absolute paths are only valid in the global scene tree, not within individual scenes. In a relative path, `"."` and `".."` indicate the current node and its parent.

9.170 Object

Inherited By: *Reference*, *Physics2DServer*, *Input*, *SpatialSound2DServer*, *Node*, *Geometry*, *TreeItem*, *PhysicsDirectSpaceState*, *Physics2DDirectSpaceState*, *MainLoop*, *InputMap*, *UndoRedo*, *PhysicsServer*, *ResourceSaver*, *Performance*, *PathRemap*, *ResourceLoader*, *AudioServer*, *SpatialSoundServer*, *IP*, *VisualServer*, *OS*, *Physics2DDirectBodyState*, *Globals*, *PhysicsDirectBodyState*, *TranslationServer*

Category: Core

9.170.1 Brief Description

Base class for all non built-in types.

9.170.2 Member Functions

void	*_get* (*String* property) virtual
Array	*_get_property_list* ( ) virtual
void	*_init* ( ) virtual
void	*_notification* (*int* what) virtual
void	*_set* (*String* property, var value) virtual

Tabla 9.16 – proviene de la página anterior

void	*free* ( )
String	*get_type* ( ) const
bool	*is_type* (*String* type) const
void	*set* (*String* property, var value)
void	*get* (*String* property) const
Array	*get_property_list* ( ) const
Array	*get_method_list* ( ) const
void	*notification* (*int* what, *bool* reversed=false)
int	*get_instance_ID* ( ) const
void	*set_script* (*Script* script)
Script	*get_script* ( ) const
void	*set_meta* (*String* name, var value)
void	*get_meta* (*String* name) const
bool	*has_meta* (*String* name) const
StringArray	*get_meta_list* ( ) const
void	*add_user_signal* (*String* signal, *Array* arguments=Array())
bool	*has_user_signal* (*String* signal) const
void	*emit_signal* (*String* signal, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL)
void	*call* (*String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=N
void	*call_deferred* (*String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL)
Variant	*callv* (*String* method, *Array* arg_array)
bool	*has_method* (*String* method) const
Array	*get_signal_list* ( ) const
Array	*get_signal_connection_list* (*String* signal) const
int	*connect* (*String* signal, *Object* target, *String* method, *Array* binds=Array(), *int* flags=0)
void	*disconnect* (*String* signal, *Object* target, *String* method)
bool	*is_connected* (*String* signal, *Object* target, *String* method) const
void	*set_block_signals* (*bool* enable)
bool	*is_blocking_signals* ( ) const
void	*set_message_translation* (*bool* enable)
bool	*can_translate_messages* ( ) const
void	*property_list_changed_notify* ( )
String	*XL_MESSAGE* (*String* message) const
String	*tr* (*String* message) const
bool	*is_queued_for_deletion* ( ) const

9.170.3 Signals

- **script_changed** ( )

9.170.4 Numeric Constants

- **NOTIFICATION_POSTINITIALIZE = 0** — Called right when the object is initialized. Not available in script.
- **NOTIFICATION_PREDELETE = 1** — Called before the object is about to be deleted.
- **CONNECT_DEFERRED = 1** — Connect a signal in deferred mode. This way, signal emissions are stored in a queue, then set on idle time.
- **CONNECT_PERSIST = 2** — Persisting connections are saved when the object is serialized to file.

- **CONNECT_ONESHOT = 4** — One short connections disconnect themselves after emission.

9.170.5 Description

Base class for all non built-in types. Everything not a built-in type starts the inheritance chain from this class.

Objects do not manage memory, if inheriting from one the object will most likely have to be deleted manually (call the *free* function from the script or delete from C++).

Some derivates add memory management, such as *Reference* (which keeps a reference count and deletes itself automatically when no longer referenced) and *Node*, which deletes the children tree when deleted.

Objects export properties, which are mainly useful for storage and editing, but not really so much in programming. Properties are exported in *_get_property_list* and handled in *_get* and *_set*. However, scripting languages and C++ have simpler means to export them.

Objects also receive notifications (*_notification*). Notifications are a simple way to notify the object about simple events, so they can all be handled together.

9.170.6 Member Function Description

- void **_get** (*String* property) virtual

Return a property, return null if the property does not exist.

- *Array* **_get_property_list** ( ) virtual

Return the property list, array of dictionaries, dictionaries must contain: name:String, type:int (see TYPE_* enum in globals) and optionally: hint:int (see PROPERTY_HINT_* in globals), hint_string:String, usage:int (see PROPERTY_USAGE_* in globals).

- void **_init** ( ) virtual
- void **_notification** (*int* what) virtual

Notification request, the notification id is received.

- void **_set** (*String* property, var value) virtual

Set a property. Return true if the property was found.

- void **free** ( )
- *String* **get_type** ( ) const

Return the type of the object as a string.

- *bool* **is_type** (*String* type) const

Check the type of the object against a string (including inheritance).

- void **set** (*String* property, var value)

Set property into the object.

- void **get** (*String* property) const

Get a property from the object.

- *Array* **get_property_list** ( ) const

Return the list of properties as an array of dictionaries, dictionaries contain: name:String, type:int (see TYPE_* enum in globals) and optionally: hint:int (see PROPERTY_HINT_* in globals), hint_string:String, usage:int (see PROPERTY_USAGE_* in globals).

- *Array* **get_method_list ( )** const
- void **notification (***int* what, *bool* reversed=false **)**

Notify the object of something.

- *int* **get_instance_ID ( )** const

Return the instance ID. All objects have a unique instance ID.

- void **set_script (***Script* script **)**

Set a script into the object, scripts extend the object functionality.

- *Script* **get_script ( )** const

Return the object script (or null if it doesn't have one).

- void **set_meta (***String* name, var value **)**

Set a metadata into the object. Metadata is serialized. Metadata can be *anything*.

- void **get_meta (***String* name **)** const

Return a metadata from the object.

- *bool* **has_meta (***String* name **)** const

Return true if a metadata is found with the requested name.

- *StringArray* **get_meta_list ( )** const

Return the list of metadata in the object.

- void **add_user_signal (***String* signal, *Array* arguments=Array() **)**

Add a user signal (can be added anytime). Arguments are optional, but can be added as an array of dictionaries, each containing "name" and "type" (from *@Global Scope* TYPE_*).

- *bool* **has_user_signal (***String* signal **)** const
- void **emit_signal (***String* signal, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL **)**

Emit a signal. Arguments are passed in an array.

- void **call (***String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL, var arg6=NULL, var arg7=NULL, var arg8=NULL, var arg9=NULL **)**

Call a function in the object, result is returned.

- void **call_deferred (***String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL **)**

Create and store a function in the object. The call will take place on idle time.

- Variant **callv (***String* method, *Array* arg_array **)**
- *bool* **has_method (***String* method **)** const
- *Array* **get_signal_list ( )** const

Return the list of signals as an array of dictionaries.

- *Array* **get_signal_connection_list (***String* signal **)** const
- *int* **connect (***String* signal, *Object* target, *String* method, *Array* binds=Array(), *int* flags=0 **)**

Connect a signal to a method at a target (member function). Binds are optional and are passed as extra arguments to the call. Flags specify optional deferred or one shot connections, see enum CONNECT_*. A signal can only be connected once to a method, and it will throw an error if already connected. If you want to avoid this, use *is_connected* to check.

- void **disconnect** (*String* signal, *Object* target, *String* method)

Disconnect a signal from a method.

- *bool* **is_connected** (*String* signal, *Object* target, *String* method) const

Return true if a connection exists for a given signal and target/method.

- void **set_block_signals** (*bool* enable)

If set to true, signal emission is blocked.

- *bool* **is_blocking_signals** ( ) const

Return true if signal emission blocking is enabled.

- void **set_message_translation** (*bool* enable)

Set true if this object can translate strings (in calls to tr()). Default is true.

- *bool* **can_translate_messages** ( ) const

Return true if this object can translate strings.

- void **property_list_changed_notify** ( )
- *String* **XL_MESSAGE** (*String* message) const

Deprecated, will go away.

- *String* **tr** (*String* message) const

Translate a message. Only works in message translation is enabled (which is by default). See *set_message_translation*.

- *bool* **is_queued_for_deletion** ( ) const

9.171 OccluderPolygon2D

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.171.1 Brief Description

9.171.2 Member Functions

void	*set_closed* (*bool* closed)
bool	*is_closed* ( ) const
void	*set_cull_mode* (*int* cull_mode)
int	*get_cull_mode* ( ) const
void	*set_polygon* (*Vector2Array* polygon)
Vector2Array	*get_polygon* ( ) const

9.171.3 Numeric Constants

- **CULL_DISABLED = 0**
- **CULL_CLOCKWISE = 1**
- **CULL_COUNTER_CLOCKWISE = 2**

9.171.4 Member Function Description

- void **set_closed** (*bool* closed)
- *bool* **is_closed** ( ) const
- void **set_cull_mode** (*int* cull_mode)
- *int* **get_cull_mode** ( ) const
- void **set_polygon** (*Vector2Array* polygon)
- *Vector2Array* **get_polygon** ( ) const

9.172 OmniLight

Inherits: *Light* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.172.1 Brief Description

OmniDirectional Light, such as a light bulb or a candle.

9.172.2 Description

An OmniDirectional light is a type of *Light* node that emits lights in all directions. The light is attenuated through the distance and this attenuation can be configured by changing the energy, radius and attenuation parameters of *Light*. TODO: Image of an omnilight.

9.173 OptionButton

Inherits: *Button* < *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.173.1 Brief Description

Button control that provides selectable options when pressed.

9.173.2 Member Functions

void	*add_item* (*String* label, *int* id=-1)
void	*add_icon_item* (*Texture* texture, *String* label, *int* id)
void	*set_item_text* (*int* idx, *String* text)
void	*set_item_icon* (*int* idx, *Texture* texture)
void	*set_item_disabled* (*int* idx, *bool* disabled)
void	*set_item_ID* (*int* idx, *int* id)
void	*set_item_metadata* (*int* idx, var metadata)
String	*get_item_text* (*int* idx) const
Texture	*get_item_icon* (*int* idx) const
int	*get_item_ID* (*int* idx) const
void	*get_item_metadata* (*int* idx) const
bool	*is_item_disabled* (*int* idx) const
int	*get_item_count* ( ) const
void	*add_separator* ( )
void	*clear* ( )
void	*select* (*int* idx)
int	*get_selected* ( ) const
int	*get_selected_ID* ( ) const
void	*get_selected_metadata* ( ) const
void	*remove_item* (*int* idx)

9.173.3 Signals

- **item_selected** (*int* ID)

9.173.4 Description

OptionButton is a type button that provides a selectable list of items when pressed. The item selected becomes the "current" item and is displayed as the button text.

9.173.5 Member Function Description

- void **add_item** (*String* label, *int* id=-1)

Add an item, with text "label" and (optionally) id. If no "id" is passed, "id" becomes the item index. New items are appended at the end.

- void **add_icon_item** (*Texture* texture, *String* label, *int* id)

Add an item, with a "texture" icon, text "label" and (optionally) id. If no "id" is passed, "id" becomes the item index. New items are appended at the end.

- void **set_item_text** (*int* idx, *String* text)

Set the text of an item at index "idx".

- void **set_item_icon** (*int* idx, *Texture* texture)

Set the icon of an item at index "idx".

- void **set_item_disabled** (*int* idx, *bool* disabled)
- void **set_item_ID** (*int* idx, *int* id)

Set the ID of an item at index “idx”.

- void **set_item_metadata** (*int* idx, var metadata)
- *String* **get_item_text** (*int* idx) const

Return the text of the item at index “idx”.

- *Texture* **get_item_icon** (*int* idx) const

Return the icon of the item at index “idx”.

- *int* **get_item_ID** (*int* idx) const

Return the ID of the item at index “idx”.

- void **get_item_metadata** (*int* idx) const
- *bool* **is_item_disabled** (*int* idx) const
- *int* **get_item_count** ( ) const

Return the amount of items in the OptionButton.

- void **add_separator** ( )

Add a separator to the list of items. Separators help to group items. Separator also takes up an index and is appended at the end.

- void **clear** ( )

Clear all the items in the *OptionButton*.

- void **select** (*int* idx)

Select an item by index and make it the current item.

- *int* **get_selected** ( ) const

Return the current item index

- *int* **get_selected_ID** ( ) const
- void **get_selected_metadata** ( ) const
- void **remove_item** (*int* idx)

9.174 OS

Inherits: *Object*

Category: Core

9.174.1 Brief Description

Operating System functions.

9.174.2 Member Functions

void	*set_clipboard* (*String* clipboard)

Continúa en la página siguiente

Tabla 9.17 – proviene de la página anterior

String	*get_clipboard* ( ) const
void	*set_video_mode* (*Vector2* size, *bool* fullscreen, *bool* resizable, *int* screen=0)
Vector2	*get_video_mode_size* (*int* screen=0) const
bool	*is_video_mode_fullscreen* (*int* screen=0) const
bool	*is_video_mode_resizable* (*int* screen=0) const
Array	*get_fullscreen_mode_list* (*int* screen=0) const
int	*get_screen_count* ( ) const
int	*get_current_screen* ( ) const
void	*set_current_screen* (*int* screen)
Vector2	*get_screen_position* (*int* screen=0) const
Vector2	*get_screen_size* (*int* screen=0) const
Vector2	*get_window_position* ( ) const
void	*set_window_position* (*Vector2* position)
Vector2	*get_window_size* ( ) const
void	*set_window_size* (*Vector2* size)
void	*set_window_fullscreen* (*bool* enabled)
bool	*is_window_fullscreen* ( ) const
void	*set_window_resizable* (*bool* enabled)
bool	*is_window_resizable* ( ) const
void	*set_window_minimized* (*bool* enabled)
bool	*is_window_minimized* ( ) const
void	*set_window_maximized* (*bool* enabled)
bool	*is_window_maximized* ( ) const
void	*set_screen_orientation* (*int* orientation)
int	*get_screen_orientation* ( ) const
void	*set_keep_screen_on* (*bool* enabled)
bool	*is_keep_screen_on* ( ) const
void	*set_iterations_per_second* (*int* iterations_per_second)
int	*get_iterations_per_second* ( ) const
void	*set_target_fps* (*int* target_fps)
float	*get_target_fps* ( ) const
void	*set_time_scale* (*float* time_scale)
float	*get_time_scale* ( )
bool	*has_touchscreen_ui_hint* ( ) const
void	*set_window_title* (*String* title)
void	*set_low_processor_usage_mode* (*bool* enable)
bool	*is_in_low_processor_usage_mode* ( ) const
int	*get_processor_count* ( ) const
String	*get_executable_path* ( ) const
int	*execute* (*String* path, *StringArray* arguments, *bool* blocking, *Array* output=Array())
int	*kill* (*int* pid)
int	*shell_open* (*String* uri)
int	*get_process_ID* ( ) const
String	*get_environment* (*String* environment) const
bool	*has_environment* (*String* environment) const
String	*get_name* ( ) const
StringArray	*get_cmdline_args* ( )
Object	*get_main_loop* ( ) const
Dictionary	*get_date* (*bool* utc=false) const
Dictionary	*get_time* (*bool* utc=false) const

Continúa en la página siguiente

Tabla 9.17 – proviene de la página anterior

Dictionary	*get_time_zone_info* ( ) const
int	*get_unix_time* ( ) const
int	*get_system_time_secs* ( ) const
void	*set_icon* (*Image* icon)
void	*delay_usec* (*int* usec) const
void	*delay_msec* (*int* msec) const
int	*get_ticks_msec* ( ) const
int	*get_splash_tick_msec* ( ) const
String	*get_locale* ( ) const
String	*get_model_name* ( ) const
String	*get_custom_level* ( ) const
bool	*can_draw* ( ) const
int	*get_frames_drawn* ( )
bool	*is_stdout_verbose* ( ) const
bool	*can_use_threads* ( ) const
bool	*is_debug_build* ( ) const
void	*dump_memory_to_file* (*String* file)
void	*dump_resources_to_file* (*String* file)
void	*print_resources_in_use* (*bool* short=false)
void	*print_all_resources* (*String* tofile="")
int	*get_static_memory_usage* ( ) const
int	*get_static_memory_peak_usage* ( ) const
int	*get_dynamic_memory_usage* ( ) const
String	*get_data_dir* ( ) const
String	*get_system_dir* (*int* dir) const
String	*get_unique_ID* ( ) const
bool	*is_ok_left_and_cancel_right* ( ) const
float	*get_frames_per_second* ( ) const
void	*print_all_textures_by_size* ( )
void	*print_resources_by_type* (*StringArray* types)
int	*native_video_play* (*String* path, *float* volume, *String* audio_track, *String* subtitle_track)
bool	*native_video_is_playing* ( )
void	*native_video_stop* ( )
void	*native_video_pause* ( )
void	*native_video_unpause* ( )
String	*get_scancode_string* (*int* code) const
bool	*is_scancode_unicode* (*int* code) const
int	*find_scancode_from_string* (*String* string) const
void	*set_use_file_access_save_and_swap* (*bool* enabled)
void	*alert* (*String* text, *String* title="Alert!")
int	*set_thread_name* (*String* name)

9.174.3 Numeric Constants

- **DAY_SUNDAY = 0**
- **DAY_MONDAY = 1**
- **DAY_TUESDAY = 2**
- **DAY_WEDNESDAY = 3**
- **DAY_THURSDAY = 4**

- **DAY_FRIDAY = 5**
- **DAY_SATURDAY = 6**
- **MONTH_JANUARY = 0**
- **MONTH_FEBRUARY = 1**
- **MONTH_MARCH = 2**
- **MONTH_APRIL = 3**
- **MONTH_MAY = 4**
- **MONTH_JUNE = 5**
- **MONTH_JULY = 6**
- **MONTH_AUGUST = 7**
- **MONTH_SEPTEMBER = 8**
- **MONTH_OCTOBER = 9**
- **MONTH_NOVEMBER = 10**
- **MONTH_DECEMBER = 11**
- **SCREEN_ORIENTATION_LANDSCAPE = 0**
- **SCREEN_ORIENTATION_PORTRAIT = 1**
- **SCREEN_ORIENTATION_REVERSE_LANDSCAPE = 2**
- **SCREEN_ORIENTATION_REVERSE_PORTRAIT = 3**
- **SCREEN_ORIENTATION_SENSOR_LANDSCAPE = 4**
- **SCREEN_ORIENTATION_SENSOR_PORTRAIT = 5**
- **SCREEN_ORIENTATION_SENSOR = 6**
- **SYSTEM_DIR_DESKTOP = 0**
- **SYSTEM_DIR_DCIM = 1**
- **SYSTEM_DIR_DOCUMENTS = 2**
- **SYSTEM_DIR_DOWNLOADS = 3**
- **SYSTEM_DIR_MOVIES = 4**
- **SYSTEM_DIR_MUSIC = 5**
- **SYSTEM_DIR_PICTURES = 6**
- **SYSTEM_DIR_RINGTONES = 7**

9.174.4 Description

Operating System functions. OS Wraps the most common functionality to communicate with the host Operating System, such as:

-Mouse Grabbing

-Mouse Cursors

-Clipboard

-Video Mode

-Date ” Time

-Timers

-Environment Variables

-Execution of Binaries

-Command Line

9.174.5 Member Function Description

- void **set_clipboard** (*String* clipboard)

Set clipboard to the OS.

- *String* **get_clipboard** ( ) const

Get clipboard from the host OS.

- void **set_video_mode** (*Vector2* size, *bool* fullscreen, *bool* resizable, *int* screen=0)

Change the video mode.

- *Vector2* **get_video_mode_size** (*int* screen=0) const

Return the current video mode size.

- *bool* **is_video_mode_fullscreen** (*int* screen=0) const

Return true if the current video mode is fullscreen.

- *bool* **is_video_mode_resizable** (*int* screen=0) const

Return true if the window is resizable.

- *Array* **get_fullscreen_mode_list** (*int* screen=0) const

Return the list of fullscreen modes.

- *int* **get_screen_count** ( ) const

Returns the number of displays attached to the host machine

- *int* **get_current_screen** ( ) const

Returns the current screen index (0 padded).

- void **set_current_screen** (*int* screen)
- *Vector2* **get_screen_position** (*int* screen=0) const
- *Vector2* **get_screen_size** (*int* screen=0) const

Returns the dimensions in pixels of the specified screen.

- *Vector2* **get_window_position** ( ) const

Returns the window position relative to the screen, the origin is the top left corner, +Y axis goes to the bottom and +X axis goes to the right.

- void **set_window_position** (*Vector2* position)

Sets the position of the window to the specified position (this function could be restricted by the window manager, meaning that there could be some unreachable areas of the screen).

- *Vector2* **get_window_size** ( ) const

Returns the size of the window (without counting window manager decorations).

- void **set_window_size** (*Vector2* size)

Sets the window size to the specified size.

- void **set_window_fullscreen** (*bool* enabled)

Sets window fullscreen mode to the *enabled* argument, *enabled* is a toggle for the fullscreen mode, calling the function with *enabled* true when the screen is not on fullscreen mode will cause the screen to go to fullscreen mode, calling the function with *enabled* false when the screen is in fullscreen mode will cause the window to exit the fullscreen mode.

- *bool* **is_window_fullscreen** ( ) const

Returns whether the window is in fullscreen mode or not.

- void **set_window_resizable** (*bool* enabled)

Set the window resizable state, if the window is not resizable it will preserve the dimensions specified in the project settings.

- *bool* **is_window_resizable** ( ) const

Returns whether the window is resizable or not.

- void **set_window_minimized** (*bool* enabled)

Set whether the window is minimized.

- *bool* **is_window_minimized** ( ) const

Return true if the window is minimized.

- void **set_window_maximized** (*bool* enabled)

Set the window size to maximized.

- *bool* **is_window_maximized** ( ) const

Return true if the window is maximized.

- void **set_screen_orientation** (*int* orientation)

Sets the current screen orientation, the argument value must be one of the SCREEN_ORIENTATION constants in this class.

- *int* **get_screen_orientation** ( ) const

Returns the current screen orientation, the return value will be one of the SCREEN_ORIENTATION constants in this class.

- void **set_keep_screen_on** (*bool* enabled)

Set keep screen on if true, or goes to sleep by device setting if false. (for Android/iOS)

- *bool* **is_keep_screen_on** ( ) const

Returns whether the screen is being kept on or not.

- void **set_iterations_per_second** (*int* iterations_per_second)

Set the amount of fixed iterations per second (for fixed process and physics).

- *int* **get_iterations_per_second** ( ) const

Return the amount of fixed iterations per second (for fixed process and physics).

- void **set_target_fps** (*int* target_fps)
- *float* **get_target_fps** ( ) const

- void **set_time_scale** (*float* time_scale)

Speeds up or slows down the physics by changing the delta variable. (delta * time_scale)

- *float* **get_time_scale** ( )
- *bool* **has_touchscreen_ui_hint** ( ) const
- void **set_window_title** (*String* title)

Sets the window title to the specified string.

- void **set_low_processor_usage_mode** (*bool* enable)

Set to true to enable the low cpu usage mode. In this mode, the screen only redraws when there are changes, and a considerable sleep time is inserted between frames. This way, editors using the engine UI only use very little cpu.

- *bool* **is_in_low_processor_usage_mode** ( ) const

Return true if low cpu usage mode is enabled.

- *int* **get_processor_count** ( ) const

Returns the number of cores available in the host machine.

- *String* **get_executable_path** ( ) const

Return the path to the current engine executable.

- *int* **execute** (*String* path, *StringArray* arguments, *bool* blocking, *Array* output=Array())

Execute the binary file in given path, optionally blocking until it returns. A process ID is returned.

- *int* **kill** (*int* pid)

Kill a process ID (this method can be used to kill processes that were not spawned by the game).

- *int* **shell_open** (*String* uri)
- *int* **get_process_ID** ( ) const

Returns the game process ID

- *String* **get_environment** (*String* environment) const

Return an environment variable.

- *bool* **has_environment** (*String* environment) const

Return true if an environment variable exists.

- *String* **get_name** ( ) const

Return the name of the host OS. Possible values are: “Android”, “BlackBerry 10”, “Flash”, “Haiku”, “iOS”, “HTML5”, “OSX”, “Server”, “Windows”, “WinRT”, “X11”

- *StringArray* **get_cmdline_args** ( )

Return the commandline passed to the engine.

- *Object* **get_main_loop** ( ) const

Return the main loop object (see *MainLoop*).

- *Dictionary* **get_date** (*bool* utc=false) const
- *Dictionary* **get_time** (*bool* utc=false) const
- *Dictionary* **get_time_zone_info** ( ) const
- *int* **get_unix_time** ( ) const

Return the current unix timestamp.

- *int* **get_system_time_secs** ( ) const
- void **set_icon** (*Image* icon)
- void **delay_usec** (*int* usec) const

Delay executing of the current thread by given microseconds.

- void **delay_msec** (*int* msec) const

Delay executing of the current thread by given milliseconds.

- *int* **get_ticks_msec** ( ) const

Return the amount of time passed in milliseconds since the engine started.

- *int* **get_splash_tick_msec** ( ) const
- *String* **get_locale** ( ) const

Return the host OS locale.

- *String* **get_model_name** ( ) const
- *String* **get_custom_level** ( ) const
- *bool* **can_draw** ( ) const

Return true if the host OS allows drawing.

- *int* **get_frames_drawn** ( )

Return the total amount of frames drawn.

- *bool* **is_stdout_verbose** ( ) const

Return true if the engine was executed with -v (verbose stdout).

- *bool* **can_use_threads** ( ) const
- *bool* **is_debug_build** ( ) const
- void **dump_memory_to_file** (*String* file)
- void **dump_resources_to_file** (*String* file)
- void **print_resources_in_use** (*bool* short=false)
- void **print_all_resources** (*String* tofile="")
- *int* **get_static_memory_usage** ( ) const
- *int* **get_static_memory_peak_usage** ( ) const

Return the max amount of static memory used (only works in debug).

- *int* **get_dynamic_memory_usage** ( ) const

Return the total amount of dynamic memory used (only works in debug).

- *String* **get_data_dir** ( ) const

Return the absolute directory path of user data path(user://).

- *String* **get_system_dir** (*int* dir) const
- *String* **get_unique_ID** ( ) const
- *bool* **is_ok_left_and_cancel_right** ( ) const

- *float* **get_frames_per_second** ( ) const

Returns the frames per second of the running game.

- void **print_all_textures_by_size** ( )
- void **print_resources_by_type** (*StringArray* types)
- *int* **native_video_play** (*String* path, *float* volume, *String* audio_track, *String* subtitle_track)
- *bool* **native_video_is_playing** ( )
- void **native_video_stop** ( )
- void **native_video_pause** ( )
- void **native_video_unpause** ( )
- *String* **get_scancode_string** (*int* code) const
- *bool* **is_scancode_unicode** (*int* code) const
- *int* **find_scancode_from_string** (*String* string) const
- void **set_use_file_access_save_and_swap** (*bool* enabled)
- void **alert** (*String* text, *String* title="Alert!")
- *int* **set_thread_name** (*String* name)

9.175 PackedDataContainer

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.175.1 Brief Description

9.175.2 Member Functions

Error	*pack* (var value)
int	*size* ( ) const

9.175.3 Member Function Description

- Error **pack** (var value)
- *int* **size** ( ) const

9.176 PackedDataContainerRef

Inherits: *Reference* < *Object*

Category: Core

9.176.1 Brief Description

9.176.2 Member Functions

int	*size* ( ) const

9.176.3 Member Function Description

- *int* **size** ( ) const

9.177 PackedScene

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.177.1 Brief Description

9.177.2 Member Functions

int	*pack* (*Node* path)
Node	*instance* (*bool* gen_edit_state=false) const
bool	*can_instance* ( ) const
SceneState	*get_state* ( )

9.177.3 Description

TODO: explain ownership, and that node does not need to own itself

9.177.4 Member Function Description

- *int* **pack** (*Node* path)

Pack will ignore any sub-nodes not owned by given node. See *Node.set_owner*.

- *Node* **instance** (*bool* gen_edit_state=false) const
- *bool* **can_instance** ( ) const
- *SceneState* **get_state** ( )

9.178 PacketPeer

Inherits: *Reference* < *Object*

Inherited By: *PacketPeerStream*, *PacketPeerUDP*

Category: Core

9.178.1 Brief Description

Abstraction and base class for packet-based protocols.

9.178.2 Member Functions

void	*get_var* ( ) const
int	*put_var* (Variant var)
RawArray	*get_packet* ( ) const
Error	*put_packet* (*RawArray* buffer)
Error	*get_packet_error* ( ) const
int	*get_available_packet_count* ( ) const

9.178.3 Description

PacketPeer is an abstraction and base class for packet-based protocols (such as UDP). It provides an API for sending and receiving packets both as raw data or variables. This makes it easy to transfer data over a protocol, without having to encode data as low level bytes or having to worry about network ordering.

9.178.4 Member Function Description

- void **get_var** ( ) const
- *int* **put_var** (Variant var)
- *RawArray* **get_packet** ( ) const
- Error **put_packet** (*RawArray* buffer)
- Error **get_packet_error** ( ) const
- *int* **get_available_packet_count** ( ) const

9.179 PacketPeerStream

Inherits: *PacketPeer* < *Reference* < *Object*

Category: Core

9.179.1 Brief Description

Wrapper to use a PacketPeer over a StreamPeer.

9.179.2 Member Functions

void	*set_stream_peer* (*StreamPeer* peer)

9.179.3 Description

PacketStreamPeer provides a wrapper for working using packets over a stream. This allows for using packet based code with StreamPeers. PacketPeerStream implements a custom protocol over the StreamPeer, so the user should not read or write to the wrapped StreamPeer directly.

9.179.4 Member Function Description

- void **set_stream_peer** (*StreamPeer* peer)

Set the StreamPeer object to be wrapped

9.180 PacketPeerUDP

Inherits: *PacketPeer* < *Reference* < *Object*

Category: Core

9.180.1 Brief Description

9.180.2 Member Functions

Error	*listen* (*int* port, *int* recv_buf_size=65536)
void	*close* ( )
Error	*wait* ( )
bool	*is_listening* ( ) const
String	*get_packet_ip* ( ) const
int	*get_packet_address* ( ) const
int	*get_packet_port* ( ) const
int	*set_send_address* (*String* host, *int* port)

9.180.3 Member Function Description

- Error **listen** (*int* port, *int* recv_buf_size=65536)
- void **close** ( )
- Error **wait** ( )
- *bool* **is_listening** ( ) const
- *String* **get_packet_ip** ( ) const
- *int* **get_packet_address** ( ) const
- *int* **get_packet_port** ( ) const
- *int* **set_send_address** (*String* host, *int* port)

9.181 Panel

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.181.1 Brief Description

Provides an opaque background for *Control* children.

9.181.2 Description

Panel is a *Control* that displays an opaque background. It's commonly used as a parent and container for other types of *Control* nodes.

9.182 PanelContainer

Inherits: *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.182.1 Brief Description

Panel container type.

9.182.2 Description

Panel container type. This container fits controls inside of the delimited area of a stylebox. It's useful for giving controls an outline.

9.183 ParallaxBackground

Inherits: *CanvasLayer* < *Node* < *Object*

Category: Core

9.183.1 Brief Description

9.183.2 Member Functions

void	*set_scroll_offset* (*Vector2* ofs)
Vector2	*get_scroll_offset* ( ) const
void	*set_scroll_base_offset* (*Vector2* ofs)
Vector2	*get_scroll_base_offset* ( ) const
void	*set_scroll_base_scale* (*Vector2* scale)
Vector2	*get_scroll_base_scale* ( ) const
void	*set_limit_begin* (*Vector2* ofs)
Vector2	*get_limit_begin* ( ) const
void	*set_limit_end* (*Vector2* ofs)
Vector2	*get_limit_end* ( ) const
void	*set_ignore_camera_zoom* (*bool* ignore)
bool	*is_ignore_camera_zoom* ( )

9.183.3 Member Function Description

- void **set_scroll_offset** (*Vector2* ofs)
- *Vector2* **get_scroll_offset** ( ) const
- void **set_scroll_base_offset** (*Vector2* ofs)
- *Vector2* **get_scroll_base_offset** ( ) const
- void **set_scroll_base_scale** (*Vector2* scale)
- *Vector2* **get_scroll_base_scale** ( ) const
- void **set_limit_begin** (*Vector2* ofs)
- *Vector2* **get_limit_begin** ( ) const
- void **set_limit_end** (*Vector2* ofs)
- *Vector2* **get_limit_end** ( ) const
- void **set_ignore_camera_zoom** (*bool* ignore)
- *bool* **is_ignore_camera_zoom** ( )

9.184 ParallaxLayer

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.184.1 Brief Description

9.184.2 Member Functions

void	*set_motion_scale* (*Vector2* scale)
Vector2	*get_motion_scale* ( ) const
void	*set_mirroring* (*Vector2* mirror)
Vector2	*get_mirroring* ( ) const

9.184.3 Member Function Description

- void **set_motion_scale** (*Vector2* scale)
- *Vector2* **get_motion_scale** ( ) const
- void **set_mirroring** (*Vector2* mirror)
- *Vector2* **get_mirroring** ( ) const

9.185 ParticleAttractor2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.185.1 Brief Description

9.185.2 Member Functions

void	*set_enabled* (*bool* enabled)
bool	*is_enabled* ( ) const
void	*set_radius* (*float* radius)
float	*get_radius* ( ) const
void	*set_disable_radius* (*float* radius)
float	*get_disable_radius* ( ) const
void	*set_gravity* (*float* gravity)
float	*get_gravity* ( ) const
void	*set_absorption* (*float* absorption)
float	*get_absorption* ( ) const
void	*set_particles_path* (*NodePath* path)
NodePath	*get_particles_path* ( ) const

9.185.3 Member Function Description

- void **set_enabled** (*bool* enabled)
- *bool* **is_enabled** ( ) const
- void **set_radius** (*float* radius)
- *float* **get_radius** ( ) const

- void **set_disable_radius** (*float* radius)
- *float* **get_disable_radius** ( ) const
- void **set_gravity** (*float* gravity)
- *float* **get_gravity** ( ) const
- void **set_absorption** (*float* absorption)
- *float* **get_absorption** ( ) const
- void **set_particles_path** (*NodePath* path)
- *NodePath* **get_particles_path** ( ) const

9.186 Particles

Inherits: *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.186.1 Brief Description

Particle system 3D Node

9.186.2 Member Functions

void	*set_amount* (*int* amount)
int	*get_amount* ( ) const
void	*set_emitting* (*bool* enabled)
bool	*is_emitting* ( ) const
void	*set_visibility_aabb* (*AABB* aabb)
AABB	*get_visibility_aabb* ( ) const
void	*set_emission_half_extents* (*Vector3* half_extents)
Vector3	*get_emission_half_extents* ( ) const
void	*set_emission_base_velocity* (*Vector3* base_velocity)
Vector3	*get_emission_base_velocity* ( ) const
void	*set_emission_points* (*Vector3Array* points)
Vector3Array	*get_emission_points* ( ) const
void	*set_gravity_normal* (*Vector3* normal)
Vector3	*get_gravity_normal* ( ) const
void	*set_variable* (*int* variable, *float* value)
float	*get_variable* (*int* variable) const
void	*set_randomness* (*int* variable, *float* randomness)
float	*get_randomness* (*int* variable) const
void	*set_color_phase_pos* (*int* phase, *float* pos)
float	*get_color_phase_pos* (*int* phase) const
void	*set_color_phase_color* (*int* phase, *Color* color)
Color	*get_color_phase_color* (*int* phase) const
void	*set_material* (*Material* material)
Material	*get_material* ( ) const

Continúa en la página siguiente

Tabla 9.18 – proviene de la página anterior

void	*set_emit_timeout* (*float* timeout)
float	*get_emit_timeout* ( ) const
void	*set_height_from_velocity* (*bool* enable)
bool	*has_height_from_velocity* ( ) const
void	*set_use_local_coordinates* (*bool* enable)
bool	*is_using_local_coordinates* ( ) const
void	*set_color_phases* (*int* count)
int	*get_color_phases* ( ) const

9.186.3 Numeric Constants

- **VAR_LIFETIME = 0**
- **VAR_SPREAD = 1**
- **VAR_GRAVITY = 2**
- **VAR_LINEAR_VELOCITY = 3**
- **VAR_ANGULAR_VELOCITY = 4**
- **VAR_LINEAR_ACCELERATION = 5**
- **VAR_DRAG = 6**
- **VAR_TANGENTIAL_ACCELERATION = 7**
- **VAR_INITIAL_SIZE = 9**
- **VAR_FINAL_SIZE = 10**
- **VAR_INITIAL_ANGLE = 11**
- **VAR_HEIGHT = 12**
- **VAR_HEIGHT_SPEED_SCALE = 13**
- **VAR_MAX = 14**

9.186.4 Description

Particles is a particle system 3D *Node* that is used to simulate several types of particle effects, such as explosions, rain, snow, fireflies, or other magical-like shinny sparkles. Particles are drawn using impostors, and given their dynamic behavior, the user must provide a visibility AABB (although helpers to create one automatically exist).

9.186.5 Member Function Description

- void **set_amount** (*int* amount)

Set total amount of particles in the system.

- *int* **get_amount** ( ) const

Return the total amount of particles in the system.

- void **set_emitting** (*bool* enabled)

Set the "emitting" property state. When emitting, the particle system generates new particles at constant rate.

- *bool* **is_emitting** ( ) const

Return the "emitting" property state (see *set_emitting*).

- void **set_visibility_aabb** (*AABB* aabb)

Set the visibility AABB for the particle system, since the default one will not work properly most of the time.

- *AABB* **get_visibility_aabb** ( ) const

Return the current visibility AABB.

- void **set_emission_half_extents** (*Vector3* half_extents)

Set the half extents for the emission box.

- *Vector3* **get_emission_half_extents** ( ) const

Return the half extents for the emission box.

- void **set_emission_base_velocity** (*Vector3* base_velocity)
- *Vector3* **get_emission_base_velocity** ( ) const
- void **set_emission_points** (*Vector3Array* points)
- *Vector3Array* **get_emission_points** ( ) const
- void **set_gravity_normal** (*Vector3* normal)

Set the normal vector towards where gravity is pulling (by default, negative Y).

- *Vector3* **get_gravity_normal** ( ) const

Return the normal vector towards where gravity is pulling (by default, negative Y).

- void **set_variable** (*int* variable, *float* value)

Set a specific variable for the particle system (see VAR_* enum).

- *float* **get_variable** (*int* variable) const

Return a specific variable for the particle system (see VAR_* enum).

- void **set_randomness** (*int* variable, *float* randomness)

Set the randomness for a specific variable of the particle system. Randomness produces small changes from the default each time a particle is emitted.

- *float* **get_randomness** (*int* variable) const

Return the randomness for a specific variable of the particle system. Randomness produces small changes from the default each time a particle is emitted.

- void **set_color_phase_pos** (*int* phase, *float* pos)

Set the position of a color phase (0 to 1).

- *float* **get_color_phase_pos** (*int* phase) const

Return the position of a color phase (0 to 1).

- void **set_color_phase_color** (*int* phase, *Color* color)

Set the color of a color phase.

- *Color* **get_color_phase_color** (*int* phase) const

Return the color of a color phase.

- void **set_material** (*Material* material)

Set the material used to draw particles.

- *Material* **get_material** ( ) const

Return the material used to draw particles.

- void **set_emit_timeout** (*float* timeout)
- *float* **get_emit_timeout** ( ) const
- void **set_height_from_velocity** (*bool* enable)
- *bool* **has_height_from_velocity** ( ) const
- void **set_use_local_coordinates** (*bool* enable)
- *bool* **is_using_local_coordinates** ( ) const
- void **set_color_phases** (*int* count)
- *int* **get_color_phases** ( ) const

9.187 Particles2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.187.1 Brief Description

2D Particle emitter

9.187.2 Member Functions

void	*set_emitting* (*bool* active)
bool	*is_emitting* ( ) const
void	*set_amount* (*int* amount)
int	*get_amount* ( ) const
void	*set_lifetime* (*float* lifetime)
float	*get_lifetime* ( ) const
void	*set_time_scale* (*float* time_scale)
float	*get_time_scale* ( ) const
void	*set_pre_process_time* (*float* time)
float	*get_pre_process_time* ( ) const
void	*set_emit_timeout* (*float* value)
float	*get_emit_timeout* ( ) const
void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const
void	*set_randomness* (*int* param, *float* value)
float	*get_randomness* (*int* param) const
Texture	*set_texture* (*Object* texture)
Texture	*get_texture* ( ) const
void	*set_color* (*Color* color)
Color	*get_color* ( ) const
ColorRamp	*set_color_ramp* (*Object* color_ramp)

Continúa en la página siguiente

Tabla 9.19 – proviene de la página anterior

ColorRamp	*get_color_ramp* ( ) const
void	*set_emissor_offset* (*Vector2* offset)
Vector2	*get_emissor_offset* ( ) const
void	*set_flip_h* (*bool* enable)
bool	*is_flipped_h* ( ) const
void	*set_flip_v* (*bool* enable)
bool	*is_flipped_v* ( ) const
void	*set_h_frames* (*int* enable)
int	*get_h_frames* ( ) const
void	*set_v_frames* (*int* enable)
int	*get_v_frames* ( ) const
void	*set_emission_half_extents* (*Vector2* extents)
Vector2	*get_emission_half_extents* ( ) const
void	*set_color_phases* (*int* phases)
int	*get_color_phases* ( ) const
void	*set_color_phase_color* (*int* phase, *Color* color)
Color	*get_color_phase_color* (*int* phase) const
void	*set_color_phase_pos* (*int* phase, *float* pos)
float	*get_color_phase_pos* (*int* phase) const
void	*pre_process* (*float* time)
void	*reset* ( )
void	*set_use_local_space* (*bool* enable)
bool	*is_using_local_space* ( ) const
void	*set_initial_velocity* (*Vector2* velocity)
Vector2	*get_initial_velocity* ( ) const
void	*set_explosiveness* (*float* amount)
float	*get_explosiveness* ( ) const
void	*set_emission_points* (*Vector2Array* points)
Vector2Array	*get_emission_points* ( ) const

9.187.3 Numeric Constants

- **PARAM_DIRECTION = 0** — Direction in radians at which the particles will be launched, Notice that when the direction is set to 0 the particles will be launched to the negative
- **PARAM_SPREAD = 1**
- **PARAM_LINEAR_VELOCITY = 2** — Velocity at which the particles will be launched.
- **PARAM_SPIN_VELOCITY = 3** — The speed at which particles will spin around its own center.
- **PARAM_ORBIT_VELOCITY = 4** — Velocity at which the particles will orbit around the emitter center
- **PARAM_GRAVITY_DIRECTION = 5** — Direction in radians at which the particles will be attracted
- **PARAM_GRAVITY_STRENGTH = 6** — Strength of the gravitation attraction for each particle
- **PARAM_RADIAL_ACCEL = 7**
- **PARAM_TANGENTIAL_ACCEL = 8**
- **PARAM_DAMPING = 9** — Amount of damping for each particle
- **PARAM_INITIAL_ANGLE = 10** — Initial angle in radians at which each particle will be spawned
- **PARAM_INITIAL_SIZE = 11** — Initial size of each particle

- **PARAM_FINAL_SIZE = 12** — Final size of each particle, the particle size will interpolate to this value during its lifetime.
- **PARAM_HUE_VARIATION = 13**
- **PARAM_ANIM_SPEED_SCALE = 14**
- **PARAM_ANIM_INITIAL_POS = 15**
- **PARAM_MAX = 16**
- **MAX_COLOR_PHASES = 4**

9.187.4 Description

Particles2D is a particle system 2D *Node* that is used to simulate several types of particle effects, such as explosions, rain, snow, fireflies, or other magical-like shinny sparkles. Particles are drawn using impostors, and given their dynamic behavior, the user must provide a visibility AABB (although helpers to create one automatically exist).

9.187.5 Member Function Description

- void **set_emitting** (*bool* active)

If this is set to true then the particle emitter will emit particles, if its false it will not.

- *bool* **is_emitting** ( ) const

Returns whether this emitter is currently emitting or not

- void **set_amount** (*int* amount)

Sets the amount of particles spawned at each emission

- *int* **get_amount** ( ) const

Returns the amount of particles spawned at each emission

- void **set_lifetime** (*float* lifetime)

Sets the amount of seconds that each particle will be visible.

- *float* **get_lifetime** ( ) const

Gets the amount of seconds that each particle will be visible.

- void **set_time_scale** (*float* time_scale)

Sets the increment or decrement for the particle lifetime. for example: if the time scale is set to 2, the particles will die and move twice as fast.

- *float* **get_time_scale** ( ) const

Returns the emitter time scale

- void **set_pre_process_time** (*float* time)
- *float* **get_pre_process_time** ( ) const
- void **set_emit_timeout** (*float* value)

Sets the amount of seconds during which the emitter will spawn particles, after the specified seconds the emitter state will be set to non emitting, so calling *is_emitting* will return false. If the timeout is 0 the emitter will spawn forever.

- *float* **get_emit_timeout** ( ) const

Returns the amount of seconds during which the emitter will spawn particles

- void **set_param** (*int* param, *float* value)

Sets the value of the specified emitter parameter (see the constants secction for the list of parameters)

- *float* **get_param** (*int* param) const

Returns the value of the specified emitter parameter

- void **set_randomness** (*int* param, *float* value)

Sets the randomness value of the specified emitter parameter (see the constants secction for the list of parameters), 0 means no randomness, so every particle will have the parameters specified, 1 means that the parameter will be choosen at random, the closer the randomness value gets to 0 the more conservative the variation of the parameter will be.

- *float* **get_randomness** (*int* param) const

Returns the randomness value of the specified emitter parameter

- *Texture* **set_texture** (*Object* texture)

Sets the texture for each particle

- *Texture* **get_texture** ( ) const

Returns the texture for emitted particles

- void **set_color** (*Color* color)

Set the tint color for each particle.

- *Color* **get_color** ( ) const

Returns the tint color for each particle.

- *ColorRamp* **set_color_ramp** (*Object* color_ramp)

Sets the *ColorRamp* used to tint each particle. Particle will be tinted according to their lifetimes.

- *ColorRamp* **get_color_ramp** ( ) const

Returns the *ColorRamp* used to tint each particle

- void **set_emissor_offset** (*Vector2* offset)

Sets the particle spawn origin position relative to the emitter center. for example if this value is set to (50, 50), the particle will spawn 50 units to the right and 50 units to the bottom of the emitter center.

- *Vector2* **get_emissor_offset** ( ) const

Returns the particle spawn origin position relative to the emitter.

- void **set_flip_h** (*bool* enable)
- *bool* **is_flipped_h** ( ) const
- void **set_flip_v** (*bool* enable)
- *bool* **is_flipped_v** ( ) const
- void **set_h_frames** (*int* enable)
- *int* **get_h_frames** ( ) const
- void **set_v_frames** (*int* enable)
- *int* **get_v_frames** ( ) const
- void **set_emission_half_extents** (*Vector2* extents)

Sets the half extents of the emission box, particles will be spawned at random inside this box.

- *Vector2* **get_emission_half_extents** ( ) const

Returns the half extents of the emission box.

- void **set_color_phases** (*int* phases)
- *int* **get_color_phases** ( ) const
- void **set_color_phase_color** (*int* phase, *Color* color)
- *Color* **get_color_phase_color** (*int* phase) const
- void **set_color_phase_pos** (*int* phase, *float* pos)
- *float* **get_color_phase_pos** (*int* phase) const
- void **pre_process** (*float* time)
- void **reset** ( )
- void **set_use_local_space** (*bool* enable)
- *bool* **is_using_local_space** ( ) const
- void **set_initial_velocity** (*Vector2* velocity)
- *Vector2* **get_initial_velocity** ( ) const
- void **set_explosiveness** (*float* amount)
- *float* **get_explosiveness** ( ) const
- void **set_emission_points** (*Vector2Array* points)
- *Vector2Array* **get_emission_points** ( ) const

9.188 Patch9Frame

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.188.1 Brief Description

9.188.2 Member Functions

void	*set_texture* (*Object* texture)
Object	*get_texture* ( ) const
void	*set_modulate* (*Color* modulate)
Color	*get_modulate* ( ) const
void	*set_patch_margin* (*int* margin, *int* value)
int	*get_patch_margin* (*int* margin) const
void	*set_draw_center* (*bool* draw_center)
bool	*get_draw_center* ( ) const

9.188.3 Member Function Description

- void **set_texture** (*Object* texture)
- *Object* **get_texture** ( ) const
- void **set_modulate** (*Color* modulate)
- *Color* **get_modulate** ( ) const
- void **set_patch_margin** (*int* margin, *int* value)
- *int* **get_patch_margin** (*int* margin) const
- void **set_draw_center** (*bool* draw_center)
- *bool* **get_draw_center** ( ) const

9.189 Path

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.189.1 Brief Description

Container for a *Curve3D*.

9.189.2 Member Functions

void	*set_curve* (*Curve3D* curve)
Curve3D	*get_curve* ( ) const

9.189.3 Description

This class is a container/Node-ification of a *Curve3D*, so it can have *Spatial* properties and *Node* info.

9.189.4 Member Function Description

- void **set_curve** (*Curve3D* curve)

Sets the *Curve3D*.

- *Curve3D* **get_curve** ( ) const

Returns the *Curve3D* contained.

9.190 Path2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.190.1 Brief Description

Container for a *Curve2D*.

9.190.2 Member Functions

void	*set_curve* (*Curve2D* curve)
Curve2D	*get_curve* ( ) const

9.190.3 Description

This class is a container/Node-ification of a *Curve2D*, so it can have *Node2D* properties and *Node* info.

9.190.4 Member Function Description

- void **set_curve** (*Curve2D* curve)

Sets the *Curve2D*.

- *Curve2D* **get_curve** ( ) const

Returns the *Curve2D* contained.

9.191 PathFollow

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.191.1 Brief Description

Point sampler for a *Path*.

9.191.2 Member Functions

void	*set_offset* (*float* offset)
float	*get_offset* ( ) const
void	*set_h_offset* (*float* h_offset)
float	*get_h_offset* ( ) const
void	*set_v_offset* (*float* v_offset)
float	*get_v_offset* ( ) const
void	*set_unit_offset* (*float* unit_offset)
float	*get_unit_offset* ( ) const
void	*set_rotation_mode* (*int* rotation_mode)
int	*get_rotation_mode* ( ) const
void	*set_cubic_interpolation* (*bool* enable)
bool	*get_cubic_interpolation* ( ) const
void	*set_loop* (*bool* loop)
bool	*has_loop* ( ) const

9.191.3 Numeric Constants

- **ROTATION_NONE = 0** — Forbids the PathFollow to rotate.
- **ROTATION_Y = 1** — Allows the PathFollow to rotate in the Y axis only.
- **ROTATION_XY = 2** — Allows the PathFollow to rotate in both the X, and Y axes.
- **ROTATION_XYZ = 3** — Allows the PathFollow to rotate in any axis.

9.191.4 Description

This node takes its parent *Path*, and returns the coordinates of a point within it, given a distance from the first vertex.

It is useful for making other nodes follow a path, without coding the movement pattern. For that, the nodes must be descendants of this node. Then, when setting an offset in this node, the descendant nodes will move accordingly.

9.191.5 Member Function Description

- void **set_offset** (*float* offset)

Sets the distance from the first vertex, measured in 3D units along the path. This sets this node's position to a point within the path.

- *float* **get_offset** ( ) const

Returns the distance along the path in 3D units.

- void **set_h_offset** (*float* h_offset)

Moves this node in the X axis. As this node's position will be set every time its offset is set, this allows many PathFollow to share the same curve (and thus the same movement pattern), yet not return the same position for a given path offset.

A similar effect may be achieved moving the this node's descendants.

- *float* **get_h_offset** ( ) const

Returns the X displacement this node has from its parent *Path*.

- void **set_v_offset** (*float* v_offset)

Moves this node in the Y axis, for the same reasons of *set_h_offset*.

- *float* **get_v_offset** ( ) const

Returns the Y displacement this node has from its parent *Path*.

- void **set_unit_offset** (*float* unit_offset)

Sets the distance from the first vertex, considering 0.0 as the first vertex and 1.0 as the last. This is just another way of expressing the offset within the path, as the offset supplied is multiplied internally by the path's length.

- *float* **get_unit_offset** ( ) const

Returns the distance along the path as a number in the range 0.0 (for the first vertex) to 1.0 (for the last).

- void **set_rotation_mode** (*int* rotation_mode)

Allows or forbids rotation on one or more axes, per the constants below.

- *int* **get_rotation_mode** ( ) const

Returns the rotation mode. The constants below list which axes are allowed to rotate for each mode.

- void **set_cubic_interpolation** (*bool* enable)

The points along the *Curve3D* of the *Path* are precomputed before use, for faster calculations. The point at the requested offset is then calculated interpolating between two adjacent cached points. This may present a problem if the curve makes sharp turns, as the cached points may not follow the curve closely enough.

There are two answers to this problem: Either increase the number of cached points and increase memory consumption, or make a cubic interpolation between two points at the cost of (slightly) slower calculations.

This method controls whether the position between two cached points is interpolated linearly, or cubicly.

- *bool* **get_cubic_interpolation** ( ) const

This method returns whether the position between two cached points (see *set_cubic_interpolation*) is interpolated linearly, or cubicly.

- void **set_loop** (*bool* loop)

If set, any offset outside the path's length (whether set by *set_offset* or *set_unit_offset* will wrap around, instead of stopping at the ends. Set it for cyclic paths.

- *bool* **has_loop** ( ) const

Returns whether this node wraps its offsets around, or truncates them to the path ends.

9.192 PathFollow2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.192.1 Brief Description

Point sampler for a *Path2D*.

9.192.2 Member Functions

void	*set_offset* (*float* offset)
float	*get_offset* ( ) const
void	*set_h_offset* (*float* h_offset)
float	*get_h_offset* ( ) const
void	*set_v_offset* (*float* v_offset)
float	*get_v_offset* ( ) const
void	*set_unit_offset* (*float* unit_offset)
float	*get_unit_offset* ( ) const
void	*set_rotate* (*bool* enable)
bool	*is_rotating* ( ) const
void	*set_cubic_interpolation* (*bool* enable)
bool	*get_cubic_interpolation* ( ) const
void	*set_loop* (*bool* loop)
bool	*has_loop* ( ) const

9.192.3 Description

This node takes its parent *Path2D*, and returns the coordinates of a point within it, given a distance from the first vertex.

It is useful for making other nodes follow a path, without coding the movement pattern. For that, the nodes must be descendants of this node. Then, when setting an offset in this node, the descendant nodes will move accordingly.

9.192.4 Member Function Description

- void **set_offset** (*float* offset)

Sets the distance from the first vertex, measured in pixels along the path. This sets this node's position to a point within the path.

- *float* **get_offset** ( ) const

Returns the distance along the path in pixels.

- void **set_h_offset** (*float* h_offset)

Moves this node horizontally. As this node's position will be set every time its offset is set, this allows many PathFollow2D to share the same curve (and thus the same movement pattern), yet not return the same position for a given path offset.

A similar effect may be achieved moving this node's descendants.

- *float* **get_h_offset** ( ) const

Returns the horizontal displacement this node has from its parent *Path2D*.

- void **set_v_offset** (*float* v_offset)

Moves the PathFollow2D vertically, for the same reasons of *set_h_offset*.

- *float* **get_v_offset** ( ) const

Returns the vertical displacement this node has from its parent *Path2D*.

- void **set_unit_offset** (*float* unit_offset)

Sets the distance from the first vertex, considering 0.0 as the first vertex and 1.0 as the last. This is just another way of expressing the offset within the path, as the offset supplied is multiplied internally by the path's length.

- *float* **get_unit_offset** ( ) const

Returns the distance along the path as a number in the range 0.0 (for the first vertex) to 1.0 (for the last).

- void **set_rotate** (*bool* enable)

If set, this node rotates to follow the path, making its descendants rotate.

- *bool* **is_rotating** ( ) const

Returns whether this node rotates to follow the path.

- void **set_cubic_interpolation** (*bool* enable)

The points along the *Curve2D* of the *Path2D* are precomputed before use, for faster calculations. The point at the requested offset is then calculated interpolating between two adjacent cached points. This may present a problem if the curve makes sharp turns, as the cached points may not follow the curve closely enough.

There are two answers to this problem: Either increase the number of cached points and increase memory consumption, or make a cubic interpolation between two points at the cost of (slightly) slower calculations.

This method controls whether the position between two cached points is interpolated linearly, or cubicly.

- *bool* **get_cubic_interpolation** ( ) const

This method returns whether the position between two cached points (see *set_cubic_interpolation*) is interpolated linearly, or cubicly.

- void **set_loop** (*bool* loop)

If set, any offset outside the path's length (whether set by *set_offset* or *set_unit_offset* will wrap around, instead of stopping at the ends. Set it for cyclic paths.

- *bool* **has_loop** ( ) const

Returns whether this node wraps its offsets around, or truncates them to the path ends.

9.193 PathRemap

Inherits: *Object*

Category: Core

9.193.1 Brief Description

Singleton containing the list of remapped resources.

9.193.2 Member Functions

void	*add_remap* (*String* from, *String* to, *String* locale="")
bool	*has_remap* (*String* path) const
String	*get_remap* (*String* path) const
void	*erase_remap* (*String* path)
void	*clear_remaps* ( )

9.193.3 Description

When exporting, the types of some resources may change internally so they are converted to more optimized versions. While it's not usually necessary to access to this directly (path remapping happens automatically when opening a file), it's exported just for information.

9.193.4 Member Function Description

- void **add_remap** (*String* from, *String* to, *String* locale="")

Add a remap from a file to another.

- *bool* **has_remap** (*String* path) const

Return true if a file is being remapped.

- *String* **get_remap** (*String* path) const

Return the remapped new path of a file.

- void **erase_remap** (*String* path)

Erase a remap.

- void **clear_remaps** ( )

Clear all remaps.

9.194 PCKPacker

Inherits: *Reference* < *Object*

Category: Core

9.194.1 Brief Description

9.194.2 Member Functions

int	*pck_start* (*String* pck_name, *int* alignment)
int	*add_file* (*String* pck_path, *String* source_path)
int	*flush* (*bool* verbose)

9.194.3 Member Function Description

- *int* **pck_start** (*String* pck_name, *int* alignment)
- *int* **add_file** (*String* pck_path, *String* source_path)
- *int* **flush** (*bool* verbose)

9.195 Performance

Inherits: *Object*

Category: Core

9.195.1 Brief Description

9.195.2 Member Functions

float	*get_monitor* (*int* monitor) const

9.195.3 Numeric Constants

- **TIME_FPS = 0**
- **TIME_PROCESS = 1**
- **TIME_FIXED_PROCESS = 2**
- **MEMORY_STATIC = 3**
- **MEMORY_DYNAMIC = 4**
- **MEMORY_STATIC_MAX = 5**

- **MEMORY_DYNAMIC_MAX = 6**
- **MEMORY_MESSAGE_BUFFER_MAX = 7**
- **OBJECT_COUNT = 8**
- **OBJECT_RESOURCE_COUNT = 9**
- **OBJECT_NODE_COUNT = 10**
- **RENDER_OBJECTS_IN_FRAME = 11**
- **RENDER_VERTICES_IN_FRAME = 12**
- **RENDER_MATERIAL_CHANGES_IN_FRAME = 13**
- **RENDER_SHADER_CHANGES_IN_FRAME = 14**
- **RENDER_SURFACE_CHANGES_IN_FRAME = 15**
- **RENDER_DRAW_CALLS_IN_FRAME = 16**
- **RENDER_USAGE_VIDEO_MEM_TOTAL = 20**
- **RENDER_VIDEO_MEM_USED = 17**
- **RENDER_TEXTURE_MEM_USED = 18**
- **RENDER_VERTEX_MEM_USED = 19**
- **PHYSICS_2D_ACTIVE_OBJECTS = 21**
- **PHYSICS_2D_COLLISION_PAIRS = 22**
- **PHYSICS_2D_ISLAND_COUNT = 23**
- **PHYSICS_3D_ACTIVE_OBJECTS = 24**
- **PHYSICS_3D_COLLISION_PAIRS = 25**
- **PHYSICS_3D_ISLAND_COUNT = 26**
- **MONITOR_MAX = 27**

9.195.4 Member Function Description

- *float* **get_monitor** (*int* monitor) const

9.196 PHashTranslation

Inherits: *Translation* **<** *Resource* **<** *Reference* **<** *Object*

Category: Core

9.196.1 Brief Description

Optimized translation.

9.196.2 Member Functions

void	*generate* (*Translation* from)

9.196.3 Description

Optimized translation. Uses real-time compressed translations, which results in very small dictionaries.

9.196.4 Member Function Description

- void **generate** (*Translation* from)

9.197 Physics2DDirectBodyState

Inherits: *Object*

Inherited By: *Physics2DDirectBodyStateSW*

Category: Core

9.197.1 Brief Description

Direct access object to a physics body in the *Physics2DServer*.

9.197.2 Member Functions

Vector2	*get_total_gravity* ( ) const
float	*get_total_linear_damp* ( ) const
float	*get_total_angular_damp* ( ) const
float	*get_inverse_mass* ( ) const
float	*get_inverse_inertia* ( ) const
void	*set_linear_velocity* (*Vector2* velocity)
Vector2	*get_linear_velocity* ( ) const
void	*set_angular_velocity* (*float* velocity)
float	*get_angular_velocity* ( ) const
void	*set_transform* (*Matrix32* transform)
Matrix32	*get_transform* ( ) const
void	*set_sleep_state* (*bool* enabled)
bool	*is_sleeping* ( ) const
int	*get_contact_count* ( ) const
Vector2	*get_contact_local_pos* (*int* contact_idx) const
Vector2	*get_contact_local_normal* (*int* contact_idx) const
int	*get_contact_local_shape* (*int* contact_idx) const
RID	*get_contact_collider* (*int* contact_idx) const
Vector2	*get_contact_collider_pos* (*int* contact_idx) const
int	*get_contact_collider_id* (*int* contact_idx) const
Object	*get_contact_collider_object* (*int* contact_idx) const
int	*get_contact_collider_shape* (*int* contact_idx) const
Variant	*get_contact_collider_shape_metadata* (*int* contact_idx) const
Vector2	*get_contact_collider_velocity_at_pos* (*int* contact_idx) const
float	*get_step* ( ) const
void	*integrate_forces* ( )
Physics2DDirectSpaceState	*get_space_state* ( )

9.197.3 Description

Direct access object to a physics body in the *Physics2DServer*. This object is passed via the direct state callback of rigid/character bodies, and is intended for changing the direct state of that body.

9.197.4 Member Function Description

- *Vector2* **get_total_gravity** ( ) const

Return the total gravity vector being currently applied to this body.

- *float* **get_total_linear_damp** ( ) const
- *float* **get_total_angular_damp** ( ) const
- *float* **get_inverse_mass** ( ) const

Return the inverse of the mass of the body.

- *float* **get_inverse_inertia** ( ) const

Return the inverse of the inertia of the body.

- void **set_linear_velocity** (*Vector2* velocity)

Change the linear velocity of the body.

- *Vector2* **get_linear_velocity** ( ) const

Return the current linear velocity of the body.

- void **set_angular_velocity** (*float* velocity)

Change the angular velocity of the body.

- *float* **get_angular_velocity** ( ) const

Return the angular velocity of the body.

- void **set_transform** (*Matrix32* transform)

Change the transform matrix of the body.

- *Matrix32* **get_transform** ( ) const

Return the transform matrix of the body.

- void **set_sleep_state** (*bool* enabled)

Set the sleeping state of the body, only affects character/rigid bodies.

- *bool* **is_sleeping** ( ) const

Return true if this body is currently sleeping (not active).

- *int* **get_contact_count** ( ) const

Return the amount of contacts this body has with other bodies. Note that by default this returns 0 unless bodies are configured to log contacts.

- *Vector2* **get_contact_local_pos** (*int* contact_idx) const

Return the local position (of this body) of the contact point.

- *Vector2* **get_contact_local_normal** (*int* contact_idx) const
- *int* **get_contact_local_shape** (*int* contact_idx) const

Return the local shape index of the collision.

- *RID* **get_contact_collider** (*int* contact_idx) const

Return the RID of the collider.

- *Vector2* **get_contact_collider_pos** (*int* contact_idx) const

Return the contact position in the collider.

- *int* **get_contact_collider_id** (*int* contact_idx) const

Return the object id of the collider.

- *Object* **get_contact_collider_object** (*int* contact_idx) const

Return the collider object, this depends on how it was created (will return a scene node if such was used to create it).

- *int* **get_contact_collider_shape** (*int* contact_idx) const

Return the collider shape index.

- Variant **get_contact_collider_shape_metadata** (*int* contact_idx) const
- *Vector2* **get_contact_collider_velocity_at_pos** (*int* contact_idx) const

Return the linear velocity vector at contact point of the collider.

- *float* **get_step** ( ) const

Return the timestep (delta) used for the simulation.

- void **integrate_forces** ( )

Call the built-in force integration code.

- *Physics2DDirectSpaceState* **get_space_state** ( )

Return the current state of space, useful for queries.

9.198 Physics2DDirectBodyStateSW

Inherits: *Physics2DDirectBodyState* < *Object*

Category: Core

9.198.1 Brief Description

9.199 Physics2DDirectSpaceState

Inherits: *Object*

Category: Core

9.199.1 Brief Description

Direct access object to a space in the *Physics2DServer*.

9.199.2 Member Functions

Array	*intersect_point* (*Vector2* point, *int* max_results=32, *Array* exclude=Array(), *int* layer_mask=2147483647, *int* type_mask=15)
Dictionary	*intersect_ray* (*Vector2* from, *Vector2* to, *Array* exclude=Array(), *int* layer_mask=2147483647, *int* type_mask=15)
Array	*intersect_shape* (*Physics2DShapeQueryParameters* shape, *int* max_results=32)
Array	*cast_motion* (*Physics2DShapeQueryParameters* shape)
Array	*collide_shape* (*Physics2DShapeQueryParameters* shape, *int* max_results=32)
Dictionary	*get_rest_info* (*Physics2DShapeQueryParameters* shape)

9.199.3 Numeric Constants

- **TYPE_MASK_STATIC_BODY = 1**
- **TYPE_MASK_KINEMATIC_BODY = 2**
- **TYPE_MASK_RIGID_BODY = 4**
- **TYPE_MASK_CHARACTER_BODY = 8**
- **TYPE_MASK_AREA = 16**
- **TYPE_MASK_COLLISION = 15**

9.199.4 Description

Direct access object to a space in the *Physics2DServer*. It's used mainly to do queries against objects and areas residing in a given space.

9.199.5 Member Function Description

- *Array* **intersect_point** (*Vector2* point, *int* max_results=32, *Array* exclude=Array(), *int* layer_mask=2147483647, *int* type_mask=15)
- *Dictionary* **intersect_ray** (*Vector2* from, *Vector2* to, *Array* exclude=Array(), *int* layer_mask=2147483647, *int* type_mask=15)

Intersect a ray in a given space, the returned object is a dictionary with the following fields:

position: place where ray is stopped.

normal: normal of the object at the point where the ray was stopped.

shape: shape index of the object against which the ray was stopped.

collider_: collider against which the ray was stopped.

collider_id: collider id of the object against which the ray was stopped.

collider: collider object against which the ray was stopped.

rid: *RID* of the object against which the ray was stopped.

If the ray did not intersect anything, then an empty dictionary (dir.empty()==true) is returned instead.

- *Array* **intersect_shape** (*Physics2DShapeQueryParameters* shape, *int* max_results=32)

Intersect a given shape (RID or *Shape2D*) against the space, the intersected shapes are returned in a special result object.

- *Array* **cast_motion** (*Physics2DShapeQueryParameters* shape)
- *Array* **collide_shape** (*Physics2DShapeQueryParameters* shape, *int* max_results=32)
- *Dictionary* **get_rest_info** (*Physics2DShapeQueryParameters* shape)

9.200 Physics2DServer

Inherits: *Object*

Inherited By: *Physics2DServerSW*

Category: Core

9.200.1 Brief Description

Physics 2D Server.

9.200.2 Member Functions

RID	*shape_create* (*int* type)
void	*shape_set_data* (*RID* shape, var data)
int	*shape_get_type* (*RID* shape) const
void	*shape_get_data* (*RID* shape) const
RID	*space_create* ( )
void	*space_set_active* (*RID* space, *bool* active)
bool	*space_is_active* (*RID* space) const
void	*space_set_param* (*RID* space, *int* param, *float* value)
float	*space_get_param* (*RID* space, *int* param) const
Physics2DDirectSpaceState	*space_get_direct_state* (*RID* space)
RID	*area_create* ( )
void	*area_set_space* (*RID* area, *RID* space)
RID	*area_get_space* (*RID* area) const
void	*area_set_space_override_mode* (*RID* area, *int* mode)
int	*area_get_space_override_mode* (*RID* area) const
void	*area_add_shape* (*RID* area, *RID* shape, *Matrix32* transform=1,0, 0,1, 0,0)
void	*area_set_shape* (*RID* area, *int* shape_idx, *RID* shape)
void	*area_set_shape_transform* (*RID* area, *int* shape_idx, *Matrix32* transform)
int	*area_get_shape_count* (*RID* area) const
RID	*area_get_shape* (*RID* area, *int* shape_idx) const
Matrix32	*area_get_shape_transform* (*RID* area, *int* shape_idx) const
void	*area_remove_shape* (*RID* area, *int* shape_idx)
void	*area_clear_shapes* (*RID* area)
void	*area_set_layer_mask* (*RID* area, *int* mask)
void	*area_set_collision_mask* (*RID* area, *int* mask)
void	*area_set_param* (*RID* area, *int* param, var value)
void	*area_set_transform* (*RID* area, *Matrix32* transform)

Continúa en l

Tabla 9.20 – proviene de la página anterior

void	*area_get_param* (*RID* area, *int* param) const
Matrix32	*area_get_transform* (*RID* area) const
void	*area_attach_object_instance_ID* (*RID* area, *int* id)
int	*area_get_object_instance_ID* (*RID* area) const
void	*area_set_monitor_callback* (*RID* area, *Object* receiver, *String* method)
RID	*body_create* (*int* mode=2, *bool* init_sleeping=false)
void	*body_set_space* (*RID* body, *RID* space)
RID	*body_get_space* (*RID* body) const
void	*body_set_mode* (*RID* body, *int* mode)
int	*body_get_mode* (*RID* body) const
void	*body_add_shape* (*RID* body, *RID* shape, *Matrix32* transform=1,0, 0,1, 0,0)
void	*body_set_shape* (*RID* body, *int* shape_idx, *RID* shape)
void	*body_set_shape_transform* (*RID* body, *int* shape_idx, *Matrix32* transform)
void	*body_set_shape_metadata* (*RID* body, *int* shape_idx, var metadata)
int	*body_get_shape_count* (*RID* body) const
RID	*body_get_shape* (*RID* body, *int* shape_idx) const
Matrix32	*body_get_shape_transform* (*RID* body, *int* shape_idx) const
void	*body_get_shape_metadata* (*RID* body, *int* shape_idx) const
void	*body_remove_shape* (*RID* body, *int* shape_idx)
void	*body_clear_shapes* (*RID* body)
void	*body_set_shape_as_trigger* (*RID* body, *int* shape_idx, *bool* enable)
bool	*body_is_shape_set_as_trigger* (*RID* body, *int* shape_idx) const
void	*body_attach_object_instance_ID* (*RID* body, *int* id)
int	*body_get_object_instance_ID* (*RID* body) const
void	*body_set_continuous_collision_detection_mode* (*RID* body, *int* mode)
int	*body_get_continuous_collision_detection_mode* (*RID* body) const
void	*body_set_layer_mask* (*RID* body, *int* mask)
int	*body_get_layer_mask* (*RID* body) const
void	*body_set_collision_mask* (*RID* body, *int* mask)
int	*body_get_collision_mask* (*RID* body) const
void	*body_set_param* (*RID* body, *int* param, *float* value)
float	*body_get_param* (*RID* body, *int* param) const
void	*body_set_state* (*RID* body, *int* state, var value)
void	*body_get_state* (*RID* body, *int* state) const
void	*body_apply_impulse* (*RID* body, *Vector2* pos, *Vector2* impulse)
void	*body_set_axis_velocity* (*RID* body, *Vector2* axis_velocity)
void	*body_add_collision_exception* (*RID* body, *RID* excepted_body)
void	*body_remove_collision_exception* (*RID* body, *RID* excepted_body)
void	*body_set_max_contacts_reported* (*RID* body, *int* amount)
int	*body_get_max_contacts_reported* (*RID* body) const
void	*body_set_one_way_collision_direction* (*RID* body, *Vector2* normal)
Vector2	*body_get_one_way_collision_direction* (*RID* body) const
void	*body_set_one_way_collision_max_depth* (*RID* body, *float* depth)
float	*body_get_one_way_collision_max_depth* (*RID* body) const
void	*body_set_omit_force_integration* (*RID* body, *bool* enable)
bool	*body_is_omitting_force_integration* (*RID* body) const
void	*body_set_force_integration_callback* (*RID* body, *Object* receiver, *String* method, var userdata=NULL
bool	*body_test_motion* (*RID* body, *Vector2* motion, *float* margin=0.08, *Physics2DTestMotionResult* result=N
void	*joint_set_param* (*RID* joint, *int* param, *float* value)
float	*joint_get_param* (*RID* joint, *int* param) const

Continúa en la

Tabla 9.20 – proviene de la página anterior

RID	*pin_joint_create* (*Vector2* anchor, *RID* body_a, *RID* body_b=RID())
RID	*groove_joint_create* (*Vector2* groove1_a, *Vector2* groove2_a, *Vector2* anchor_b, *RID* body_a=RID(), *R*
RID	*damped_spring_joint_create* (*Vector2* anchor_a, *Vector2* anchor_b, *RID* body_a, *RID* body_b=RID())
void	*damped_string_joint_set_param* (*RID* joint, *int* param, *float* value)
float	*damped_string_joint_get_param* (*RID* joint, *int* param) const
int	*joint_get_type* (*RID* joint) const
void	*free_rid* (*RID* rid)
void	*set_active* (*bool* active)
int	*get_process_info* (*int* process_info)

9.200.3 Numeric Constants

- **SHAPE_LINE = 0**
- **SHAPE_SEGMENT = 2**
- **SHAPE_CIRCLE = 3**
- **SHAPE_RECTANGLE = 4**
- **SHAPE_CAPSULE = 5**
- **SHAPE_CONVEX_POLYGON = 6**
- **SHAPE_CONCAVE_POLYGON = 7**
- **SHAPE_CUSTOM = 8**
- **AREA_PARAM_GRAVITY = 0**
- **AREA_PARAM_GRAVITY_VECTOR = 1**
- **AREA_PARAM_GRAVITY_IS_POINT = 2**
- **AREA_PARAM_GRAVITY_DISTANCE_SCALE = 3**
- **AREA_PARAM_GRAVITY_POINT_ATTENUATION = 4**
- **AREA_PARAM_LINEAR_DAMP = 5**
- **AREA_PARAM_ANGULAR_DAMP = 6**
- **AREA_PARAM_PRIORITY = 7**
- **AREA_SPACE_OVERRIDE_DISABLED = 0** — This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.
- **AREA_SPACE_OVERRIDE_COMBINE = 1** — This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.
- **AREA_SPACE_OVERRIDE_COMBINE_REPLACE = 2** — This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.
- **AREA_SPACE_OVERRIDE_REPLACE = 3** — This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.
- **AREA_SPACE_OVERRIDE_REPLACE_COMBINE = 4** — This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.
- **BODY_MODE_STATIC = 0**
- **BODY_MODE_KINEMATIC = 1**

- **BODY_MODE_RIGID = 2**
- **BODY_MODE_CHARACTER = 3**
- **BODY_PARAM_BOUNCE = 0**
- **BODY_PARAM_FRICTION = 1**
- **BODY_PARAM_MASS = 2**
- **BODY_PARAM_GRAVITY_SCALE = 3**
- **BODY_PARAM_LINEAR_DAMP = 4**
- **BODY_PARAM_ANGULAR_DAMP = 5**
- **BODY_PARAM_MAX = 6**
- **BODY_STATE_TRANSFORM = 0**
- **BODY_STATE_LINEAR_VELOCITY = 1**
- **BODY_STATE_ANGULAR_VELOCITY = 2**
- **BODY_STATE_SLEEPING = 3**
- **BODY_STATE_CAN_SLEEP = 4**
- **JOINT_PIN = 0**
- **JOINT_GROOVE = 1**
- **JOINT_DAMPED_SPRING = 2**
- **DAMPED_STRING_REST_LENGTH = 0**
- **DAMPED_STRING_STIFFNESS = 1**
- **DAMPED_STRING_DAMPING = 2**
- **CCD_MODE_DISABLED = 0**
- **CCD_MODE_CAST_RAY = 1**
- **CCD_MODE_CAST_SHAPE = 2**
- **AREA_BODY_ADDED = 0**
- **AREA_BODY_REMOVED = 1**
- **INFO_ACTIVE_OBJECTS = 0**
- **INFO_COLLISION_PAIRS = 1**
- **INFO_ISLAND_COUNT = 2**

9.200.4 Description

Physics 2D Server is the server responsible for all 2D physics.

9.200.5 Member Function Description

- *RID* **shape_create** (*int* type)
- void **shape_set_data** (*RID* shape, var data)
- *int* **shape_get_type** (*RID* shape) const

- void **shape_get_data** (*RID* shape) const
- *RID* **space_create** ( )
- void **space_set_active** (*RID* space, *bool* active)
- *bool* **space_is_active** (*RID* space) const
- void **space_set_param** (*RID* space, *int* param, *float* value)
- *float* **space_get_param** (*RID* space, *int* param) const
- *Physics2DDirectSpaceState* **space_get_direct_state** (*RID* space)
- *RID* **area_create** ( )
- void **area_set_space** (*RID* area, *RID* space)
- *RID* **area_get_space** (*RID* area) const
- void **area_set_space_override_mode** (*RID* area, *int* mode)
- *int* **area_get_space_override_mode** (*RID* area) const
- void **area_add_shape** (*RID* area, *RID* shape, *Matrix32* transform=1,0, 0,1, 0,0)
- void **area_set_shape** (*RID* area, *int* shape_idx, *RID* shape)
- void **area_set_shape_transform** (*RID* area, *int* shape_idx, *Matrix32* transform)
- *int* **area_get_shape_count** (*RID* area) const
- *RID* **area_get_shape** (*RID* area, *int* shape_idx) const
- *Matrix32* **area_get_shape_transform** (*RID* area, *int* shape_idx) const
- void **area_remove_shape** (*RID* area, *int* shape_idx)
- void **area_clear_shapes** (*RID* area)
- void **area_set_layer_mask** (*RID* area, *int* mask)
- void **area_set_collision_mask** (*RID* area, *int* mask)
- void **area_set_param** (*RID* area, *int* param, var value)
- void **area_set_transform** (*RID* area, *Matrix32* transform)
- void **area_get_param** (*RID* area, *int* param) const
- *Matrix32* **area_get_transform** (*RID* area) const
- void **area_attach_object_instance_ID** (*RID* area, *int* id)
- *int* **area_get_object_instance_ID** (*RID* area) const
- void **area_set_monitor_callback** (*RID* area, *Object* receiver, *String* method)
- *RID* **body_create** (*int* mode=2, *bool* init_sleeping=false)
- void **body_set_space** (*RID* body, *RID* space)
- *RID* **body_get_space** (*RID* body) const
- void **body_set_mode** (*RID* body, *int* mode)
- *int* **body_get_mode** (*RID* body) const
- void **body_add_shape** (*RID* body, *RID* shape, *Matrix32* transform=1,0, 0,1, 0,0)
- void **body_set_shape** (*RID* body, *int* shape_idx, *RID* shape)

- void **body_set_shape_transform** (*RID* body, *int* shape_idx, *Matrix32* transform)
- void **body_set_shape_metadata** (*RID* body, *int* shape_idx, var metadata)
- *int* **body_get_shape_count** (*RID* body) const
- *RID* **body_get_shape** (*RID* body, *int* shape_idx) const
- *Matrix32* **body_get_shape_transform** (*RID* body, *int* shape_idx) const
- void **body_get_shape_metadata** (*RID* body, *int* shape_idx) const
- void **body_remove_shape** (*RID* body, *int* shape_idx)
- void **body_clear_shapes** (*RID* body)
- void **body_set_shape_as_trigger** (*RID* body, *int* shape_idx, *bool* enable)
- *bool* **body_is_shape_set_as_trigger** (*RID* body, *int* shape_idx) const
- void **body_attach_object_instance_ID** (*RID* body, *int* id)
- *int* **body_get_object_instance_ID** (*RID* body) const
- void **body_set_continuous_collision_detection_mode** (*RID* body, *int* mode)
- *int* **body_get_continuous_collision_detection_mode** (*RID* body) const
- void **body_set_layer_mask** (*RID* body, *int* mask)
- *int* **body_get_layer_mask** (*RID* body) const
- void **body_set_collision_mask** (*RID* body, *int* mask)
- *int* **body_get_collision_mask** (*RID* body) const
- void **body_set_param** (*RID* body, *int* param, *float* value)
- *float* **body_get_param** (*RID* body, *int* param) const
- void **body_set_state** (*RID* body, *int* state, var value)
- void **body_get_state** (*RID* body, *int* state) const
- void **body_apply_impulse** (*RID* body, *Vector2* pos, *Vector2* impulse)
- void **body_set_axis_velocity** (*RID* body, *Vector2* axis_velocity)
- void **body_add_collision_exception** (*RID* body, *RID* excepted_body)
- void **body_remove_collision_exception** (*RID* body, *RID* excepted_body)
- void **body_set_max_contacts_reported** (*RID* body, *int* amount)
- *int* **body_get_max_contacts_reported** (*RID* body) const
- void **body_set_one_way_collision_direction** (*RID* body, *Vector2* normal)
- *Vector2* **body_get_one_way_collision_direction** (*RID* body) const
- void **body_set_one_way_collision_max_depth** (*RID* body, *float* depth)
- *float* **body_get_one_way_collision_max_depth** (*RID* body) const
- void **body_set_omit_force_integration** (*RID* body, *bool* enable)
- *bool* **body_is_omitting_force_integration** (*RID* body) const
- void **body_set_force_integration_callback** (*RID* body, *Object* receiver, *String* method, var userdata=NULL)

- *bool* **body_test_motion** (*RID* body, *Vector2* motion, *float* margin=0.08, *Physics2DTestMotionResult* result=NULL)
- void **joint_set_param** (*RID* joint, *int* param, *float* value)
- *float* **joint_get_param** (*RID* joint, *int* param) const
- *RID* **pin_joint_create** (*Vector2* anchor, *RID* body_a, *RID* body_b=RID())
- *RID* **groove_joint_create** (*Vector2* groove1_a, *Vector2* groove2_a, *Vector2* anchor_b, *RID* body_a=RID(), *RID* body_b=RID())
- *RID* **damped_spring_joint_create** (*Vector2* anchor_a, *Vector2* anchor_b, *RID* body_a, *RID* body_b=RID())
- void **damped_string_joint_set_param** (*RID* joint, *int* param, *float* value)
- *float* **damped_string_joint_get_param** (*RID* joint, *int* param) const
- *int* **joint_get_type** (*RID* joint) const
- void **free_rid** (*RID* rid)
- void **set_active** (*bool* active)
- *int* **get_process_info** (*int* process_info)

9.201 Physics2DServerSW

Inherits: *Physics2DServer* < *Object*

Category: Core

9.201.1 Brief Description

9.202 Physics2DShapeQueryParameters

Inherits: *Reference* < *Object*

Category: Core

9.202.1 Brief Description

9.202.2 Member Functions

void	*set_shape* (*Shape2D* shape)
void	*set_shape_rid* (*RID* shape)
RID	*get_shape_rid* ( ) const
void	*set_transform* (*Matrix32* transform)
Matrix32	*get_transform* ( ) const
void	*set_motion* (*Vector2* motion)
Vector2	*get_motion* ( ) const
void	*set_margin* (*float* margin)
float	*get_margin* ( ) const
void	*set_layer_mask* (*int* layer_mask)
int	*get_layer_mask* ( ) const
void	*set_object_type_mask* (*int* object_type_mask)
int	*get_object_type_mask* ( ) const
void	*set_exclude* (*Array* exclude)
Array	*get_exclude* ( ) const

9.202.3 Member Function Description

- void **set_shape** (*Shape2D* shape)
- void **set_shape_rid** (*RID* shape)
- *RID* **get_shape_rid** ( ) const
- void **set_transform** (*Matrix32* transform)
- *Matrix32* **get_transform** ( ) const
- void **set_motion** (*Vector2* motion)
- *Vector2* **get_motion** ( ) const
- void **set_margin** (*float* margin)
- *float* **get_margin** ( ) const
- void **set_layer_mask** (*int* layer_mask)
- *int* **get_layer_mask** ( ) const
- void **set_object_type_mask** (*int* object_type_mask)
- *int* **get_object_type_mask** ( ) const
- void **set_exclude** (*Array* exclude)
- *Array* **get_exclude** ( ) const

9.203 Physics2DShapeQueryResult

Inherits: *Reference* < *Object*

Category: Core

9.203.1 Brief Description

9.203.2 Member Functions

int	*get_result_count* ( ) const
RID	*get_result_rid* (*int* idx) const
int	*get_result_object_id* (*int* idx) const
Object	*get_result_object* (*int* idx) const
int	*get_result_object_shape* (*int* idx) const

9.203.3 Member Function Description

- *int* **get_result_count** ( ) const
- *RID* **get_result_rid** (*int* idx) const
- *int* **get_result_object_id** (*int* idx) const
- *Object* **get_result_object** (*int* idx) const
- *int* **get_result_object_shape** (*int* idx) const

9.204 Physics2DTestMotionResult

Inherits: *Reference* < *Object*

Category: Core

9.204.1 Brief Description

9.204.2 Member Functions

Vector2	*get_motion* ( ) const
Vector2	*get_motion_remainder* ( ) const
Vector2	*get_collision_point* ( ) const
Vector2	*get_collision_normal* ( ) const
Vector2	*get_collider_velocity* ( ) const
int	*get_collider_id* ( ) const
RID	*get_collider_rid* ( ) const
Object	*get_collider* ( ) const
int	*get_collider_shape* ( ) const

9.204.3 Member Function Description

- *Vector2* **get_motion** ( ) const
- *Vector2* **get_motion_remainder** ( ) const
- *Vector2* **get_collision_point** ( ) const
- *Vector2* **get_collision_normal** ( ) const
- *Vector2* **get_collider_velocity** ( ) const

- *int* **get_collider_id** ( ) const
- *RID* **get_collider_rid** ( ) const
- *Object* **get_collider** ( ) const
- *int* **get_collider_shape** ( ) const

9.205 PhysicsBody

Inherits: *CollisionObject* < *Spatial* < *Node* < *Object*

Inherited By: *VehicleBody*, *KinematicBody*, *StaticBody*, *RigidBody*

Category: Core

9.205.1 Brief Description

Base class for different types of Physics bodies.

9.205.2 Member Functions

void	*set_layer_mask* (*int* mask)
int	*get_layer_mask* ( ) const
void	*add_collision_exception_with* (*PhysicsBody* body)
void	*remove_collision_exception_with* (*PhysicsBody* body)

9.205.3 Description

PhysicsBody is an abstract base class for implementing a physics body. All PhysicsBody types inherit from it.

9.205.4 Member Function Description

- void **set_layer_mask** (*int* mask)
- *int* **get_layer_mask** ( ) const
- void **add_collision_exception_with** (*PhysicsBody* body)
- void **remove_collision_exception_with** (*PhysicsBody* body)

9.206 PhysicsBody2D

Inherits: *CollisionObject2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Inherited By: *RigidBody2D*, *StaticBody2D*, *KinematicBody2D*

Category: Core

9.206.1 Brief Description

Base class for all objects affected by physics.

9.206.2 Member Functions

void	*set_layer_mask* (*int* mask)
int	*get_layer_mask* ( ) const
void	*set_collision_mask* (*int* mask)
int	*get_collision_mask* ( ) const
void	*set_collision_mask_bit* (*int* bit, *bool* value)
bool	*get_collision_mask_bit* (*int* bit) const
void	*set_layer_mask_bit* (*int* bit, *bool* value)
bool	*get_layer_mask_bit* (*int* bit) const
void	*set_one_way_collision_direction* (*Vector2* dir)
Vector2	*get_one_way_collision_direction* ( ) const
void	*set_one_way_collision_max_depth* (*float* depth)
float	*get_one_way_collision_max_depth* ( ) const
void	*add_collision_exception_with* (*PhysicsBody2D* body)
void	*remove_collision_exception_with* (*PhysicsBody2D* body)

9.206.3 Description

PhysicsBody2D is an abstract base class for implementing a physics body. All *Body2D types inherit from it.

9.206.4 Member Function Description

- void **set_layer_mask** (*int* mask)

Set the physics layers this area is in.

Collidable objects can exist in any of 32 different layers. These layers are not visual, but more of a tagging system instead. A collidable can use these layers/tags to select with which objects it can collide, using *set_collision_mask*.

A contact is detected if object A is in any of the layers that object B scans, or object B is in any layer scanned by object A.

- *int* **get_layer_mask** ( ) const

Return the physics layer this area is in.

- void **set_collision_mask** (*int* mask)

Set the physics layers this area can scan for collisions.

- *int* **get_collision_mask** ( ) const

Return the physics layers this area can scan for collisions.

- void **set_collision_mask_bit** (*int* bit, *bool* value)

Set/clear individual bits on the collision mask. This makes selecting the areas scanned easier.

- *bool* **get_collision_mask_bit** (*int* bit) const

Return an individual bit on the collision mask.

- void **set_layer_mask_bit** (*int* bit, *bool* value)

Set/clear individual bits on the layer mask. This makes getting a body in/out of only one layer easier.

- *bool* **get_layer_mask_bit** (*int* bit) const

Return an individual bit on the collision mask.

- void **set_one_way_collision_direction** (*Vector2* dir)

Set a direction in which bodies can go through this one. If this value is different from (0,0), any movement within 90 degrees of this vector is considered a valid movement. Set this direction to (0,0) to disable one-way collisions.

- *Vector2* **get_one_way_collision_direction** ( ) const

Return the direction used for one-way collision detection.

- void **set_one_way_collision_max_depth** (*float* depth)

Set how far a body can go through this one, when it allows one-way collisions (see *set_one_way_collision_direction*).

- *float* **get_one_way_collision_max_depth** ( ) const

Return how far a body can go through this one, when it allows one-way collisions.

- void **add_collision_exception_with** (*PhysicsBody2D* body)

Adds a body to the collision exception list. This list contains bodies that this body will not collide with.

- void **remove_collision_exception_with** (*PhysicsBody2D* body)

Removes a body from the collision exception list.

9.207 PhysicsDirectBodyState

Inherits: *Object*

Inherited By: *PhysicsDirectBodyStateSW*

Category: Core

9.207.1 Brief Description

9.207.2 Member Functions

Vector3	*get_total_gravity* ( ) const
float	*get_total_linear_damp* ( ) const
float	*get_total_angular_damp* ( ) const
float	*get_inverse_mass* ( ) const
Vector3	*get_inverse_inertia* ( ) const
void	*set_linear_velocity* (*Vector3* velocity)
Vector3	*get_linear_velocity* ( ) const
void	*set_angular_velocity* (*Vector3* velocity)
Vector3	*get_angular_velocity* ( ) const
void	*set_transform* (*Transform* transform)
Transform	*get_transform* ( ) const
void	*add_force* (*Vector3* force, *Vector3* pos)
void	*apply_impulse* (*Vector3* pos, *Vector3* j)
void	*set_sleep_state* (*bool* enabled)
bool	*is_sleeping* ( ) const
int	*get_contact_count* ( ) const
Vector3	*get_contact_local_pos* (*int* contact_idx) const
Vector3	*get_contact_local_normal* (*int* contact_idx) const
int	*get_contact_local_shape* (*int* contact_idx) const
RID	*get_contact_collider* (*int* contact_idx) const
Vector3	*get_contact_collider_pos* (*int* contact_idx) const
int	*get_contact_collider_id* (*int* contact_idx) const
Object	*get_contact_collider_object* (*int* contact_idx) const
int	*get_contact_collider_shape* (*int* contact_idx) const
Vector3	*get_contact_collider_velocity_at_pos* (*int* contact_idx) const
float	*get_step* ( ) const
void	*integrate_forces* ( )
PhysicsDirectSpaceState	*get_space_state* ( )

9.207.3 Member Function Description

- *Vector3* **get_total_gravity** ( ) const
- *float* **get_total_linear_damp** ( ) const
- *float* **get_total_angular_damp** ( ) const
- *float* **get_inverse_mass** ( ) const
- *Vector3* **get_inverse_inertia** ( ) const
- void **set_linear_velocity** (*Vector3* velocity)
- *Vector3* **get_linear_velocity** ( ) const
- void **set_angular_velocity** (*Vector3* velocity)
- *Vector3* **get_angular_velocity** ( ) const
- void **set_transform** (*Transform* transform)
- *Transform* **get_transform** ( ) const

- void **add_force** (*Vector3* force, *Vector3* pos)
- void **apply_impulse** (*Vector3* pos, *Vector3* j)
- void **set_sleep_state** (*bool* enabled)
- *bool* **is_sleeping** ( ) const
- *int* **get_contact_count** ( ) const
- *Vector3* **get_contact_local_pos** (*int* contact_idx) const
- *Vector3* **get_contact_local_normal** (*int* contact_idx) const
- *int* **get_contact_local_shape** (*int* contact_idx) const
- *RID* **get_contact_collider** (*int* contact_idx) const
- *Vector3* **get_contact_collider_pos** (*int* contact_idx) const
- *int* **get_contact_collider_id** (*int* contact_idx) const
- *Object* **get_contact_collider_object** (*int* contact_idx) const
- *int* **get_contact_collider_shape** (*int* contact_idx) const
- *Vector3* **get_contact_collider_velocity_at_pos** (*int* contact_idx) const
- *float* **get_step** ( ) const
- void **integrate_forces** ( )
- *PhysicsDirectSpaceState* **get_space_state** ( )

9.208 PhysicsDirectBodyStateSW

Inherits: *PhysicsDirectBodyState* < *Object*

Category: Core

9.208.1 Brief Description

9.209 PhysicsDirectSpaceState

Inherits: *Object*

Category: Core

9.209.1 Brief Description

9.209.2 Member Functions

Dictionary	*intersect_ray* (*Vector3* from, *Vector3* to, *Array* exclude=Array(), *int* layer_mask=2147483647, *int* type_mask=15)
Array	*intersect_shape* (*PhysicsShapeQueryParameters* shape, *int* max_results=32)
Array	*cast_motion* (*PhysicsShapeQueryParameters* shape, *Vector3* motion)
Array	*collide_shape* (*PhysicsShapeQueryParameters* shape, *int* max_results=32)
Dictionary	*get_rest_info* (*PhysicsShapeQueryParameters* shape)

9.209.3 Numeric Constants

- **TYPE_MASK_STATIC_BODY = 1**
- **TYPE_MASK_KINEMATIC_BODY = 2**
- **TYPE_MASK_RIGID_BODY = 4**
- **TYPE_MASK_CHARACTER_BODY = 8**
- **TYPE_MASK_AREA = 16**
- **TYPE_MASK_COLLISION = 15**

9.209.4 Member Function Description

- *Dictionary* **intersect_ray** (*Vector3* from, *Vector3* to, *Array* exclude=Array(), *int* layer_mask=2147483647, *int* type_mask=15)
- *Array* **intersect_shape** (*PhysicsShapeQueryParameters* shape, *int* max_results=32)
- *Array* **cast_motion** (*PhysicsShapeQueryParameters* shape, *Vector3* motion)
- *Array* **collide_shape** (*PhysicsShapeQueryParameters* shape, *int* max_results=32)
- *Dictionary* **get_rest_info** (*PhysicsShapeQueryParameters* shape)

9.210 PhysicsServer

Inherits: *Object*

Inherited By: *PhysicsServerSW*

Category: Core

9.210.1 Brief Description

9.210.2 Member Functions

RID	*shape_create* (*int* type)

Continúa en la página siguiente

Tabla 9.21 – proviene de la página anterior

void	*shape_set_data* (*RID* shape, var data)
int	*shape_get_type* (*RID* shape) const
void	*shape_get_data* (*RID* shape) const
RID	*space_create* ( )
void	*space_set_active* (*RID* space, *bool* active)
bool	*space_is_active* (*RID* space) const
void	*space_set_param* (*RID* space, *int* param, *float* value)
float	*space_get_param* (*RID* space, *int* param) const
PhysicsDirectSpaceState	*space_get_direct_state* (*RID* space)
RID	*area_create* ( )
void	*area_set_space* (*RID* area, *RID* space)
RID	*area_get_space* (*RID* area) const
void	*area_set_space_override_mode* (*RID* area, *int* mode)
int	*area_get_space_override_mode* (*RID* area) const
void	*area_add_shape* (*RID* area, *RID* shape, *Transform* transform=Transform())
void	*area_set_shape* (*RID* area, *int* shape_idx, *RID* shape)
void	*area_set_shape_transform* (*RID* area, *int* shape_idx, *Transform* transform)
int	*area_get_shape_count* (*RID* area) const
RID	*area_get_shape* (*RID* area, *int* shape_idx) const
Transform	*area_get_shape_transform* (*RID* area, *int* shape_idx) const
void	*area_remove_shape* (*RID* area, *int* shape_idx)
void	*area_clear_shapes* (*RID* area)
void	*area_set_param* (*RID* area, *int* param, var value)
void	*area_set_transform* (*RID* area, *Transform* transform)
void	*area_get_param* (*RID* area, *int* param) const
Transform	*area_get_transform* (*RID* area) const
void	*area_attach_object_instance_ID* (*RID* area, *int* id)
int	*area_get_object_instance_ID* (*RID* area) const
void	*area_set_monitor_callback* (*RID* area, *Object* receiver, *String* method)
void	*area_set_ray_pickable* (*RID* area, *bool* enable)
bool	*area_is_ray_pickable* (*RID* area) const
RID	*body_create* (*int* mode=2, *bool* init_sleeping=false)
void	*body_set_space* (*RID* body, *RID* space)
RID	*body_get_space* (*RID* body) const
void	*body_set_mode* (*RID* body, *int* mode)
int	*body_get_mode* (*RID* body) const
void	*body_add_shape* (*RID* body, *RID* shape, *Transform* transform=Transform())
void	*body_set_shape* (*RID* body, *int* shape_idx, *RID* shape)
void	*body_set_shape_transform* (*RID* body, *int* shape_idx, *Transform* transform)
int	*body_get_shape_count* (*RID* body) const
RID	*body_get_shape* (*RID* body, *int* shape_idx) const
Transform	*body_get_shape_transform* (*RID* body, *int* shape_idx) const
void	*body_remove_shape* (*RID* body, *int* shape_idx)
void	*body_clear_shapes* (*RID* body)
void	*body_attach_object_instance_ID* (*RID* body, *int* id)
int	*body_get_object_instance_ID* (*RID* body) const
void	*body_set_enable_continuous_collision_detection* (*RID* body, *bool* enable)
bool	*body_is_continuous_collision_detection_enabled* (*RID* body) const
void	*body_set_param* (*RID* body, *int* param, *float* value)
float	*body_get_param* (*RID* body, *int* param) const

Continúa en la página siguiente

Tabla 9.21 – proviene de la página anterior

void	*body_set_state* (*RID* body, *int* state, var value)
void	*body_get_state* (*RID* body, *int* state) const
void	*body_apply_impulse* (*RID* body, *Vector3* pos, *Vector3* impulse)
void	*body_set_axis_velocity* (*RID* body, *Vector3* axis_velocity)
void	*body_set_axis_lock* (*RID* body, *int* axis)
int	*body_get_axis_lock* (*RID* body) const
void	*body_add_collision_exception* (*RID* body, *RID* excepted_body)
void	*body_remove_collision_exception* (*RID* body, *RID* excepted_body)
void	*body_set_max_contacts_reported* (*RID* body, *int* amount)
int	*body_get_max_contacts_reported* (*RID* body) const
void	*body_set_omit_force_integration* (*RID* body, *bool* enable)
bool	*body_is_omitting_force_integration* (*RID* body) const
void	*body_set_force_integration_callback* (*RID* body, *Object* receiver, *String* method, var userdata=NULL)
void	*body_set_ray_pickable* (*RID* body, *bool* enable)
bool	*body_is_ray_pickable* (*RID* body) const
RID	*joint_create_pin* (*RID* body_A, *Vector3* local_A, *RID* body_B, *Vector3* local_B)
void	*pin_joint_set_param* (*RID* joint, *int* param, *float* value)
float	*pin_joint_get_param* (*RID* joint, *int* param) const
void	*pin_joint_set_local_A* (*RID* joint, *Vector3* local_A)
Vector3	*pin_joint_get_local_A* (*RID* joint) const
void	*pin_joint_set_local_B* (*RID* joint, *Vector3* local_B)
Vector3	*pin_joint_get_local_B* (*RID* joint) const
RID	*joint_create_hinge* (*RID* body_A, *Transform* hinge_A, *RID* body_B, *Transform* hinge_B)
void	*hinge_joint_set_param* (*RID* joint, *int* param, *float* value)
float	*hinge_joint_get_param* (*RID* joint, *int* param) const
void	*hinge_joint_set_flag* (*RID* joint, *int* flag, *bool* enabled)
bool	*hinge_joint_get_flag* (*RID* joint, *int* flag) const
RID	*joint_create_slider* (*RID* body_A, *Transform* local_ref_A, *RID* body_B, *Transform* local_ref_B)
void	*slider_joint_set_param* (*RID* joint, *int* param, *float* value)
float	*slider_joint_get_param* (*RID* joint, *int* param) const
RID	*joint_create_cone_twist* (*RID* body_A, *Transform* local_ref_A, *RID* body_B, *Transform* local_ref_B)
void	*cone_twist_joint_set_param* (*RID* joint, *int* param, *float* value)
float	*cone_twist_joint_get_param* (*RID* joint, *int* param) const
int	*joint_get_type* (*RID* joint) const
void	*joint_set_solver_priority* (*RID* joint, *int* priority)
int	*joint_get_solver_priority* (*RID* joint) const
RID	*joint_create_generic_6dof* (*RID* body_A, *Transform* local_ref_A, *RID* body_B, *Transform* local_ref_B)
void	*generic_6dof_joint_set_param* (*RID* joint, *int* axis, *int* param, *float* value)
float	*generic_6dof_joint_get_param* (*RID* joint, *int* axis, *int* param)
void	*generic_6dof_joint_set_flag* (*RID* joint, *int* axis, *int* flag, *bool* enable)
bool	*generic_6dof_joint_get_flag* (*RID* joint, *int* axis, *int* flag)
void	*free_rid* (*RID* rid)
void	*set_active* (*bool* active)
int	*get_process_info* (*int* process_info)

9.210.3 Numeric Constants

- **JOINT_PIN = 0**
- **JOINT_HINGE = 1**
- **JOINT_SLIDER = 2**

- **JOINT_CONE_TWIST = 3**
- **JOINT_6DOF = 4**
- **PIN_JOINT_BIAS = 0**
- **PIN_JOINT_DAMPING = 1**
- **PIN_JOINT_IMPULSE_CLAMP = 2**
- **HINGE_JOINT_BIAS = 0**
- **HINGE_JOINT_LIMIT_UPPER = 1**
- **HINGE_JOINT_LIMIT_LOWER = 2**
- **HINGE_JOINT_LIMIT_BIAS = 3**
- **HINGE_JOINT_LIMIT_SOFTNESS = 4**
- **HINGE_JOINT_LIMIT_RELAXATION = 5**
- **HINGE_JOINT_MOTOR_TARGET_VELOCITY = 6**
- **HINGE_JOINT_MOTOR_MAX_IMPULSE = 7**
- **HINGE_JOINT_FLAG_USE_LIMIT = 0**
- **HINGE_JOINT_FLAG_ENABLE_MOTOR = 1**
- **SLIDER_JOINT_LINEAR_LIMIT_UPPER = 0**
- **SLIDER_JOINT_LINEAR_LIMIT_LOWER = 1**
- **SLIDER_JOINT_LINEAR_LIMIT_SOFTNESS = 2**
- **SLIDER_JOINT_LINEAR_LIMIT_RESTITUTION = 3**
- **SLIDER_JOINT_LINEAR_LIMIT_DAMPING = 4**
- **SLIDER_JOINT_LINEAR_MOTION_SOFTNESS = 5**
- **SLIDER_JOINT_LINEAR_MOTION_RESTITUTION = 6**
- **SLIDER_JOINT_LINEAR_MOTION_DAMPING = 7**
- **SLIDER_JOINT_LINEAR_ORTHOGONAL_SOFTNESS = 8**
- **SLIDER_JOINT_LINEAR_ORTHOGONAL_RESTITUTION = 9**
- **SLIDER_JOINT_LINEAR_ORTHOGONAL_DAMPING = 10**
- **SLIDER_JOINT_ANGULAR_LIMIT_UPPER = 11**
- **SLIDER_JOINT_ANGULAR_LIMIT_LOWER = 12**
- **SLIDER_JOINT_ANGULAR_LIMIT_SOFTNESS = 13**
- **SLIDER_JOINT_ANGULAR_LIMIT_RESTITUTION = 14**
- **SLIDER_JOINT_ANGULAR_LIMIT_DAMPING = 15**
- **SLIDER_JOINT_ANGULAR_MOTION_SOFTNESS = 16**
- **SLIDER_JOINT_ANGULAR_MOTION_RESTITUTION = 17**
- **SLIDER_JOINT_ANGULAR_MOTION_DAMPING = 18**
- **SLIDER_JOINT_ANGULAR_ORTHOGONAL_SOFTNESS = 19**
- **SLIDER_JOINT_ANGULAR_ORTHOGONAL_RESTITUTION = 20**

- **SLIDER_JOINT_ANGULAR_ORTHOGONAL_DAMPING = 21**
- **SLIDER_JOINT_MAX = 22**
- **CONE_TWIST_JOINT_SWING_SPAN = 0**
- **CONE_TWIST_JOINT_TWIST_SPAN = 1**
- **CONE_TWIST_JOINT_BIAS = 2**
- **CONE_TWIST_JOINT_SOFTNESS = 3**
- **CONE_TWIST_JOINT_RELAXATION = 4**
- **G6DOF_JOINT_LINEAR_LOWER_LIMIT = 0**
- **G6DOF_JOINT_LINEAR_UPPER_LIMIT = 1**
- **G6DOF_JOINT_LINEAR_LIMIT_SOFTNESS = 2**
- **G6DOF_JOINT_LINEAR_RESTITUTION = 3**
- **G6DOF_JOINT_LINEAR_DAMPING = 4**
- **G6DOF_JOINT_ANGULAR_LOWER_LIMIT = 5**
- **G6DOF_JOINT_ANGULAR_UPPER_LIMIT = 6**
- **G6DOF_JOINT_ANGULAR_LIMIT_SOFTNESS = 7**
- **G6DOF_JOINT_ANGULAR_DAMPING = 8**
- **G6DOF_JOINT_ANGULAR_RESTITUTION = 9**
- **G6DOF_JOINT_ANGULAR_FORCE_LIMIT = 10**
- **G6DOF_JOINT_ANGULAR_ERP = 11**
- **G6DOF_JOINT_ANGULAR_MOTOR_TARGET_VELOCITY = 12**
- **G6DOF_JOINT_ANGULAR_MOTOR_FORCE_LIMIT = 13**
- **G6DOF_JOINT_FLAG_ENABLE_LINEAR_LIMIT = 0**
- **G6DOF_JOINT_FLAG_ENABLE_ANGULAR_LIMIT = 1**
- **G6DOF_JOINT_FLAG_ENABLE_MOTOR = 2**
- **SHAPE_PLANE = 0**
- **SHAPE_RAY = 1**
- **SHAPE_SPHERE = 2**
- **SHAPE_BOX = 3**
- **SHAPE_CAPSULE = 4**
- **SHAPE_CONVEX_POLYGON = 5**
- **SHAPE_CONCAVE_POLYGON = 6**
- **SHAPE_HEIGHTMAP = 7**
- **SHAPE_CUSTOM = 8**
- **AREA_PARAM_GRAVITY = 0**
- **AREA_PARAM_GRAVITY_VECTOR = 1**
- **AREA_PARAM_GRAVITY_IS_POINT = 2**

- **AREA_PARAM_GRAVITY_DISTANCE_SCALE = 3**
- **AREA_PARAM_GRAVITY_POINT_ATTENUATION = 4**
- **AREA_PARAM_LINEAR_DAMP = 5**
- **AREA_PARAM_ANGULAR_DAMP = 6**
- **AREA_PARAM_PRIORITY = 7**
- **AREA_SPACE_OVERRIDE_DISABLED = 0** — This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.
- **AREA_SPACE_OVERRIDE_COMBINE = 1** — This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.
- **AREA_SPACE_OVERRIDE_COMBINE_REPLACE = 2** — This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.
- **AREA_SPACE_OVERRIDE_REPLACE = 3** — This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.
- **AREA_SPACE_OVERRIDE_REPLACE_COMBINE = 4** — This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.
- **BODY_MODE_STATIC = 0**
- **BODY_MODE_KINEMATIC = 1**
- **BODY_MODE_RIGID = 2**
- **BODY_MODE_CHARACTER = 3**
- **BODY_PARAM_BOUNCE = 0**
- **BODY_PARAM_FRICTION = 1**
- **BODY_PARAM_MASS = 2**
- **BODY_PARAM_GRAVITY_SCALE = 3**
- **BODY_PARAM_ANGULAR_DAMP = 5**
- **BODY_PARAM_LINEAR_DAMP = 4**
- **BODY_PARAM_MAX = 6**
- **BODY_STATE_TRANSFORM = 0**
- **BODY_STATE_LINEAR_VELOCITY = 1**
- **BODY_STATE_ANGULAR_VELOCITY = 2**
- **BODY_STATE_SLEEPING = 3**
- **BODY_STATE_CAN_SLEEP = 4**
- **AREA_BODY_ADDED = 0**
- **AREA_BODY_REMOVED = 1**
- **INFO_ACTIVE_OBJECTS = 0**
- **INFO_COLLISION_PAIRS = 1**
- **INFO_ISLAND_COUNT = 2**

9.210.4 Member Function Description

- *RID* **shape_create** (*int* type)
- void **shape_set_data** (*RID* shape, var data)
- *int* **shape_get_type** (*RID* shape) const
- void **shape_get_data** (*RID* shape) const
- *RID* **space_create** ( )
- void **space_set_active** (*RID* space, *bool* active)
- *bool* **space_is_active** (*RID* space) const
- void **space_set_param** (*RID* space, *int* param, *float* value)
- *float* **space_get_param** (*RID* space, *int* param) const
- *PhysicsDirectSpaceState* **space_get_direct_state** (*RID* space)
- *RID* **area_create** ( )
- void **area_set_space** (*RID* area, *RID* space)
- *RID* **area_get_space** (*RID* area) const
- void **area_set_space_override_mode** (*RID* area, *int* mode)
- *int* **area_get_space_override_mode** (*RID* area) const
- void **area_add_shape** (*RID* area, *RID* shape, *Transform* transform=Transform())
- void **area_set_shape** (*RID* area, *int* shape_idx, *RID* shape)
- void **area_set_shape_transform** (*RID* area, *int* shape_idx, *Transform* transform)
- *int* **area_get_shape_count** (*RID* area) const
- *RID* **area_get_shape** (*RID* area, *int* shape_idx) const
- *Transform* **area_get_shape_transform** (*RID* area, *int* shape_idx) const
- void **area_remove_shape** (*RID* area, *int* shape_idx)
- void **area_clear_shapes** (*RID* area)
- void **area_set_param** (*RID* area, *int* param, var value)
- void **area_set_transform** (*RID* area, *Transform* transform)
- void **area_get_param** (*RID* area, *int* param) const
- *Transform* **area_get_transform** (*RID* area) const
- void **area_attach_object_instance_ID** (*RID* area, *int* id)
- *int* **area_get_object_instance_ID** (*RID* area) const
- void **area_set_monitor_callback** (*RID* area, *Object* receiver, *String* method)
- void **area_set_ray_pickable** (*RID* area, *bool* enable)
- *bool* **area_is_ray_pickable** (*RID* area) const
- *RID* **body_create** (*int* mode=2, *bool* init_sleeping=false)
- void **body_set_space** (*RID* body, *RID* space)
- *RID* **body_get_space** (*RID* body) const

- void **body_set_mode** (*RID* body, *int* mode)
- *int* **body_get_mode** (*RID* body) const
- void **body_add_shape** (*RID* body, *RID* shape, *Transform* transform=Transform())
- void **body_set_shape** (*RID* body, *int* shape_idx, *RID* shape)
- void **body_set_shape_transform** (*RID* body, *int* shape_idx, *Transform* transform)
- *int* **body_get_shape_count** (*RID* body) const
- *RID* **body_get_shape** (*RID* body, *int* shape_idx) const
- *Transform* **body_get_shape_transform** (*RID* body, *int* shape_idx) const
- void **body_remove_shape** (*RID* body, *int* shape_idx)
- void **body_clear_shapes** (*RID* body)
- void **body_attach_object_instance_ID** (*RID* body, *int* id)
- *int* **body_get_object_instance_ID** (*RID* body) const
- void **body_set_enable_continuous_collision_detection** (*RID* body, *bool* enable)
- *bool* **body_is_continuous_collision_detection_enabled** (*RID* body) const
- void **body_set_param** (*RID* body, *int* param, *float* value)
- *float* **body_get_param** (*RID* body, *int* param) const
- void **body_set_state** (*RID* body, *int* state, var value)
- void **body_get_state** (*RID* body, *int* state) const
- void **body_apply_impulse** (*RID* body, *Vector3* pos, *Vector3* impulse)
- void **body_set_axis_velocity** (*RID* body, *Vector3* axis_velocity)
- void **body_set_axis_lock** (*RID* body, *int* axis)
- *int* **body_get_axis_lock** (*RID* body) const
- void **body_add_collision_exception** (*RID* body, *RID* excepted_body)
- void **body_remove_collision_exception** (*RID* body, *RID* excepted_body)
- void **body_set_max_contacts_reported** (*RID* body, *int* amount)
- *int* **body_get_max_contacts_reported** (*RID* body) const
- void **body_set_omit_force_integration** (*RID* body, *bool* enable)
- *bool* **body_is_omitting_force_integration** (*RID* body) const
- void **body_set_force_integration_callback** (*RID* body, *Object* receiver, *String* method, var userdata=NULL)
- void **body_set_ray_pickable** (*RID* body, *bool* enable)
- *bool* **body_is_ray_pickable** (*RID* body) const
- *RID* **joint_create_pin** (*RID* body_A, *Vector3* local_A, *RID* body_B, *Vector3* local_B)
- void **pin_joint_set_param** (*RID* joint, *int* param, *float* value)
- *float* **pin_joint_get_param** (*RID* joint, *int* param) const
- void **pin_joint_set_local_A** (*RID* joint, *Vector3* local_A)
- *Vector3* **pin_joint_get_local_A** (*RID* joint) const

- void **pin_joint_set_local_B** (*RID* joint, *Vector3* local_B)
- *Vector3* **pin_joint_get_local_B** (*RID* joint) const
- *RID* **joint_create_hinge** (*RID* body_A, *Transform* hinge_A, *RID* body_B, *Transform* hinge_B)
- void **hinge_joint_set_param** (*RID* joint, *int* param, *float* value)
- *float* **hinge_joint_get_param** (*RID* joint, *int* param) const
- void **hinge_joint_set_flag** (*RID* joint, *int* flag, *bool* enabled)
- *bool* **hinge_joint_get_flag** (*RID* joint, *int* flag) const
- *RID* **joint_create_slider** (*RID* body_A, *Transform* local_ref_A, *RID* body_B, *Transform* local_ref_B)
- void **slider_joint_set_param** (*RID* joint, *int* param, *float* value)
- *float* **slider_joint_get_param** (*RID* joint, *int* param) const
- *RID* **joint_create_cone_twist** (*RID* body_A, *Transform* local_ref_A, *RID* body_B, *Transform* local_ref_B)
- void **cone_twist_joint_set_param** (*RID* joint, *int* param, *float* value)
- *float* **cone_twist_joint_get_param** (*RID* joint, *int* param) const
- *int* **joint_get_type** (*RID* joint) const
- void **joint_set_solver_priority** (*RID* joint, *int* priority)
- *int* **joint_get_solver_priority** (*RID* joint) const
- *RID* **joint_create_generic_6dof** (*RID* body_A, *Transform* local_ref_A, *RID* body_B, *Transform* local_ref_B)
- void **generic_6dof_joint_set_param** (*RID* joint, *int* axis, *int* param, *float* value)
- *float* **generic_6dof_joint_get_param** (*RID* joint, *int* axis, *int* param)
- void **generic_6dof_joint_set_flag** (*RID* joint, *int* axis, *int* flag, *bool* enable)
- *bool* **generic_6dof_joint_get_flag** (*RID* joint, *int* axis, *int* flag)
- void **free_rid** (*RID* rid)
- void **set_active** (*bool* active)
- *int* **get_process_info** (*int* process_info)

9.211 PhysicsServerSW

Inherits: *PhysicsServer* **<** *Object*

Category: Core

9.211.1 Brief Description

9.212 PhysicsShapeQueryParameters

Inherits: *Reference* **<** *Object*

Category: Core

9.212.1 Brief Description

9.212.2 Member Functions

void	*set_shape* (*Shape* shape)
void	*set_shape_rid* (*RID* shape)
RID	*get_shape_rid* ( ) const
void	*set_transform* (*Transform* transform)
Transform	*get_transform* ( ) const
void	*set_margin* (*float* margin)
float	*get_margin* ( ) const
void	*set_layer_mask* (*int* layer_mask)
int	*get_layer_mask* ( ) const
void	*set_object_type_mask* (*int* object_type_mask)
int	*get_object_type_mask* ( ) const
void	*set_exclude* (*Array* exclude)
Array	*get_exclude* ( ) const

9.212.3 Member Function Description

- void **set_shape** (*Shape* shape)
- void **set_shape_rid** (*RID* shape)
- *RID* **get_shape_rid** ( ) const
- void **set_transform** (*Transform* transform)
- *Transform* **get_transform** ( ) const
- void **set_margin** (*float* margin)
- *float* **get_margin** ( ) const
- void **set_layer_mask** (*int* layer_mask)
- *int* **get_layer_mask** ( ) const
- void **set_object_type_mask** (*int* object_type_mask)
- *int* **get_object_type_mask** ( ) const
- void **set_exclude** (*Array* exclude)
- *Array* **get_exclude** ( ) const

9.213 PhysicsShapeQueryResult

Inherits: *Reference* < *Object*

Category: Core

9.213.1 Brief Description

Result of a shape query in Physics2DServer.

9.213.2 Member Functions

int	*get_result_count* ( ) const
RID	*get_result_rid* (*int* idx) const
int	*get_result_object_id* (*int* idx) const
Object	*get_result_object* (*int* idx) const
int	*get_result_object_shape* (*int* idx) const

9.213.3 Member Function Description

- *int* **get_result_count** ( ) const
- *RID* **get_result_rid** (*int* idx) const
- *int* **get_result_object_id** (*int* idx) const
- *Object* **get_result_object** (*int* idx) const
- *int* **get_result_object_shape** (*int* idx) const

9.214 PinJoint

Inherits: *Joint* < *Spatial* < *Node* < *Object*

Category: Core

9.214.1 Brief Description

9.214.2 Member Functions

void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const

9.214.3 Numeric Constants

- **PARAM_BIAS = 0**
- **PARAM_DAMPING = 1**
- **PARAM_IMPULSE_CLAMP = 2**

9.214.4 Member Function Description

- void **set_param** (*int* param, *float* value)
- *float* **get_param** (*int* param) const

9.215 PinJoint2D

Inherits: *Joint2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.215.1 Brief Description

Pin Joint for 2D Shapes.

9.215.2 Member Functions

void	*set_softness* (*float* softness)
float	*get_softness* ( ) const

9.215.3 Description

Pin Joint for 2D Rigid Bodies. It pins 2 bodies (rigid or static) together, or a single body to a fixed position in space.

9.215.4 Member Function Description

- void **set_softness** (*float* softness)
- *float* **get_softness** ( ) const

9.216 Plane

Category: Built-In Types

9.216.1 Brief Description

Plane in hessian form.

9.216.2 Member Functions

Vector3	*center* ( )
float	*distance_to* (*Vector3* point)
Vector3	*get_any_point* ( )
bool	*has_point* (*Vector3* point, *float* epsilon=0.00001)
Vector3	*intersect_3* (*Plane* b, *Plane* c)
Vector3	*intersects_ray* (*Vector3* from, *Vector3* dir)
Vector3	*intersects_segment* (*Vector3* begin, *Vector3* end)
bool	*is_point_over* (*Vector3* point)
Plane	*normalized* ( )
Vector3	*project* (*Vector3* point)
Plane	Plane (*float* a, *float* b, *float* c, *float* d)
Plane	Plane (*Vector3* v1, *Vector3* v2, *Vector3* v3)
Plane	Plane (*Vector3* normal, *float* d)

9.216.3 Member Variables

- *Vector3* **normal**
- *float* **x**
- *float* **y**
- *float* **z**
- *float* **d**

9.216.4 Description

Plane represents a normalized plane equation. Basically, "normal" is the normal of the plane (a,b,c normalized), and "d" is the distance from the origin to the plane (in the direction of "normal"). "Over" or "Above" the plane is considered the side of the plane towards where the normal is pointing.

9.216.5 Member Function Description

- *Vector3* **center** ( )

Returns the center of the plane.

- *float* **distance_to** (*Vector3* point)

Returns the shortest distance from the plane to the position "point".

- *Vector3* **get_any_point** ( )

Returns a point on the plane.

- *bool* **has_point** (*Vector3* point, *float* epsilon=0.00001)

Returns true if "point" is inside the plane (by a very minimum threshold).

- *Vector3* **intersect_3** (*Plane* b, *Plane* c)

Returns the intersection point of the three planes "b", "c" and this plane. If no intersection is found null is returned.

- *Vector3* **intersects_ray** (*Vector3* from, *Vector3* dir)

Returns the intersection point of a ray consisting of the position "from" and the direction normal "dir" with this plane. If no intersection is found null is returned.

- *Vector3* **intersects_segment** (*Vector3* begin, *Vector3* end)

Returns the intersection point of a segment from position "begin" to position "end" with this plane. If no intersection is found null is returned.

- *bool* **is_point_over** (*Vector3* point)

Returns true if "point" is located above the plane.

- *Plane* **normalized** ( )

Returns a copy of the plane, normalized.

- *Vector3* **project** (*Vector3* point)

Returns the orthogonal projection of point "p" into a point in the plane.

- *Plane* **Plane** (*float* a, *float* b, *float* c, *float* d)

Creates a plane from the three parameters "a", "b", "c" and "d".

- *Plane* **Plane** (*Vector3* v1, *Vector3* v2, *Vector3* v3)

Creates a plane from three points.

- *Plane* **Plane** (*Vector3* normal, *float* d)

Creates a plane from the normal and the plane's distance to the origin.

9.217 PlaneShape

Inherits: *Shape* < *Resource* < *Reference* < *Object*

Category: Core

9.217.1 Brief Description

9.217.2 Member Functions

void	*set_plane* (*Plane* plane)
Plane	*get_plane* ( ) const

9.217.3 Member Function Description

- void **set_plane** (*Plane* plane)
- *Plane* **get_plane** ( ) const

9.218 Polygon2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.218.1 Brief Description

2D polygon representation

9.218.2 Member Functions

void	*set_polygon* (*Vector2Array* polygon)
Vector2Array	*get_polygon* ( ) const
void	*set_uv* (*Vector2Array* uv)
Vector2Array	*get_uv* ( ) const
void	*set_color* (*Color* color)
Color	*get_color* ( ) const
void	*set_texture* (*Object* texture)
Object	*get_texture* ( ) const
void	*set_texture_offset* (*Vector2* texture_offset)
Vector2	*get_texture_offset* ( ) const
void	*set_texture_rotation* (*float* texture_rotation)
float	*get_texture_rotation* ( ) const
void	*set_texture_scale* (*Vector2* texture_scale)
Vector2	*get_texture_scale* ( ) const
void	*set_invert* (*bool* invert)
bool	*get_invert* ( ) const
void	*set_invert_border* (*float* invert_border)
float	*get_invert_border* ( ) const
void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const

9.218.3 Description

A Polygon2D is defined by a set of n points connected together by line segments, meaning that the point 1 will be connected with point 2, point 2 with point 3 ..., point n-1 with point n and point n with point 1 in order to close the loop and define a plane.

9.218.4 Member Function Description

- void **set_polygon** (*Vector2Array* polygon)

Defines the set of points that will represent the polygon.

- *Vector2Array* **get_polygon** ( ) const

Returns the set of points that defines this polygon

- void **set_uv** (*Vector2Array* uv)

Sets the uv value for every point of the polygon

- *Vector2Array* **get_uv** ( ) const

Returns the uv value associated with every point of the polygon

- void **set_color** (*Color* color)

Sets the polygon fill color, if the polygon has a texture defined, the defined texture will be tinted to the polygon fill color.

- *Color* **get_color ( )** const

Returns the polygon fill color

- void **set_texture (** *Object* texture **)**

Sets the polygon texture.

- *Object* **get_texture ( )** const

Returns the polygon texture

- void **set_texture_offset (** *Vector2* texture_offset **)**

Sets the offset of the polygon texture. Initially the texture will appear anchored to the polygon position, the offset is used to move the texture location away from that point (notice that the texture origin is set to its top left corner, so when offset is 0,0 the top left corner of the texture is at the polygon position), for example setting the offset to 10, 10 will move the texture 10 units to the left and 10 units to the top.

- *Vector2* **get_texture_offset ( )** const

Returns the polygon texture offset.

- void **set_texture_rotation (** *float* texture_rotation **)**

Sets the amount of rotation of the polygon texture, `texture_rotation` is specified in radians and clockwise rotation.

- *float* **get_texture_rotation ( )** const

Returns the rotation in radians of the texture polygon.

- void **set_texture_scale (** *Vector2* texture_scale **)**
- *Vector2* **get_texture_scale ( )** const
- void **set_invert (** *bool* invert **)**

Sets the polygon as the defined polygon bounding box minus the defined polygon (the defined polygon will appear as a hole on square that contains the defined polygon).

- *bool* **get_invert ( )** const

Returns whether this polygon is inverted or not

- void **set_invert_border (** *float* invert_border **)**
- *float* **get_invert_border ( )** const
- void **set_offset (** *Vector2* offset **)**

Sets the amount of distance from the polygon points from the polygon position, for example if the offset is set to 10,10 then all the polygon points will move 10 units to the right and 10 units to the bottom.

- *Vector2* **get_offset ( )** const

Returns the polygon points offset to the polygon position.

9.219 PolygonPathFinder

Inherits: *Resource* **<** *Reference* **<** *Object*

Category: Core

9.219.1 Brief Description

9.219.2 Member Functions

void	*setup* (*Vector2Array* points, *IntArray* connections)
Vector2Array	*find_path* (*Vector2* from, *Vector2* to)
Vector2Array	*get_intersections* (*Vector2* from, *Vector2* to) const
Vector2	*get_closest_point* (*Vector2* point) const
bool	*is_point_inside* (*Vector2* point) const
void	*set_point_penalty* (*int* idx, *float* penalty)
float	*get_point_penalty* (*int* idx) const
Rect2	*get_bounds* ( ) const

9.219.3 Member Function Description

- void **setup** (*Vector2Array* points, *IntArray* connections)
- *Vector2Array* **find_path** (*Vector2* from, *Vector2* to)
- *Vector2Array* **get_intersections** (*Vector2* from, *Vector2* to) const
- *Vector2* **get_closest_point** (*Vector2* point) const
- *bool* **is_point_inside** (*Vector2* point) const
- void **set_point_penalty** (*int* idx, *float* penalty)
- *float* **get_point_penalty** (*int* idx) const
- *Rect2* **get_bounds** ( ) const

9.220 Popup

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *PopupPanel*, *PopupDialog*, *PopupMenu*, *WindowDialog*

Category: Core

9.220.1 Brief Description

Base container control for popups and dialogs.

9.220.2 Member Functions

void	*popup_centered* (*Vector2* size=Vector2(0,0))
void	*popup_centered_ratio* (*float* ratio=0.75)
void	*popup_centered_minsize* (*Vector2* minsize=Vector2(0,0))
void	*popup* ( )
void	*set_exclusive* (*bool* enable)
bool	*is_exclusive* ( ) const

9.220.3 Signals

- **popup_hide** ( )
- **about_to_show** ( )

9.220.4 Numeric Constants

- **NOTIFICATION_POST_POPUP = 80** — Notification sent right after the popup is shown.
- **NOTIFICATION_POPUP_HIDE = 81** — Notification sent right after the popup is hidden.

9.220.5 Description

Popup is a base *Control* used to show dialogs and popups. It's a subwindow and modal by default (see *Control*) and has helpers for custom popup behavior.

9.220.6 Member Function Description

- void **popup_centered** (*Vector2* size=Vector2(0,0))

Popup (show the control in modal form) in the center of the screen, at the current size, or at a size determined by "size".

- void **popup_centered_ratio** (*float* ratio=0.75)

Popup (show the control in modal form) in the center of the screen, scaled at a ratio of size of the screen.

- void **popup_centered_minsize** (*Vector2* minsize=Vector2(0,0))

Popup (show the control in modal form) in the center of the screen, ensuring the size is never smaller than `minsize`.

- void **popup** ( )

Popup (show the control in modal form).

- void **set_exclusive** (*bool* enable)

Make the popup hide other popups when shown on the screen.

- *bool* **is_exclusive** ( ) const

Returns whether the popup will hide other popups when shown on the screen.

9.221 PopupDialog

Inherits: *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.221.1 Brief Description

Base class for Popup Dialogs.

9.221.2 Description

PopupDialog is a base class for popup dialogs, along with *WindowDialog*.

9.222 PopupMenu

Inherits: *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.222.1 Brief Description

PopupMenu displays a list of options.

9.222.2 Member Functions

void	*add_icon_item* (*Object* texture, *String* label, *int* id=-1, *int* accel=0)
void	*add_item* (*String* label, *int* id=-1, *int* accel=0)
void	*add_icon_check_item* (*Object* texture, *String* label, *int* id=-1, *int* accel=0)
void	*add_check_item* (*String* label, *int* id=-1, *int* accel=0)
void	*add_submenu_item* (*String* label, *String* submenu, *int* id=-1)
void	*set_item_text* (*int* idx, *String* text)
void	*set_item_icon* (*int* idx, *Object* icon)
void	*set_item_accelerator* (*int* idx, *int* accel)
void	*set_item_metadata* (*int* idx, var metadata)
void	*set_item_checked* (*int* idx, *bool* checked)
void	*set_item_disabled* (*int* idx, *bool* disabled)
void	*set_item_submenu* (*int* idx, *String* submenu)
void	*set_item_as_separator* (*int* idx, *bool* enable)
void	*set_item_as_checkable* (*int* idx, *bool* enable)
void	*set_item_ID* (*int* idx, *int* id)
String	*get_item_text* (*int* idx) const
Object	*get_item_icon* (*int* idx) const
void	*get_item_metadata* (*int* idx) const
int	*get_item_accelerator* (*int* idx) const
String	*get_item_submenu* (*int* idx) const
bool	*is_item_separator* (*int* idx) const
bool	*is_item_checkable* (*int* idx) const
bool	*is_item_checked* (*int* idx) const
bool	*is_item_disabled* (*int* idx) const
int	*get_item_ID* (*int* idx) const
int	*get_item_index* (*int* id) const
int	*get_item_count* ( ) const
void	*add_separator* ( )
void	*remove_item* (*int* idx)
void	*clear* ( )

9.222.3 Signals

- **item_pressed** (*int* ID)

9.222.4 Description

PopupMenu is the typical Control that displays a list of options. They are popular in toolbars or context menus.

9.222.5 Member Function Description

- void **add_icon_item** (*Object* texture, *String* label, *int* id=-1, *int* accel=0)

Add a new item with text "label" and icon "texture". An id can optionally be provided, as well as an accelerator keybinding. If no id is provided, one will be created from the index.

- void **add_item** (*String* label, *int* id=-1, *int* accel=0)

Add a new item with text "label". An id can optionally be provided, as well as an accelerator keybinding. If no id is provided, one will be created from the index.

- void **add_icon_check_item** (*Object* texture, *String* label, *int* id=-1, *int* accel=0)

Add a new checkable item with text "label" and icon "texture". An id can optionally be provided, as well as an accelerator. If no id is provided, one will be created from the index. Note that checkable items just display a checkmark, but don't have any built-in checking behavior and must be checked/unchecked manually.

- void **add_check_item** (*String* label, *int* id=-1, *int* accel=0)

Add a new checkable item with text "label". An id can optionally be provided, as well as an accelerator. If no id is provided, one will be created from the index. Note that checkable items just display a checkmark, but don't have any built-in checking behavior and must be checked/unchecked manually.

- void **add_submenu_item** (*String* label, *String* submenu, *int* id=-1)

Adds an item with a submenu. The submenu is the name of a child PopupMenu node that would be shown when the item is clicked. An id can optionally be provided, but if is isn't provided, one will be created from the index.

- void **set_item_text** (*int* idx, *String* text)

Set the text of the item at index "idx".

- void **set_item_icon** (*int* idx, *Object* icon)

Set the icon of the item at index "idx".

- void **set_item_accelerator** (*int* idx, *int* accel)

Set the accelerator of the item at index "idx". Accelerators are special combinations of keys that activate the item, no matter which control is focused.

- void **set_item_metadata** (*int* idx, var metadata)

Sets the metadata of an item, which might be of any type. You can later get it with *get_item_metadata*, which provides a simple way of assigning context data to items.

- void **set_item_checked** (*int* idx, *bool* checked)

Set the checkstate status of the item at index "idx".

- void **set_item_disabled** (*int* idx, *bool* disabled)

Sets whether the item at index "idx" is disabled or not. When it is disabled it can't be selected, or its action invoked.

- void **set_item_submenu** (*int* idx, *String* submenu)

Sets the submenu of the item at index "idx". The submenu is the name of a child PopupMenu node that would be shown when the item is clicked.

- void **set_item_as_separator** (*int* idx, *bool* enable)

Mark the item at index "idx" as a seperator, which means that it would be displayed as a mere line.

- void **set_item_as_checkable** (*int* idx, *bool* enable)

Set whether the item at index "idx" has a checkbox. Note that checkable items just display a checkmark, but don't have any built-in checking behavior and must be checked/unchecked manually.

- void **set_item_ID** (*int* idx, *int* id)

Set the id of the item at index "idx".

- *String* **get_item_text** (*int* idx) const

Return the text of the item at index "idx".

- *Object* **get_item_icon** (*int* idx) const

Return the icon of the item at index "idx".

- void **get_item_metadata** (*int* idx) const

Return the metadata of an item, which might be of any type. You can set it with *set_item_metadata*, which provides a simple way of assigning context data to items.

- *int* **get_item_accelerator** (*int* idx) const

Return the accelerator of the item at index "idx". Accelerators are special combinations of keys that activate the item, no matter which control is focused.

- *String* **get_item_submenu** (*int* idx) const

Return the submenu name of the item at index "idx".

- *bool* **is_item_separator** (*int* idx) const

Return whether the item is a seperator. If it is, it would be displayed as a line.

- *bool* **is_item_checkable** (*int* idx) const

Return whether the item at index "idx" has a checkbox. Note that checkable items just display a checkmark, but don't have any built-in checking behavior and must be checked/unchecked manually.

- *bool* **is_item_checked** (*int* idx) const

Return the checkstate status of the item at index "idx".

- *bool* **is_item_disabled** (*int* idx) const

Return whether the item at index "idx" is disabled. When it is disabled it can't be selected, or its action invoked.

- *int* **get_item_ID** (*int* idx) const

Return the id of the item at index "idx".

- *int* **get_item_index** (*int* id) const

Find and return the index of the item containing a given id.

- *int* **get_item_count** ( ) const

Return the amount of items.

- void **add_separator** ( )

Add a separator between items. Separators also occupy an index.

- void **remove_item** (*int* idx)

Removes the item at index "idx" from the menu. Note that the indexes of items after the removed item are going to be shifted by one.

- void **clear** ( )

Clear the popup menu, in effect removing all items.

9.223 PopupPanel

Inherits: *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.223.1 Brief Description

Class for displaying popups with a panel background.

9.223.2 Description

Class for displaying popups with a panel background. In some cases it might be simpler to use than *Popup*, since it provides a configurable background. If you are making windows, better check *WindowDialog*.

9.224 Portal

Inherits: *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.224.1 Brief Description

Portals provide virtual openings to rooms.

9.224.2 Member Functions

void	*set_shape* (*Vector2Array* points)
Vector2Array	*get_shape* ( ) const
void	*set_enabled* (*bool* enable)
bool	*is_enabled* ( ) const
void	*set_disable_distance* (*float* distance)
float	*get_disable_distance* ( ) const
void	*set_disabled_color* (*Color* color)
Color	*get_disabled_color* ( ) const
void	*set_connect_range* (*float* range)
float	*get_connect_range* ( ) const

9.224.3 Description

Portals provide virtual openings to *VisualInstance* nodes, so cameras can look at them from the outside. Note that portals are a visibility optimization technique, and are in no way related to the game of the same name (as in, they are not used for teleportation). For more information on how rooms and portals work, see *VisualInstance*. Portals are represented as 2D convex polygon shapes (in the X,Y local plane), and are placed on the surface of the areas occupied by a *VisualInstance*, to indicate that the room can be accessed or looked-at through them. If two rooms are next to each other, and two similar portals in each of them share the same world position (and are parallel and opposed to each other), they will automatically "connect" and form "doors" (for example, the portals that connect a kitchen to a living room are placed in the door they share). Portals must always have a *VisualInstance* node as a parent, grandparent or far parent, or else they will not be active.

9.224.4 Member Function Description

- void **set_shape** (*Vector2Array* points)

Set the portal shape. The shape is an array of *Vector2* points, representing a convex polygon in the X,Y plane.

- *Vector2Array* **get_shape** ( ) const

Return the portal shape. The shape is an array of *Vector2* points, representing a convex polygon in the X,Y plane.

- void **set_enabled** (*bool* enable)

Enable the portal (it is enabled by default though), disabling it will cause the parent *VisualInstance* to not be visible any longer when looking through the portal.

- *bool* **is_enabled** ( ) const

Return whether the portal is active. When disabled it causes the parent *VisualInstance* to not be visible any longer when looking through the portal.

- void **set_disable_distance** (*float* distance)

Set the distance threshold for disabling the portal. Every time that the portal goes beyond "distance", it disables itself, becoming the opaque color (see *set_disabled_color*).

- *float* **get_disable_distance** ( ) const

Return the distance threshold for disabling the portal. Every time that the portal goes beyond "distance", it disables itself, becoming the opaque color (see *set_disabled_color*).

- void **set_disabled_color** (*Color* color)

When the portal goes beyond the disable distance (see *set_disable_distance*), it becomes opaque and displayed with color "color".

- *Color* **get_disabled_color** ( ) const

Return the color for when the portal goes beyond the disable distance (see *set_disable_distance*) and becomes disabled.

- void **set_connect_range** (*float* range)

Set the range for auto-connecting two portals from different rooms sharing the same space.

- *float* **get_connect_range** ( ) const

Return the range for auto-connecting two portals from different rooms sharing the same space.

9.225 Position2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.225.1 Brief Description

Generic 2D Position hint for editing.

9.225.2 Description

Generic 2D Position hint for editing. It's just like a plain *Node2D* but displays as a cross in the 2D-Editor at all times.

9.226 Position3D

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.226.1 Brief Description

Generic 3D Position hint for editing

9.226.2 Description

Generic 3D Position hint for editing. It's just like a plain *Spatial* but displays as a cross in the 3D-Editor at all times.

9.227 ProgressBar

Inherits: *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.227.1 Brief Description

General purpose progress bar.

9.227.2 Member Functions

void	*set_percent_visible* (*bool* visible)
bool	*is_percent_visible* ( ) const

9.227.3 Description

General purpose progress bar. Shows fill percentage from right to left.

9.227.4 Member Function Description

- void **set_percent_visible** (*bool* visible)
- *bool* **is_percent_visible** ( ) const

9.228 ProximityGroup

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.228.1 Brief Description

General purpose proximity-detection node.

9.228.2 Member Functions

void	*set_group_name* (*String* name)
void	*broadcast* (*String* name, var parameters)
void	*set_dispatch_mode* (*int* mode)
void	*set_grid_radius* (*Vector3* radius)
Vector3	*get_grid_radius* ( ) const

9.228.3 Signals

- **broadcast** (*String* name, *Array* parameters)

9.228.4 Description

General purpose proximity-detection node.

9.228.5 Member Function Description

- void **set_group_name** (*String* name)
- void **broadcast** (*String* name, var parameters)
- void **set_dispatch_mode** (*int* mode)
- void **set_grid_radius** (*Vector3* radius)
- *Vector3* **get_grid_radius** ( ) const

9.229 Quad

Inherits: *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.229.1 Brief Description

9.229.2 Member Functions

void	*set_axis* (*int* axis)
int	*get_axis* ( ) const
void	*set_size* (*Vector2* size)
Vector2	*get_size* ( ) const
void	*set_centered* (*bool* centered)
bool	*is_centered* ( ) const
void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const

9.229.3 Member Function Description

- void **set_axis** (*int* axis)
- *int* **get_axis** ( ) const
- void **set_size** (*Vector2* size)
- *Vector2* **get_size** ( ) const
- void **set_centered** (*bool* centered)
- *bool* **is_centered** ( ) const
- void **set_offset** (*Vector2* offset)
- *Vector2* **get_offset** ( ) const

9.230 Quat

Category: Built-In Types

9.230.1 Brief Description

Quaternion.

9.230.2 Member Functions

Quat	*cubic_slerp* (*Quat* b, *Quat* pre_a, *Quat* post_b, *float* t)
float	*dot* (*Quat* b)
Quat	*inverse* ( )
float	*length* ( )
float	*length_squared* ( )
Quat	*normalized* ( )
Quat	*slerp* (*Quat* b, *float* t)
Quat	*slerpni* (*Quat* b, *float* t)
Vector3	*xform* (*Vector3* v)
Quat	Quat (*float* x, *float* y, *float* z, *float* w)
Quat	Quat (*Vector3* axis, *float* angle)
Quat	Quat (*Matrix3* from)

9.230.3 Member Variables

- *float* **x**
- *float* **y**
- *float* **z**
- *float* **w**

9.230.4 Description

Quaternion is a 4 dimensional vector that is used to represent a rotation. It mainly exists to perform SLERP (spherical-linear interpolation) between to rotations obtained by a Matrix3 cheaply. Adding quaternions also cheaply adds the rotations, however quaternions need to be often normalized, or else they suffer from precision issues.

9.230.5 Member Function Description

- *Quat* **cubic_slerp** (*Quat* b, *Quat* pre_a, *Quat* post_b, *float* t)
- *float* **dot** (*Quat* b)

Returns the dot product between two quaternions.

- *Quat* **inverse** ( )

Returns the inverse of the quaternion (applies to the inverse rotation too).

- *float* **length** ( )

Returns the length of the quaternion.

- *float* **length_squared** ( )

Returns the length of the quaternion, squared.

- *Quat* **normalized** ( )

Returns a copy of the quaternion, normalized to unit length.

- *Quat* **slerp** (*Quat* b, *float* t)

Perform a spherical-linear interpolation with another quaternion.

- *Quat* **slerpni** (*Quat* b, *float* t)
- *Vector3* **xform** (*Vector3* v)
- *Quat* **Quat** (*float* x, *float* y, *float* z, *float* w)
- *Quat* **Quat** (*Vector3* axis, *float* angle)
- *Quat* **Quat** (*Matrix3* from)

9.231 Range

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *SpinBox*, *ScrollBar*, *ProgressBar*, *TextureProgress*, *Slider*

Category: Core

9.231.1 Brief Description

Abstract base class for range-based controls.

9.231.2 Member Functions

float	*get_val* ( ) const
float	*get_value* ( ) const
float	*get_min* ( ) const
float	*get_max* ( ) const
float	*get_step* ( ) const
float	*get_page* ( ) const
float	*get_unit_value* ( ) const
void	*set_val* (*float* value)
void	*set_value* (*float* value)
void	*set_min* (*float* minimum)
void	*set_max* (*float* maximum)
void	*set_step* (*float* step)
void	*set_page* (*float* pagesize)
void	*set_unit_value* (*float* value)
void	*set_rounded_values* (*bool* enabled)
bool	*is_rounded_values* ( ) const
void	*set_exp_unit_value* (*bool* enabled)
bool	*is_unit_value_exp* ( ) const
void	*share* (*Object* with)
void	*unshare* ( )

9.231.3 Signals

- **value_changed** (*float* value)
- **changed** ( )

9.231.4 Description

Range is a base class for *Control* nodes that change a floating point *value* between a *minimum* and a *maximum*, using *step* and *page*, for example a *ScrollBar*.

9.231.5 Member Function Description

- *float* **get_val** ( ) const

Return the current value.

- *float* **get_value** ( ) const
- *float* **get_min** ( ) const

Return the minimum value.

- *float* **get_max** ( ) const

Return the maximum value.

- *float* **get_step** ( ) const

Return the stepping, if step is 0, stepping is disabled.

- *float* **get_page** ( ) const

Return the page size, if page is 0, paging is disabled.

- *float* **get_unit_value** ( ) const

Return value mapped to 0 to 1 (unit) range.

- void **set_val** (*float* value)
- void **set_value** (*float* value)
- void **set_min** (*float* minimum)

Set minimum value, clamped range value to it if it's less.

- void **set_max** (*float* maximum)
- void **set_step** (*float* step)

Set step value. If step is 0, stepping will be disabled.

- void **set_page** (*float* pagesize)

Set page size. Page is mainly used for scrollbars or anything that controls text scrolling.

- void **set_unit_value** (*float* value)

Set value mapped to 0 to 1 (unit) range, it will then be converted to the actual value within min and max.

- void **set_rounded_values** (*bool* enabled)
- *bool* **is_rounded_values** ( ) const
- void **set_exp_unit_value** (*bool* enabled)
- *bool* **is_unit_value_exp** ( ) const
- void **share** (*Object* with)
- void **unshare** ( )

9.232 RawArray

Category: Built-In Types

9.232.1 Brief Description

Raw byte array.

9.232.2 Member Functions

String	*get_string_from_ascii* ( )
String	*get_string_from_utf8* ( )
void	*push_back* (*int* byte)
void	*resize* (*int* idx)
void	*set* (*int* idx, *int* byte)
int	*size* ( )
RawArray	*RawArray* (*Array* from)

9.232.3 Description

Raw byte array. Contains bytes. Optimized for memory usage, can't fragment the memory.

9.232.4 Member Function Description

- *String* **get_string_from_ascii** ( )

Returns a copy of the array's contents formatted as String. Fast alternative to get_string_from_utf8(), assuming the content is ASCII-only (unlike the UTF-8 function, this function maps every byte to a character in the string, so any multibyte sequence will be torn apart).

- *String* **get_string_from_utf8** ( )

Returns a copy of the array's contents formatted as String, assuming the array is formatted as UTF-8. Slower than get_string_from_ascii(), but works for UTF-8. Usually you should prefer this function over get_string_from_ascii() to support international input.

- void **push_back** (*int* byte)
- void **resize** (*int* idx)
- void **set** (*int* idx, *int* byte)
- *int* **size** ( )
- *RawArray* **RawArray** (*Array* from)

9.233 RayCast

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.233.1 Brief Description

9.233.2 Member Functions

void	*set_enabled* (*bool* enabled)
bool	*is_enabled* ( ) const
void	*set_cast_to* (*Vector3* local_point)
Vector3	*get_cast_to* ( ) const
bool	*is_colliding* ( ) const
Object	*get_collider* ( ) const
int	*get_collider_shape* ( ) const
Vector3	*get_collision_point* ( ) const
Vector3	*get_collision_normal* ( ) const
void	*add_exception_rid* (*RID* rid)
void	*add_exception* (*Object* node)
void	*remove_exception_rid* (*RID* rid)
void	*remove_exception* (*Object* node)
void	*clear_exceptions* ( )

9.233.3 Member Function Description

- void **set_enabled** (*bool* enabled)
- *bool* **is_enabled** ( ) const
- void **set_cast_to** (*Vector3* local_point)
- *Vector3* **get_cast_to** ( ) const
- *bool* **is_colliding** ( ) const
- *Object* **get_collider** ( ) const
- *int* **get_collider_shape** ( ) const
- *Vector3* **get_collision_point** ( ) const
- *Vector3* **get_collision_normal** ( ) const
- void **add_exception_rid** (*RID* rid)
- void **add_exception** (*Object* node)
- void **remove_exception_rid** (*RID* rid)
- void **remove_exception** (*Object* node)
- void **clear_exceptions** ( )

9.234 RayCast2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.234.1 Brief Description

Query the closest object intersecting a ray

9.234.2 Member Functions

void	*set_enabled* (*bool* enabled)
bool	*is_enabled* ( ) const
void	*set_cast_to* (*Vector2* local_point)
Vector2	*get_cast_to* ( ) const
bool	*is_colliding* ( ) const
Object	*get_collider* ( ) const
int	*get_collider_shape* ( ) const
Vector2	*get_collision_point* ( ) const
Vector2	*get_collision_normal* ( ) const
void	*add_exception_rid* (*RID* rid)
void	*add_exception* (*Object* node)
void	*remove_exception_rid* (*RID* rid)
void	*remove_exception* (*Object* node)
void	*clear_exceptions* ( )
void	*set_layer_mask* (*int* mask)
int	*get_layer_mask* ( ) const
void	*set_type_mask* (*int* mask)
int	*get_type_mask* ( ) const

9.234.3 Description

A RayCast2D represents a line from its origin to its destination position `cast_to`, it is used to query the 2D space in order to find the closest object intersecting with the ray.

9.234.4 Member Function Description

- void **set_enabled** (*bool* enabled)

Enables the RayCast2D. Only enabled raycasts will be able to query the space and report collisions.

- *bool* **is_enabled** ( ) const

Returns whether this raycast is enabled or not

- void **set_cast_to** (*Vector2* local_point)

Sets the ray destination point, so that the ray will test from the ray's origin to `local_point`

- *Vector2* **get_cast_to** ( ) const

Return the destination point of this ray object

- *bool* **is_colliding** ( ) const

Return whether the closest object the ray is pointing to is colliding with the vector (considering the vector length).

- *Object* **get_collider** ( ) const

Return the closest object the ray is pointing to. Note that this does not consider the length of the vector, so you must also use *is_colliding* to check if the object returned is actually colliding with the ray.

- *int* **get_collider_shape** **(** **)** const

Returns the collision shape of the closest object the ray is pointing to.

- *Vector2* **get_collision_point** **(** **)** const

Returns the collision point in which the ray intersects the closest object.

- *Vector2* **get_collision_normal** **(** **)** const

Returns the normal of the intersecting object shape face containing the collision point.

- void **add_exception_rid** **(** *RID* rid **)**
- void **add_exception** **(** *Object* node **)**

Adds a collision exception so the ray does not report collisions with the specified `node`.

- void **remove_exception_rid** **(** *RID* rid **)**
- void **remove_exception** **(** *Object* node **)**

Removes a collision exception so the ray does report collisions with the specified `node`.

- void **clear_exceptions** **(** **)**

Removes all collision exception for this ray.

- void **set_layer_mask** **(** *int* mask **)**
- *int* **get_layer_mask** **(** **)** const

Returns the layer mask for this ray.

- void **set_type_mask** **(** *int* mask **)**
- *int* **get_type_mask** **(** **)** const

9.235 RayShape

Inherits: *Shape* **<** *Resource* **<** *Reference* **<** *Object*

Category: Core

9.235.1 Brief Description

9.235.2 Member Functions

void	*set_length* **(** *float* length **)**
float	*get_length* **(** **)** const

9.235.3 Member Function Description

- void **set_length** **(** *float* length **)**
- *float* **get_length** **(** **)** const

9.236 RayShape2D

Inherits: *Shape2D* < *Resource* < *Reference* < *Object*

Category: Core

9.236.1 Brief Description

Ray 2D shape resource for physics.

9.236.2 Member Functions

void	*set_length* (*float* length)
float	*get_length* ( ) const

9.236.3 Description

Ray 2D shape resource for physics. A ray is not really a collision body, instead it tries to separate itself from whatever is touching its far endpoint. It's often useful for characters.

9.236.4 Member Function Description

- void **set_length** (*float* length)

Set the length of the ray.

- *float* **get_length** ( ) const

Return the length of the ray.

9.237 RealArray

Category: Built-In Types

9.237.1 Brief Description

Real Array .

9.237.2 Member Functions

void	*push_back* (*float* value)
void	*resize* (*int* idx)
void	*set* (*int* idx, *float* value)
int	*size* ( )
RealArray	*RealArray* (*Array* from)

9.237.3 Description

Real Array. Array of floating point values. Can only contain floats. Optimized for memory usage, can't fragment the memory.

9.237.4 Member Function Description

- void **push_back** (*float* value)
- void **resize** (*int* idx)
- void **set** (*int* idx, *float* value)
- *int* **size** ( )
- *RealArray* **RealArray** (*Array* from)

9.238 Rect2

Category: Built-In Types

9.238.1 Brief Description

2D Axis-aligned bounding box.

9.238.2 Member Functions

Rect2	*clip* (*Rect2* b)
bool	*encloses* (*Rect2* b)
Rect2	*expand* (*Vector2* to)
float	*get_area* ( )
Rect2	*grow* (*float* by)
bool	*has_no_area* ( )
bool	*has_point* (*Vector2* point)
bool	*intersects* (*Rect2* b)
Rect2	*merge* (*Rect2* b)
Rect2	Rect2 (*Vector2* pos, *Vector2* size)
Rect2	Rect2 (*float* x, *float* y, *float* width, *float* height)

9.238.3 Member Variables

- *Vector2* **pos** - Position (starting corner).
- *Vector2* **size** - Size from position to end.
- *Vector2* **end** - Ending corner.

9.238.4 Description

Rect2 provides an 2D Axis-Aligned Bounding Box. It consists of a position, a size, and several utility functions. It is typically used for fast overlap tests.

9.238.5 Member Function Description

- *Rect2* **clip** (*Rect2* b)

Returns the intersection of this *Rect2* and b.

- *bool* **encloses** (*Rect2* b)

Returns true if this *Rect2* completely encloses another one.

- *Rect2* **expand** (*Vector2* to)

Return this *Rect2* expanded to include a given point.

- *float* **get_area** ( )

Get the area of the *Rect2*.

- *Rect2* **grow** (*float* by)

Return a copy of the *Rect2* grown a given amount of units towards all the sides.

- *bool* **has_no_area** ( )

Return true if the *Rect2* is flat or empty.

- *bool* **has_point** (*Vector2* point)

Return true if the *Rect2* contains a point.

- *bool* **intersects** (*Rect2* b)

Return true if the *Rect2* overlaps with another.

- *Rect2* **merge** (*Rect2* b)

Combine this *Rect2* with another, a larger one is returned that contains both.

- *Rect2* **Rect2** (*Vector2* pos, *Vector2* size)

Construct a *Rect2* by position and size.

- *Rect2* **Rect2** (*float* x, *float* y, *float* width, *float* height)

Construct a *Rect2* by x, y, width and height.

9.239 RectangleShape2D

Inherits: *Shape2D* < *Resource* < *Reference* < *Object*

Category: Core

9.239.1 Brief Description

Rectangle Shape for 2D Physics.

9.239.2 Member Functions

void	*set_extents* (*Vector2* extents)
Vector2	*get_extents* ( ) const

9.239.3 Description

Rectangle Shape for 2D Physics. This shape is useful for modeling box-like 2D objects.

9.239.4 Member Function Description

- void **set_extents** (*Vector2* extents)

Set the half extents, the actual width and height of this shape is twice the half extents.

- *Vector2* **get_extents** ( ) const

Return the half extents, the actual width and height of this shape is twice the half extents.

9.240 Reference

Inherits: *Object*

Inherited By: *RegEx*, *SurfaceTool*, *EditorScenePostImport*, *PhysicsShapeQueryResult*, *Physics2DTestMotionResult*, *FuncRef*, *File*, *TCP_Server*, *Physics2DShapeQueryResult*, *ConfigFile*, *StreamPeer*, *HTTPClient*, *AudioStreamPlayback*, *MeshDataTool*, *GDFunctionState*, *Physics2DShapeQueryParameters*, *EditorScript*, *Mutex*, *PacketPeer*, *Semaphore*, *XMLParser*, *EditorImportPlugin*, *Directory*, *Marshalls*, *WeakRef*, *SceneState*, *GDNativeClass*, *PCKPacker*, *Resource*, *Thread*, *PackedDataContainerRef*, *ResourceInteractiveLoader*, *ResourceImportMetadata*, *PhysicsShapeQueryParameters*

Category: Core

9.240.1 Brief Description

Base class for anything that keeps a reference count.

9.240.2 Member Functions

bool	*init_ref* ( )
void	*reference* ( )
bool	*unreference* ( )

9.240.3 Description

Base class for anything that keeps a reference count. Resource and many other helper objects inherit this. References keep an internal reference counter so they are only released when no longer in use.

9.240.4 Member Function Description

- *bool* **init_ref** ( )
- void **reference** ( )

Increase the internal reference counter. Use this only if you really know what you are doing.

- *bool* **unreference** ( )

Decrease the internal reference counter. Use this only if you really know what you are doing.

9.241 ReferenceFrame

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.241.1 Brief Description

Reference frame for GUI.

9.241.2 Description

Reference frame for GUI. It's just like an empty control, except a red box is displayed while editing around its size at all times.

9.242 RegEx

Inherits: *Reference* < *Object*

Category: Core

9.242.1 Brief Description

Simple regular expression matcher.

9.242.2 Member Functions

int	*compile* (*String* pattern, *int* capture=9)
int	*find* (*String* text, *int* start=0, *int* end=-1) const
void	*clear* ( )
bool	*is_valid* ( ) const
int	*get_capture_count* ( ) const
String	*get_capture* (*int* capture) const
int	*get_capture_start* (*int* capture) const
StringArray	*get_captures* ( ) const

9.242.3 Description

Class for finding text patterns in a string using regular expressions. Regular expressions are a way to define patterns of text to be searched.

This class only finds patterns in a string. It can not perform replacements.

Usage of regular expressions is too long to be explained here, but Internet is full of tutorials and detailed explanations.

Currently supported features:

Capturing `()` and non-capturing `(?:)` groups

Any character `.`

Shorthand character classes `\w \W \s \S \d \D`

User-defined character classes such as `:ref:`A-Za-z<class_a-za-z>``

Simple quantifiers `?`, `\*` and `+`

Range quantifiers `{x,y}`

Lazy (non-greedy) quantifiers `\*?`

Beginning `^` and end `$` anchors

Alternation `|`

Backreferences `\1` and `\g{1}`

POSIX character classes `:ref:`[:alnum:<class_[:alnum:>`]`

Lookahead `(?=)`, `(?!)` and lookbehind `(?<=)`, `(?<!)`

ASCII `\xFF` and Unicode `\uFFFF` code points (in a style similar to Python)

Word boundaries `\b`, `\B`

9.242.4 Member Function Description

- *int* **compile** (*String* pattern, *int* capture=9)

Compiles and assign the regular expression pattern to use. The limit on the number of capturing groups can be specified or made unlimited if negative.

- *int* **find** (*String* text, *int* start=0, *int* end=-1) const

This method tries to find the pattern within the string, and returns the position where it was found. It also stores any capturing group (see *get_capture*) for further retrieval.

- void **clear** ( )

This method resets the state of the object, as it was freshly created. Namely, it unassigns the regular expression of this object, and forgets all captures made by the last *find*.

- *bool* **is_valid** ( ) const

Returns whether this object has a valid regular expression assigned.

- *int* **get_capture_count** ( ) const

Returns the number of capturing groups. A captured group is the part of a string that matches a part of the pattern delimited by parentheses (unless they are non-capturing parentheses *(?:)*).

- *String* **get_capture** (*int* capture) const

Returns a captured group. A captured group is the part of a string that matches a part of the pattern delimited by parentheses (unless they are non-capturing parentheses *(?:)*).

- *int* **get_capture_start** (*int* capture) const

- *StringArray* **get_captures** ( ) const

Return a list of all the captures made by the regular expression.

9.243 RemoteTransform2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.243.1 Brief Description

9.243.2 Member Functions

void	*set_remote_node* (*NodePath* path)
NodePath	*get_remote_node* ( ) const

9.243.3 Member Function Description

- void **set_remote_node** (*NodePath* path)
- *NodePath* **get_remote_node** ( ) const

9.244 RenderTargetTexture

Inherits: *Texture* < *Resource* < *Reference* < *Object*

Category: Core

9.244.1 Brief Description

9.245 Resource

Inherits: *Reference* < *Object*

Inherited By: *Theme*, *AudioStream*, *EventStream*, *CubeMap*, *Translation*, *Curve2D*, *Shape*, *Shape2D*, *BakedLight*, *ColorRamp*, *StyleBox*, *Environment*, *Material*, *VideoStream*, *RoomBounds*, *PackedScene*, *Texture*, *Script*, *OccluderPolygon2D*, *Mesh*, *TileSet*, *BitMap*, *Animation*, *Sample*, *PolygonPathFinder*, *Shader*, *World*, *SampleLibrary*, *World2D*, *Font*, *SpriteFrames*, *MeshLibrary*, *Curve3D*, *NavigationPolygon*, *MultiMesh*, *CanvasItemMaterial*, *PackedDataContainer*, *NavigationMesh*

Category: Core

9.245.1 Brief Description

Base class for all resources.

9.245.2 Member Functions

void	*set_path* (*String* path)
void	*take_over_path* (*String* path)
String	*get_path* ( ) const
void	*set_name* (*String* name)
String	*get_name* ( ) const
RID	*get_rid* ( ) const
void	*set_import_metadata* (*Object* metadata)
Object	*get_import_metadata* ( ) const
Object	*duplicate* (*bool* subresources=false)

9.245.3 Signals

- **changed** ( )

9.245.4 Description

Resource is the base class for all resource types. Resources are primarily data containers. They are reference counted and freed when no longer in use. They are also loaded only once from disk, and further attempts to load the resource will return the same reference (all this in contrast to a *Node*, which is not reference counted and can be instanced from disk as many times as desired). Resources can be saved externally on disk or bundled into another object, such as a *Node* or another resource.

9.245.5 Member Function Description

- void **set_path** (*String* path)

Set the path of the resource. This is useful mainly for editors when saving/loading, and shouldn't be changed by anything else. Fails if another *Resource* already has path "path".

- void **take_over_path** (*String* path)

Set the path of the resource. Differs from set_path(), if another *Resource* exists with "path" it over-takes it, instead of failing.

- *String* **get_path** ( ) const

Return the path of the resource. This is useful mainly for editors when saving/loading, and shouldn't be changed by anything else.

- void **set_name** (*String* name)

Set the name of the resources, any name is valid (it doesn't have to be unique). Name is for descriptive purposes only.

- *String* **get_name** ( ) const

Return the name of the resources, any name is valid (it doesn't have to be unique). Name is for descriptive purposes only.

- *RID* **get_rid** ( ) const

Return the RID of the resource (or an empty RID). Many resources (such as *Texture*, *Mesh*, etc) are high level abstractions of resources stored in a server, so this function will return the original RID.

- void **set_import_metadata** (*Object* metadata)

- *Object* **get_import_metadata** ( ) const
- *Object* **duplicate** (*bool* subresources=false)

9.246 ResourceImportMetadata

Inherits: *Reference* < *Object*

Category: Core

9.246.1 Brief Description

9.246.2 Member Functions

void	*set_editor* (*String* name)
String	*get_editor* ( ) const
void	*add_source* (*String* path, *String* md5="")
String	*get_source_path* (*int* idx) const
String	*get_source_md5* (*int* idx) const
void	*remove_source* (*int* idx)
int	*get_source_count* ( ) const
void	*set_option* (*String* key, var value)
void	*get_option* (*String* key) const
StringArray	*get_options* ( ) const

9.246.3 Member Function Description

- void **set_editor** (*String* name)
- *String* **get_editor** ( ) const
- void **add_source** (*String* path, *String* md5="")
- *String* **get_source_path** (*int* idx) const
- *String* **get_source_md5** (*int* idx) const
- void **remove_source** (*int* idx)
- *int* **get_source_count** ( ) const
- void **set_option** (*String* key, var value)
- void **get_option** (*String* key) const
- *StringArray* **get_options** ( ) const

9.247 ResourceInteractiveLoader

Inherits: *Reference* < *Object*

Category: Core

9.247.1 Brief Description

Interactive Resource Loader.

9.247.2 Member Functions

Object	*get_resource* ( )
int	*poll* ( )
int	*wait* ( )
int	*get_stage* ( ) const
int	*get_stage_count* ( ) const

9.247.3 Description

Interactive Resource Loader. This object is returned by ResourceLoader when performing an interactive load. It allows to load with high granularity, so this is mainly useful for displaying load bars/percentages.

9.247.4 Member Function Description

- *Object* **get_resource** ( )

Return the loaded resource (only if loaded). Otherwise, returns null.

- *int* **poll** ( )

Poll the load. If OK is returned, this means poll will have to be called again. If ERR_EOF is returned, them the load has finished and the resource can be obtained by calling *get_resource*.

- *int* **wait** ( )
- *int* **get_stage** ( ) const

Return the load stage. The total amount of stages can be queried with *get_stage_count*

- *int* **get_stage_count** ( ) const

Return the total amount of stages (calls to *poll*) needed to completely load this resource.

9.248 ResourceLoader

Inherits: *Object*

Category: Core

9.248.1 Brief Description

Resource Loader.

9.248.2 Member Functions

ResourceInteractiveLoader	*load_interactive* (*String* path, *String* type_hint="")
Resource	*load* (*String* path, *String* type_hint="", *bool* p_no_cache=false)
StringArray	*get_recognized_extensions_for_type* (*String* type)
void	*set_abort_on_missing_resources* (*bool* abort)
StringArray	*get_dependencies* (*String* path)
bool	*has* (*String* path)

9.248.3 Description

Resource Loader. This is a static object accessible as *ResourceLoader*. GDScript has a simplified load() function, though.

9.248.4 Member Function Description

- *ResourceInteractiveLoader* **load_interactive** (*String* path, *String* type_hint="")

Load a resource interactively, the returned object allows to load with high granularity.

- *Resource* **load** (*String* path, *String* type_hint="", *bool* p_no_cache=false)
- *StringArray* **get_recognized_extensions_for_type** (*String* type)

Return the list of recognized extensions for a resource type.

- void **set_abort_on_missing_resources** (*bool* abort)

Change the behavior on missing sub-resources. Default is to abort load.

- *StringArray* **get_dependencies** (*String* path)
- *bool* **has** (*String* path)

9.249 ResourcePreloader

Inherits: *Node* < *Object*

Category: Core

9.249.1 Brief Description

Resource Preloader Node.

9.249.2 Member Functions

void	*add_resource* (*String* name, *Object* resource)
void	*remove_resource* (*String* name)
void	*rename_resource* (*String* name, *String* newname)
bool	*has_resource* (*String* name) const
Object	*get_resource* (*String* name) const
StringArray	*get_resource_list* ( ) const

9.249.3 Description

Resource Preloader Node. This node is used to preload sub-resources inside a scene, so when the scene is loaded all the resources are ready to use and be retrieved from here.

9.249.4 Member Function Description

- void **add_resource** (*String* name, *Object* resource)

Add a resource to the preloader. Set the text-id that will be used to identify it (retrieve it/erase it/etc).

- void **remove_resource** (*String* name)

Remove a resource from the preloader by text id.

- void **rename_resource** (*String* name, *String* newname)

Rename a resource inside the preloader, from a text-id to a new text-id.

- *bool* **has_resource** (*String* name) const

Return true if the preloader has a given resource.

- *Object* **get_resource** (*String* name) const

Return the resource given a text-id.

- *StringArray* **get_resource_list** ( ) const

Return the list of resources inside the preloader.

9.250 ResourceSaver

Inherits: *Object*

Category: Core

9.250.1 Brief Description

Resource Saving Interface.

9.250.2 Member Functions

int	*save* (*String* path, *Resource* resource, *int* flags=0)
StringArray	*get_recognized_extensions* (*Object* type)

9.250.3 Numeric Constants

- **FLAG_RELATIVE_PATHS = 1**
- **FLAG_BUNDLE_RESOURCES = 2**
- **FLAG_CHANGE_PATH = 4**
- **FLAG_OMIT_EDITOR_PROPERTIES = 8**

- **FLAG_SAVE_BIG_ENDIAN = 16**
- **FLAG_COMPRESS = 32**

9.250.4 Description

Resource Saving Interface. This interface is used for saving resources to disk.

9.250.5 Member Function Description

- *int* **save** (*String* path, *Resource* resource, *int* flags=0)

Save a resource to disk, to a given path.

- *StringArray* **get_recognized_extensions** (*Object* type)

Return the list of extensions available for saving a resource of a given type.

9.251 RichTextLabel

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.251.1 Brief Description

Label that displays rich text.

9.251.2 Member Functions

void	*add_text* (*String* text)
void	*add_image* (*Texture* image)
void	*newline* ( )
void	*push_font* (*Object* font)
void	*push_color* (*Color* color)
void	*push_align* (*int* align)
void	*push_indent* (*int* level)
void	*push_list* (*int* type)
void	*push_meta* (var data)
void	*push_underline* ( )
void	*push_table* (*int* columns)
void	*set_table_column_expand* (*int* column, *bool* expand, *int* ratio)
void	*push_cell* ( )
void	*pop* ( )
void	*clear* ( )
void	*set_meta_underline* (*bool* enable)
bool	*is_meta_underlined* ( ) const
void	*set_scroll_active* (*bool* active)
bool	*is_scroll_active* ( ) const

Continúa en la página siguiente

Tabla 9.22 – proviene de la página anterior

void	*set_scroll_follow* (*bool* follow)
bool	*is_scroll_following* ( ) const
Object	*get_v_scroll* ( )
void	*scroll_to_line* (*int* line)
void	*set_tab_size* (*int* spaces)
int	*get_tab_size* ( ) const
void	*set_selection_enabled* (*bool* enabled)
bool	*is_selection_enabled* ( ) const
int	*parse_bbcode* (*String* bbcode)
int	*append_bbcode* (*String* bbcode)
void	*set_bbcode* (*String* text)
String	*get_bbcode* ( ) const
void	*set_visible_characters* (*int* amount)
int	*get_visible_characters* ( ) const
int	*get_total_character_count* ( ) const
void	*set_use_bbcode* (*bool* enable)
bool	*is_using_bbcode* ( ) const

9.251.3 Signals

- **meta_clicked** (*Nil* meta)

9.251.4 Numeric Constants

- **ALIGN_LEFT = 0**
- **ALIGN_CENTER = 1**
- **ALIGN_RIGHT = 2**
- **ALIGN_FILL = 3**
- **LIST_NUMBERS = 0**
- **LIST_LETTERS = 1**
- **LIST_DOTS = 2**
- **ITEM_FRAME = 0**
- **ITEM_TEXT = 1**
- **ITEM_IMAGE = 2**
- **ITEM_NEWLINE = 3**
- **ITEM_FONT = 4**
- **ITEM_COLOR = 5**
- **ITEM_UNDERLINE = 6**
- **ITEM_ALIGN = 7**
- **ITEM_INDENT = 8**
- **ITEM_LIST = 9**
- **ITEM_META = 11**

9.251.5 Description

Label that displays rich text. Rich text can contain custom text, fonts, images and some basic formatting. It also adapts itself to given width/heights.

9.251.6 Member Function Description

- void **add_text** (*String* text)
- void **add_image** (*Texture* image)
- void **newline** ( )
- void **push_font** (*Object* font)
- void **push_color** (*Color* color)
- void **push_align** (*int* align)
- void **push_indent** (*int* level)
- void **push_list** (*int* type)
- void **push_meta** (var data)
- void **push_underline** ( )
- void **push_table** (*int* columns)
- void **set_table_column_expand** (*int* column, *bool* expand, *int* ratio)
- void **push_cell** ( )
- void **pop** ( )
- void **clear** ( )
- void **set_meta_underline** (*bool* enable)
- *bool* **is_meta_underlined** ( ) const
- void **set_scroll_active** (*bool* active)
- *bool* **is_scroll_active** ( ) const
- void **set_scroll_follow** (*bool* follow)
- *bool* **is_scroll_following** ( ) const
- *Object* **get_v_scroll** ( )
- void **scroll_to_line** (*int* line)
- void **set_tab_size** (*int* spaces)
- *int* **get_tab_size** ( ) const
- void **set_selection_enabled** (*bool* enabled)

Set to true if selecting the text inside this richtext is allowed.

- *bool* **is_selection_enabled** ( ) const

Return true if selecting the text inside this richtext is allowed.

- *int* **parse_bbcode** (*String* bbcode)
- *int* **append_bbcode** (*String* bbcode)

- void **set_bbcode** (*String* text)
- *String* **get_bbcode** ( ) const
- void **set_visible_characters** (*int* amount)
- *int* **get_visible_characters** ( ) const
- *int* **get_total_character_count** ( ) const
- void **set_use_bbcode** (*bool* enable)
- *bool* **is_using_bbcode** ( ) const

9.252 RID

Category: Built-In Types

9.252.1 Brief Description

9.252.2 Member Functions

int	*get_id* ( )
RID	*RID* (*Object* from)

9.252.3 Member Function Description

- *int* **get_id** ( )
- *RID* **RID** (*Object* from)

9.253 RigidBody

Inherits: *PhysicsBody* < *CollisionObject* < *Spatial* < *Node* < *Object*

Category: Core

9.253.1 Brief Description

Rigid body node.

9.253.2 Member Functions

void	*_integrate_forces* (*PhysicsDirectBodyState* state) virtual
void	*set_mode* (*int* mode)
int	*get_mode* ( ) const
void	*set_mass* (*float* mass)
float	*get_mass* ( ) const
void	*set_weight* (*float* weight)

Continúa en la página siguiente

Tabla 9.23 – proviene de la página anterior

float	*get_weight* ( ) const
void	*set_friction* (*float* friction)
float	*get_friction* ( ) const
void	*set_bounce* (*float* bounce)
float	*get_bounce* ( ) const
void	*set_linear_velocity* (*Vector3* linear_velocity)
Vector3	*get_linear_velocity* ( ) const
void	*set_angular_velocity* (*Vector3* angular_velocity)
Vector3	*get_angular_velocity* ( ) const
void	*set_gravity_scale* (*float* gravity_scale)
float	*get_gravity_scale* ( ) const
void	*set_linear_damp* (*float* linear_damp)
float	*get_linear_damp* ( ) const
void	*set_angular_damp* (*float* angular_damp)
float	*get_angular_damp* ( ) const
void	*set_max_contacts_reported* (*int* amount)
int	*get_max_contacts_reported* ( ) const
void	*set_use_custom_integrator* (*bool* enable)
bool	*is_using_custom_integrator* ( )
void	*set_contact_monitor* (*bool* enabled)
bool	*is_contact_monitor_enabled* ( ) const
void	*set_use_continuous_collision_detection* (*bool* enable)
bool	*is_using_continuous_collision_detection* ( ) const
void	*set_axis_velocity* (*Vector3* axis_velocity)
void	*apply_impulse* (*Vector3* pos, *Vector3* impulse)
void	*set_sleeping* (*bool* sleeping)
bool	*is_sleeping* ( ) const
void	*set_can_sleep* (*bool* able_to_sleep)
bool	*is_able_to_sleep* ( ) const
void	*set_axis_lock* (*int* axis_lock)
int	*get_axis_lock* ( ) const
Array	*get_colliding_bodies* ( ) const

9.253.3 Signals

- **body_enter** (*Object* body)
- **body_enter_shape** (*int* body_id, *Object* body, *int* body_shape, *int* local_shape)
- **body_exit** (*Object* body)
- **body_exit_shape** (*int* body_id, *Object* body, *int* body_shape, *int* local_shape)
- **sleeping_state_changed** ( )

9.253.4 Numeric Constants

- **MODE_STATIC = 1** — Static mode. The body behaves like a *StaticBody*, and can only move by user code.
- **MODE_KINEMATIC = 3** — Kinematic body. The body behaves like a *KinematicBody*, and can only move by user code.

- **MODE_RIGID = 0** — Rigid body. This is the "natural" state of a rigid body. It is affected by forces, and can move, rotate, and be affected by user code.
- **MODE_CHARACTER = 2**

9.253.5 Description

Rigid body node. This node is used for placing rigid bodies in the scene. It can contain a number of shapes, and also shift mode between regular Rigid body, Kinematic, Character or Static.

9.253.6 Member Function Description

- void **_integrate_forces** (*PhysicsDirectBodyState* state) virtual

Called during physics processing, allowing you to read and safely modify the simulation state for the object. By default it works in addition to the usual physics behavior, but *set_use_custom_integrator* allows you to disable the default behavior and do fully custom force integration for a body.

- void **set_mode** (*int* mode)

Set the body mode, from the MODE_* enum. This allows to change to a static body or a character body.

- *int* **get_mode** ( ) const

Return the current body mode, see *set_mode*.

- void **set_mass** (*float* mass)

Set the body mass.

- *float* **get_mass** ( ) const

Return the current body mass.

- void **set_weight** (*float* weight)

Set the body weight given standard earth-weight (gravity 9.8).

- *float* **get_weight** ( ) const

Return the current body weight, given standard earth-weight (gravity 9.8).

- void **set_friction** (*float* friction)

Set the body friction, from 0 (frictionless) to 1 (max friction).

- *float* **get_friction** ( ) const

Return the current body friction, from 0 (frictionless) to 1 (max friction).

- void **set_bounce** (*float* bounce)

Set the body bounciness, from 0 (no bounciness) to 1 (max bounciness).

- *float* **get_bounce** ( ) const

Return the current body bounciness.

- void **set_linear_velocity** (*Vector3* linear_velocity)

Set the body linear velocity. Can be used sporadically, but **DON'T SET THIS IN EVERY FRAME**, because physics may be running in another thread and definitely runs at a different granularity. Use *_integrate_forces* as your process loop if you want to have precise control of the body state.

- *Vector3* **get_linear_velocity** ( ) const

Return the current body linear velocity.

- void **set_angular_velocity** (*Vector3* angular_velocity)

Set the body angular velocity. Can be used sporadically, but **DON'T SET THIS IN EVERY FRAME**, because physics may be running in another thread and definitely runs at a different granularity. Use *_integrate_forces* as your process loop if you want to have precise control of the body state.

- *Vector3* **get_angular_velocity** ( ) const

Return the current body angular velocity.

- void **set_gravity_scale** (*float* gravity_scale)

Set the gravity factor. This factor multiplies gravity intensity just for this body.

- *float* **get_gravity_scale** ( ) const

Return the current body gravity scale.

- void **set_linear_damp** (*float* linear_damp)

Set the linear damp for this body. Default of -1, cannot be less than -1. If this value is different from -1, any linear damp derived from the world or areas will be overridden.

- *float* **get_linear_damp** ( ) const

Return the current body linear damp. Default is -1.

- void **set_angular_damp** (*float* angular_damp)

Set the angular damp for this body. Default of -1, cannot be less than -1. If this value is different from -1, any angular damp derived from the world or areas will be overridden.

- *float* **get_angular_damp** ( ) const

Return the current body angular damp. Default is -1.

- void **set_max_contacts_reported** (*int* amount)

Set the maximum contacts to report. Bodies can keep a log of the contacts with other bodies, this is enabled by setting the maximum amount of contacts reported to a number greater than 0.

- *int* **get_max_contacts_reported** ( ) const

Return the maximum contacts that can be reported. See *set_max_contacts_reported*.

- void **set_use_custom_integrator** (*bool* enable)

Pass true to disable the internal force integration (like gravity or air friction) for this body. Other than collision response, the body will only move as determined by the *_integrate_forces* function, if defined.

- *bool* **is_using_custom_integrator** ( )

Return whether the body is using a custom integrator.

- void **set_contact_monitor** (*bool* enabled)

Enable contact monitoring. This allows the body to emit signals when it collides with another.

- *bool* **is_contact_monitor_enabled** ( ) const

Return whether contact monitoring is enabled.

- void **set_use_continuous_collision_detection** (*bool* enable)

Set the continuous collision detection mode from the enum CCD_MODE_*.

Continuous collision detection tries to predict where a moving body will collide, instead of moving it and correcting its movement if it collided. The first is more precise, and misses less impacts by small, fast-moving objects. The second is faster to compute, but can miss small, fast-moving objects.

- *bool* **is_using_continuous_collision_detection** **(** **)** const

Return whether this body is using continuous collision detection.

- void **set_axis_velocity** **(** *Vector3* axis_velocity **)**

Set an axis velocity. The velocity in the given vector axis will be set as the given vector length. This is useful for jumping behavior.

- void **apply_impulse** **(** *Vector3* pos, *Vector3* impulse **)**

Apply a positioned impulse (which will be affected by the body mass and shape). This is the equivalent of hitting a billiard ball with a cue: a force that is applied once, and only once. Both the impulse and the offset from the body origin are in global coordinates.

- void **set_sleeping** **(** *bool* sleeping **)**

Set whether a body is sleeping or not. Sleeping bodies are not affected by forces until a collision or an *apply_impulse* wakes them up. Until then, they behave like a static body.

- *bool* **is_sleeping** **(** **)** const

Return whether the body is sleeping.

- void **set_can_sleep** **(** *bool* able_to_sleep **)**

Set the body ability to fall asleep when not moving. This saves an enormous amount of processor time when there are plenty of rigid bodies (non static) in a scene.

Sleeping bodies are not affected by forces until a collision or an *apply_impulse* / set_applied_force wakes them up. Until then, they behave like a static body.

- *bool* **is_able_to_sleep** **(** **)** const

Return whether the body has the ability to fall asleep when not moving. See *set_can_sleep*.

- void **set_axis_lock** **(** *int* axis_lock **)**

Set the axis lock of the body, from the AXIS_LOCK_* enum. Axis lock stops the body from moving along the specified axis(X/Y/Z) and rotating along the other two axes.

- *int* **get_axis_lock** **(** **)** const

Return the current axis lock of the body. One of AXIS_LOCK_* enum.

- *Array* **get_colliding_bodies** **(** **)** const

Return a list of the bodies colliding with this one.

9.254 RigidBody2D

Inherits: *PhysicsBody2D* < *CollisionObject2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.254.1 Brief Description

Rigid body 2D node.

9.254.2 Member Functions

void	*_integrate_forces* (*Physics2DDirectBodyState* state) virtual
void	*set_mode* (*int* mode)
int	*get_mode* ( ) const
void	*set_mass* (*float* mass)
float	*get_mass* ( ) const
void	*set_weight* (*float* weight)
float	*get_weight* ( ) const
void	*set_friction* (*float* friction)
float	*get_friction* ( ) const
void	*set_bounce* (*float* bounce)
float	*get_bounce* ( ) const
void	*set_gravity_scale* (*float* gravity_scale)
float	*get_gravity_scale* ( ) const
void	*set_linear_damp* (*float* linear_damp)
float	*get_linear_damp* ( ) const
void	*set_angular_damp* (*float* angular_damp)
float	*get_angular_damp* ( ) const
void	*set_linear_velocity* (*Vector2* linear_velocity)
Vector2	*get_linear_velocity* ( ) const
void	*set_angular_velocity* (*float* angular_velocity)
float	*get_angular_velocity* ( ) const
void	*set_max_contacts_reported* (*int* amount)
int	*get_max_contacts_reported* ( ) const
void	*set_use_custom_integrator* (*bool* enable)
bool	*is_using_custom_integrator* ( )
void	*set_contact_monitor* (*bool* enabled)
bool	*is_contact_monitor_enabled* ( ) const
void	*set_continuous_collision_detection_mode* (*int* mode)
int	*get_continuous_collision_detection_mode* ( ) const
void	*set_axis_velocity* (*Vector2* axis_velocity)
void	*apply_impulse* (*Vector2* pos, *Vector2* impulse)
void	*set_applied_force* (*Vector2* force)
Vector2	*get_applied_force* ( ) const
void	*set_sleeping* (*bool* sleeping)
bool	*is_sleeping* ( ) const
void	*set_can_sleep* (*bool* able_to_sleep)
bool	*is_able_to_sleep* ( ) const
bool	*test_motion* (*Vector2* motion, *float* margin=0.08, *Physics2DTestMotionResult* result=NULL)
Array	*get_colliding_bodies* ( ) const

9.254.3 Signals

- **body_enter** (*Object* body)
- **body_enter_shape** (*int* body_id, *Object* body, *int* body_shape, *int* local_shape)

- **body_exit** (*Object* body)
- **body_exit_shape** (*int* body_id, *Object* body, *int* body_shape, *int* local_shape)
- **sleeping_state_changed** ( )

9.254.4 Numeric Constants

- **MODE_STATIC = 1** — Static mode. The body behaves like a *StaticBody2D*, and can only move by user code.
- **MODE_KINEMATIC = 3** — Kinematic body. The body behaves like a *KinematicBody2D*, and can only move by user code.
- **MODE_RIGID = 0** — Rigid body. This is the "natural" state of a rigid body. It is affected by forces, and can move, rotate, and be affected by user code.
- **MODE_CHARACTER = 2** — Character body. This behaves like a rigid body, but can not rotate.
- **CCD_MODE_DISABLED = 0** — Disables continuous collision detection. This is the fastest way to detect body collisions, but can miss small, fast-moving objects.
- **CCD_MODE_CAST_RAY = 1** — Enables continuous collision detection by raycasting. It is faster than shapecasting, but less precise.
- **CCD_MODE_CAST_SHAPE = 2** — Enables continuous collision detection by shapecasting. It is the slowest CCD method, and the most precise.

9.254.5 Description

Rigid body 2D node. This node is used for placing rigid bodies in the scene. It can contain a number of shapes, and also shift state between regular Rigid body, Kinematic, Character or Static.

Character mode forbids the node from being rotated. This node can have a custom force integrator function, for writing complex physics motion behavior per node.

As a warning, don't change this node position every frame or very often. Sporadic changes work fine, but physics runs at a different granularity (fixed hz) than usual rendering (process callback) and maybe even in a separate thread, so changing this from a process loop will yield strange behavior.

9.254.6 Member Function Description

- void **_integrate_forces** (*Physics2DDirectBodyState* state) virtual

Called during physics processing, allowing you to read and safely modify the simulation state for the object. By default it works in addition to the usual physics behavior, but *set_use_custom_integrator* allows you to disable the default behavior and do fully custom force integration for a body.

- void **set_mode** (*int* mode)

Set the body mode, from the MODE_* enum. This allows to change to a static body or a character body.

- *int* **get_mode** ( ) const

Return the current body mode, see *set_mode*.

- void **set_mass** (*float* mass)

Set the body mass.

- *float* **get_mass** ( ) const

Return the body mass.

- void **set_weight** (*float* weight)

Set the body weight given standard earth-weight (gravity 9.8). Not really useful for 2D since most measures for this node are in pixels.

- *float* **get_weight** ( ) const

Return the body weight given standard earth-weight (gravity 9.8).

- void **set_friction** (*float* friction)

Set the body friction, from 0 (frictionless) to 1 (full friction).

- *float* **get_friction** ( ) const

Return the body friction.

- void **set_bounce** (*float* bounce)

Set the body bounciness, from 0 (no bounce) to 1 (full bounce).

- *float* **get_bounce** ( ) const

Return the body bounciness.

- void **set_gravity_scale** (*float* gravity_scale)

Set the gravity factor. This factor multiplies gravity intensity just for this body.

- *float* **get_gravity_scale** ( ) const

Return the gravity factor.

- void **set_linear_damp** (*float* linear_damp)

Set the linear damp for this body. If this value is different from -1, any linear damp derived from the world or areas will be overridden.

- *float* **get_linear_damp** ( ) const

Return the linear damp for this body.

- void **set_angular_damp** (*float* angular_damp)

Set the angular damp for this body. If this value is different from -1, any angular damp derived from the world or areas will be overridden.

- *float* **get_angular_damp** ( ) const

Return the angular damp for this body.

- void **set_linear_velocity** (*Vector2* linear_velocity)

Set the body linear velocity. Can be used sporadically, but **DON'T SET THIS IN EVERY FRAME**, because physics may be running in another thread and definitely runs at a different granularity. Use *_integrate_forces* as your process loop if you want to have precise control of the body state.

- *Vector2* **get_linear_velocity** ( ) const

Return the body linear velocity. This changes by physics granularity. See *set_linear_velocity*.

- void **set_angular_velocity** (*float* angular_velocity)

Set the body angular velocity. Can be used sporadically, but **DON'T SET THIS IN EVERY FRAME**, because physics may be running in another thread and definitely runs at a different granularity. Use *_integrate_forces* as your process loop if you want to have precise control of the body state.

- *float* **get_angular_velocity** **(** **)** const

Return the body angular velocity. This changes by physics granularity. See *set_angular_velocity*.

- void **set_max_contacts_reported** **(** *int* amount **)**

Set the maximum contacts to report. Bodies can keep a log of the contacts with other bodies, this is enabled by setting the maximum amount of contacts reported to a number greater than 0.

- *int* **get_max_contacts_reported** **(** **)** const

Return the maximum contacts that can be reported. See *set_max_contacts_reported*.

- void **set_use_custom_integrator** **(** *bool* enable **)**

Pass true to disable the internal force integration (like gravity or air friction) for this body. Other than collision response, the body will only move as determined by the *_integrate_forces* function, if defined.

- *bool* **is_using_custom_integrator** **(** **)**

Return true if the body is not doing any built-in force integration.

- void **set_contact_monitor** **(** *bool* enabled **)**

Enable contact monitoring. This allows the body to emit signals when it collides with another.

- *bool* **is_contact_monitor_enabled** **(** **)** const

Return whether contact monitoring is enabled.

- void **set_continuous_collision_detection_mode** **(** *int* mode **)**

Set the continuous collision detection mode from the enum CCD_MODE_*.

Continuous collision detection tries to predict where a moving body will collide, instead of moving it and correcting its movement if it collided. The first is more precise, and misses less impacts by small, fast-moving objects. The second is faster to compute, but can miss small, fast-moving objects.

- *int* **get_continuous_collision_detection_mode** **(** **)** const

Return whether this body is using continuous collision detection.

- void **set_axis_velocity** **(** *Vector2* axis_velocity **)**

Set an axis velocity. The velocity in the given vector axis will be set as the given vector length. This is useful for jumping behavior.

- void **apply_impulse** **(** *Vector2* pos, *Vector2* impulse **)**

Apply a positioned impulse (which will be affected by the body mass and shape). This is the equivalent of hitting a billiard ball with a cue: a force that is applied once, and only once.

- void **set_applied_force** **(** *Vector2* force **)**

Set the applied force vector. This is the equivalent of pushing a box over the ground: the force applied is applied constantly.

- *Vector2* **get_applied_force** **(** **)** const

Return the applied force vector.

- void **set_sleeping** **(** *bool* sleeping **)**

Set whether a body is sleeping or not. Sleeping bodies are not affected by forces until a collision or an *apply_impulse* / *set_applied_force* wakes them up. Until then, they behave like a static body.

- *bool* **is_sleeping** **(** **)** const

Return whether the body is sleeping.

- void **set_can_sleep** (*bool* able_to_sleep)

Set the body ability to fall asleep when not moving. This saves an enormous amount of processor time when there are plenty of rigid bodies (non static) in a scene.

Sleeping bodies are not affected by forces until a collision or an *apply_impulse* / *set_applied_force* wakes them up. Until then, they behave like a static body.

- *bool* **is_able_to_sleep** ( ) const

Return true if the body has the ability to fall asleep when not moving. See *set_can_sleep*.

- *bool* **test_motion** (*Vector2* motion, *float* margin=0.08, *Physics2DTestMotionResult* result=NULL)

Return whether the body would collide, if it tried to move in the given vector. This method allows two extra parameters: A margin, which increases slightly the size of the shapes involved in the collision detection, and an object of type *Physics2DTestMotionResult*, which will store additional information about the collision (should there be one).

- *Array* **get_colliding_bodies** ( ) const

Return a list of the bodies colliding with this one.

9.255 Room

Inherits: *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.255.1 Brief Description

Room data resource.

9.255.2 Member Functions

void	*set_room* (*Room* room)
Room	*get_room* ( ) const
void	*compute_room_from_subtree* ( )
void	*set_simulate_acoustics* (*bool* enable)
bool	*is_simulating_acoustics* ( ) const

9.255.3 Description

Room contains the data to define the bounds of a scene (using a BSP Tree). It is instanced by a *VisualInstance* node to create rooms. See that class documentation for more information about rooms.

9.255.4 Member Function Description

- void **set_room** (*Room* room)
- *Room* **get_room** ( ) const
- void **compute_room_from_subtree** ( )
- void **set_simulate_acoustics** (*bool* enable)

- *bool* **is_simulating_acoustics** ( ) const

9.256 RoomBounds

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.256.1 Brief Description

9.256.2 Member Functions

void	*set_bounds* (*Dictionary* bsp_tree)
Dictionary	*get_bounds* ( ) const
void	*set_geometry_hint* (*Vector3Array* triangles)
Vector3Array	*get_geometry_hint* ( ) const
void	*regenerate_bsp* ( )
void	*regenerate_bsp_cubic* ( )

9.256.3 Member Function Description

- void **set_bounds** (*Dictionary* bsp_tree)
- *Dictionary* **get_bounds** ( ) const
- void **set_geometry_hint** (*Vector3Array* triangles)
- *Vector3Array* **get_geometry_hint** ( ) const
- void **regenerate_bsp** ( )
- void **regenerate_bsp_cubic** ( )

9.257 Sample

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.257.1 Brief Description

Audio sample (sound) class.

9.257.2 Member Functions

void	*create* (*int* format, *bool* stereo, *int* length)
int	*get_format* ( ) const
bool	*is_stereo* ( ) const
int	*get_length* ( ) const
void	*set_data* (*RawArray* data)
RawArray	*get_data* ( ) const
void	*set_mix_rate* (*int* hz)
int	*get_mix_rate* ( ) const
void	*set_loop_format* (*int* format)
int	*get_loop_format* ( ) const
void	*set_loop_begin* (*int* pos)
int	*get_loop_begin* ( ) const
void	*set_loop_end* (*int* pos)
int	*get_loop_end* ( ) const

9.257.3 Numeric Constants

- **FORMAT_PCM8 = 0** — 8-bits signed PCM audio.
- **FORMAT_PCM16 = 1** — 16-bits signed little endian PCM audio.
- **FORMAT_IMA_ADPCM = 2** — IMA-ADPCM Audio.
- **LOOP_NONE = 0** — No loop enabled.
- **LOOP_FORWARD = 1** — Forward looping (when playback reaches loop end, goes back to loop begin).
- **LOOP_PING_PONG = 2** — Ping-pong looping (when playback reaches loop end, plays backward until loop begin). Not available in all platforms.

9.257.4 Description

Sample provides an audio sample class, containing audio data, together with some information for playback, such as format, mix rate and loop. It is used by sound playback routines.

9.257.5 Member Function Description

- void **create** (*int* format, *bool* stereo, *int* length)

Create new data for the sample, with format (see FORMAT_* constants), stereo hint, and length in samples (not bytes).

Calling this method overrides previously existing data. Stereo samples are interleaved pairs of left and right points (in that order), but count as one sample for length purposes.

- *int* **get_format** ( ) const

Return the sample format.

- *bool* **is_stereo** ( ) const

Return whether the current sample was created as stereo.

- *int* **get_length** ( ) const

Return the sample length in samples. Stereo samples count as one, even if they are made of a left and a right sample.

- void **set_data** (*RawArray* data)

Set sample data. Data must be little endian, no matter the host platform, and exactly as long as to fit all samples. The length of this array can be calculated as follows:

Get the sample length (*get_length*).

If the sample format is FORMAT_PCM16, multiply it by 2.

If the sample format is FORMAT_IMA_ADPCM, divide it by 2 (rounding any fraction up), then add 4.

If the sample is stereo (*is_stereo*), multiply it by 2.

- *RawArray* **get_data** ( ) const

Return sample data as little endian.

- void **set_mix_rate** (*int* hz)

Set the mix rate for the sample (expected playback frequency).

- *int* **get_mix_rate** ( ) const

Return the mix rate for the sample.

- void **set_loop_format** (*int* format)

Set the loop format (use LOOP_* constants as argument).

- *int* **get_loop_format** ( ) const

Return the loop format.

- void **set_loop_begin** (*int* pos)

Set the loop begin position. It must be a valid frame and less than the loop end position.

- *int* **get_loop_begin** ( ) const

Return the loop begin position.

- void **set_loop_end** (*int* pos)

Set the loop end position. It must be a valid frame and greater than the loop begin position.

- *int* **get_loop_end** ( ) const

Return the loop end position.

9.258 SampleLibrary

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.258.1 Brief Description

Library that contains a collection of samples.

9.258.2 Member Functions

void	*add_sample* (*String* name, *Sample* sample)
Sample	*get_sample* (*String* name) const
bool	*has_sample* (*String* name) const
void	*remove_sample* (*String* name)
void	*sample_set_volume_db* (*String* name, *float* db)
float	*sample_get_volume_db* (*String* name) const
void	*sample_set_pitch_scale* (*String* name, *float* pitch)
float	*sample_get_pitch_scale* (*String* name) const

9.258.3 Description

Library that contains a collection of *Sample*, each identified by a text ID. This is used as a data container for the majority of the SamplePlayer classes and derivatives.

9.258.4 Member Function Description

- void **add_sample** (*String* name, *Sample* sample)

Add a sample to the library, with a given text ID.

- *Sample* **get_sample** (*String* name) const

Return the sample from the library matching the given text ID. Return null if the sample is not found.

- *bool* **has_sample** (*String* name) const

Return true if the sample text ID exists in the library.

- void **remove_sample** (*String* name)

Remove the sample matching the given text ID.

- void **sample_set_volume_db** (*String* name, *float* db)

Set the volume (in dB) for the given sample.

- *float* **sample_get_volume_db** (*String* name) const

Return the volume (in dB) for the given sample.

- void **sample_set_pitch_scale** (*String* name, *float* pitch)

Set the pitch scale for the given sample.

- *float* **sample_get_pitch_scale** (*String* name) const

Return the pitch scale for the given sample.

9.259 SamplePlayer

Inherits: *Node* < *Object*

Category: Core

9.259.1 Brief Description

Sample Player node.

9.259.2 Member Functions

void	*set_sample_library* (*SampleLibrary* library)
SampleLibrary	*get_sample_library* ( ) const
void	*set_polyphony* (*int* max_voices)
int	*get_polyphony* ( ) const
int	*play* (*String* name, *bool* unique=false)
void	*stop* (*int* voice)
void	*stop_all* ( )
void	*set_mix_rate* (*int* voice, *int* hz)
void	*set_pitch_scale* (*int* voice, *float* ratio)
void	*set_volume* (*int* voice, *float* volume)
void	*set_volume_db* (*int* voice, *float* db)
void	*set_pan* (*int* voice, *float* pan, *float* depth=0, *float* height=0)
void	*set_filter* (*int* voice, *int* type, *float* cutoff_hz, *float* resonance, *float* gain=0)
void	*set_chorus* (*int* voice, *float* send)
void	*set_reverb* (*int* voice, *int* room_type, *float* send)
int	*get_mix_rate* (*int* voice) const
float	*get_pitch_scale* (*int* voice) const
float	*get_volume* (*int* voice) const
float	*get_volume_db* (*int* voice) const
float	*get_pan* (*int* voice) const
float	*get_pan_depth* (*int* voice) const
float	*get_pan_height* (*int* voice) const
int	*get_filter_type* (*int* voice) const
float	*get_filter_cutoff* (*int* voice) const
float	*get_filter_resonance* (*int* voice) const
float	*get_filter_gain* (*int* voice) const
float	*get_chorus* (*int* voice) const
int	*get_reverb_room* (*int* voice) const
float	*get_reverb* (*int* voice) const
void	*set_default_pitch_scale* (*float* ratio)
void	*set_default_volume* (*float* volume)
void	*set_default_volume_db* (*float* db)
void	*set_default_pan* (*float* pan, *float* depth=0, *float* height=0)
void	*set_default_filter* (*int* type, *float* cutoff_hz, *float* resonance, *float* gain=0)
void	*set_default_chorus* (*float* send)
void	*set_default_reverb* (*int* room_type, *float* send)
float	*get_default_pitch_scale* ( ) const
float	*get_default_volume* ( ) const
float	*get_default_volume_db* ( ) const
float	*get_default_pan* ( ) const
float	*get_default_pan_depth* ( ) const
float	*get_default_pan_height* ( ) const
int	*get_default_filter_type* ( ) const
float	*get_default_filter_cutoff* ( ) const

Continúa en la página siguiente

Tabla 9.25 – proviene de la página anterior

float	*get_default_filter_resonance* ( ) const
float	*get_default_filter_gain* ( ) const
float	*get_default_chorus* ( ) const
int	*get_default_reverb_room* ( ) const
float	*get_default_reverb* ( ) const
bool	*is_active* ( ) const
bool	*is_voice_active* (*int* voice) const

9.259.3 Numeric Constants

- **FILTER_NONE = 0** — Filter is disabled for voice.
- **FILTER_LOWPASS = 1** — Low-pass filter is used for voice.
- **FILTER_BANDPASS = 2** — Band-pass filter is used for voice.
- **FILTER_HIPASS = 3** — High-pass filter is used for voice.
- **FILTER_NOTCH = 4** — Notch (band reject) filter is used for voice.
- **FILTER_PEAK = 5** — Peak (exclusive band) filter is used for voice.
- **FILTER_BANDLIMIT = 6** — Band-limit filter is used for voice, in this case resonance is the high-pass cutoff. A band-limit filter has a different frequency response than a notch filter, but otherwise both are band-rejecting filters.
- **FILTER_LOW_SHELF = 7** — Low-shelf filter is used for voice.
- **FILTER_HIGH_SHELF = 8** — High-shelf filter is used for voice.
- **REVERB_SMALL = 0** — Small reverberation room (house room).
- **REVERB_MEDIUM = 1** — Medium reverberation room (street)
- **REVERB_LARGE = 2** — Large reverberation room (theatre)
- **REVERB_HALL = 3** — Huge reverberation room (cathedral, warehouse).
- **INVALID_VOICE_ID = -1** — Value returned if the voice ID is invalid.

9.259.4 Description

SamplePlayer is a *Node* meant for simple sample playback. A library of samples is loaded and played back "as is", without positioning or anything.

9.259.5 Member Function Description

- void **set_sample_library** (*SampleLibrary* library)

Set the sample library for the player.

- *SampleLibrary* **get_sample_library** ( ) const

Return the sample library used by the player.

- void **set_polyphony** (*int* max_voices)

Set the polyphony of the player (maximum amount of simultaneous voices).

- *int* **get_polyphony** ( ) const

Return the polyphony of the player.

- *int* **play** (*String* name, *bool* unique=false)

Play a sample referenced by its name.

Optionally, the playback can be made "unique" to force stopping all other samples currently played. The voices allocated for playback will then be returned.

- void **stop** (*int* voice)

Stop a given voice.

- void **stop_all** ( )

Stop all playing voices.

- void **set_mix_rate** (*int* voice, *int* hz)

Set the mix rate (in Hz) of a given voice.

- void **set_pitch_scale** (*int* voice, *float* ratio)

Set the pitch scale of a given voice. A ratio of 1.0 is the normal scale.

- void **set_volume** (*int* voice, *float* volume)

Set the volume of a given voice using a linear scale.

The "volume" argument should be a positive factor ranging from 0.0 (mute) up to 16.0 (i.e. 24 dB).

A factor of 1.0 means that the voice will be played at normal system volume. Factors above 1.0 might be limited by the platform's audio output.

- void **set_volume_db** (*int* voice, *float* db)

Set the volume of a given voice in dB.

The "dB" argument can range from -80 to 24 dB, 0 dB being the maximum volume. Every 6 dB (resp. -6 dB), the volume is increased (resp. reduced) by half.

- void **set_pan** (*int* voice, *float* pan, *float* depth=0, *float* height=0)

Set the panning of a voice. Panning goes from -1.0 (left) to +1.0 (right).

Optionally, for hardware than support 3D sound, one can also set depth and height (also in range -1.0 to +1.0).

- void **set_filter** (*int* voice, *int* type, *float* cutoff_hz, *float* resonance, *float* gain=0)

Set the filter for a given voice, using the given type (see FILTER_* constants), cutoff frequency (from 20 to 16,384 Hz) and resonance (from 0 to 4.0).

Optionally, a gain can also be given (from 0 to 2.0).

- void **set_chorus** (*int* voice, *float* send)

Set the chorus send level of a voice (from 0 to 1.0). For setting chorus parameters, see *AudioServer*.

- void **set_reverb** (*int* voice, *int* room_type, *float* send)

Set the reverberation type (see REVERB_* constants) and send level (from 0 to 1.0) of a voice.

- *int* **get_mix_rate** (*int* voice) const

Return the current mix rate for a given voice.

- *float* **get_pitch_scale** (*int* voice) const

Return the current pitch scale for a given voice.

- *float* **get_volume** (*int* voice) const

Return the current volume (on a linear scale) for a given voice.

- *float* **get_volume_db** (*int* voice) const

Return the current volume (in dB) for a given voice.

- *float* **get_pan** (*int* voice) const

Return the current panning for a given voice.

- *float* **get_pan_depth** (*int* voice) const

Return the current pan depth for a given voice.

- *float* **get_pan_height** (*int* voice) const

Return the current pan height for a given voice.

- *int* **get_filter_type** (*int* voice) const

Return the current filter type in use (see FILTER_* constants) for a given voice.

- *float* **get_filter_cutoff** (*int* voice) const

Return the current filter cutoff frequency for a given voice.

- *float* **get_filter_resonance** (*int* voice) const

Return the current filter resonance for a given voice.

- *float* **get_filter_gain** (*int* voice) const

Return the current filter gain for a given voice.

- *float* **get_chorus** (*int* voice) const

Return the current chorus send level for a given voice.

- *int* **get_reverb_room** (*int* voice) const

Return the current reverberation room type for a given voice (see REVERB_* enum).

- *float* **get_reverb** (*int* voice) const

Return the current reverberation send level for a given voice.

- void **set_default_pitch_scale** (*float* ratio)

Set the default pitch scale of the player. A ratio of 1.0 is the normal scale.

- void **set_default_volume** (*float* volume)

Set the default volume of the player using a linear scale.

The "volume" argument should be a positive factor ranging from 0.0 (mute) up to 16.0 (i.e. 24 dB).

A factor of 1.0 means that the voice will be played at normal system volume. Factors above 1.0 might be limited by the platform's audio output.

- void **set_default_volume_db** (*float* db)

Set the default volume of the player in dB.

The "dB" argument can range from -80 to 24 dB, 0 dB being the maximum volume. Every 6 dB (resp. -6 dB), the volume is increased (resp. reduced) by half.

- void **set_default_pan** (*float* pan, *float* depth=0, *float* height=0)

Set the default panning of the player. Panning goes from -1.0 (left) to +1.0 (right).

Optionally, for hardware than support 3D sound, one can also set depth and height (also in range -1.0 to +1.0).

- void **set_default_filter** (*int* type, *float* cutoff_hz, *float* resonance, *float* gain=0)

Set the default filter for the player, using the given type (see FILTER_* constants), cutoff frequency (from 20 to 16,384 Hz) and resonance (from 0 to 4.0).

Optionally, a gain can also be given (from 0 to 2.0).

- void **set_default_chorus** (*float* send)

Set the default chorus send level of the player (from 0 to 1.0). For setting chorus parameters, see *AudioServer*.

- void **set_default_reverb** (*int* room_type, *float* send)

Set the default reverberation type (see REVERB_* constants) and send level (from 0 to 1.0) of the player.

- *float* **get_default_pitch_scale** ( ) const

Return the default pitch scale of the player.

- *float* **get_default_volume** ( ) const

Return the default volume (on a linear scale) of the player.

- *float* **get_default_volume_db** ( ) const

Return the default volume (in dB) of the player.

- *float* **get_default_pan** ( ) const

Return the default panning of the player.

- *float* **get_default_pan_depth** ( ) const

Return the default pan depth of the player.

- *float* **get_default_pan_height** ( ) const

Return the default pan height of the player.

- *int* **get_default_filter_type** ( ) const

Return the default filter type in use (see FILTER_* constants) for the player.

- *float* **get_default_filter_cutoff** ( ) const

Return the default filter cutoff frequency of the player.

- *float* **get_default_filter_resonance** ( ) const

Return the default filter resonance of the player.

- *float* **get_default_filter_gain** ( ) const

Return the default filter gain of the player.

- *float* **get_default_chorus** ( ) const

Return the default chorus send level of the player.

- *int* **get_default_reverb_room** ( ) const

Return the default reverberation room type of the player (see REVERB_* enum).

- *float* **get_default_reverb** ( ) const

Return the default reverberation send level of the player.

- *bool* **is_active** ( ) const

Return whether the player is currently active.

- *bool* **is_voice_active** (*int* voice) const

Return whether the given voice is currently active.

9.260 SamplePlayer2D

Inherits: *SoundPlayer2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.260.1 Brief Description

Sample player for positional 2D Sound.

9.260.2 Member Functions

void	*set_sample_library* (*SampleLibrary* library)
SampleLibrary	*get_sample_library* ( ) const
void	*set_polyphony* (*int* max_voices)
int	*get_polyphony* ( ) const
int	*play* (*String* sample, *int* voice=-2)
void	*voice_set_pitch_scale* (*int* voice, *float* ratio)
void	*voice_set_volume_scale_db* (*int* voice, *float* db)
bool	*is_voice_active* (*int* voice) const
void	*stop_voice* (*int* voice)
void	*stop_all* ( )
void	*set_random_pitch_scale* (*float* val)
float	*get_random_pitch_scale* ( ) const

9.260.3 Numeric Constants

- **INVALID_VOICE = -1** — Value returned if the voice or sample are invalid.
- **NEXT_VOICE = -2** — Default voice for the play method. Corresponds to the first voice following the last used voice.

9.260.4 Description

Sample player for positional 2D Sound. Plays sound samples positionally, left and right depending on the distance/place on the screen.

9.260.5 Member Function Description

- void **set_sample_library** (*SampleLibrary* library)

Set the sample library for the player.

- *SampleLibrary* **get_sample_library ( )** const

Return the sample library used by the player.

- void **set_polyphony (*int* max_voices)**

Set the polyphony of the player (maximum amount of simultaneous voices).

- *int* **get_polyphony ( )** const

Return the polyphony of the player.

- *int* **play (*String* sample, *int* voice=-2)**

Play a sample. An internal polyphony ID can optionally be passed, or defaults to NEXT_VOICE.

Return a voice ID which can be used to modify the voice parameters, or INVALID_VOICE if the voice or sample are invalid.

- void **voice_set_pitch_scale (*int* voice, *float* ratio)**

Change the pitch scale of a currently playing voice.

- void **voice_set_volume_scale_db (*int* voice, *float* db)**

Change the volume scale (in dB) of a currently playing voice.

- *bool* **is_voice_active (*int* voice)** const

Return whether a voice is still active or has stopped playing.

- void **stop_voice (*int* voice)**

Stop a given voice.

- void **stop_all ( )**

Stop all playing voices.

- void **set_random_pitch_scale (*float* val)**

Set the amplitude for random pitch scale variations. If different from zero, the pitch scale will vary randomly around 1.0 in a range defined by val.

The actual pitch scale will be, with "variation" ranging from -val to val:

* variation > 0: 1.0 + variation

* variation < 0: 1.0/(1.0 - variation)

- *float* **get_random_pitch_scale ( )** const

Return the amplitude used for random pitch scale variations.

9.261 SceneState

Inherits: *Reference* **<** *Object*

Category: Core

9.261.1 Brief Description

9.261.2 Member Functions

int	*get_node_count* ( ) const
String	*get_node_type* (*int* idx) const
String	*get_node_name* (*int* idx) const
NodePath	*get_node_path* (*int* idx, *bool* for_parent=false) const
NodePath	*get_node_owner_path* (*int* idx) const
bool	*is_node_instance_placeholder* (*int* idx) const
String	*get_node_instance_placeholder* (*int* idx) const
PackedScene	*get_node_instance* (*int* idx) const
StringArray	*get_node_groups* (*int* idx) const
int	*get_node_property_count* (*int* idx) const
String	*get_node_property_name* (*int* idx, *int* prop_idx) const
void	*get_node_property_value* (*int* idx, *int* prop_idx) const
int	*get_connection_count* ( ) const
NodePath	*get_connection_source* (*int* idx) const
String	*get_connection_signal* (*int* idx) const
NodePath	*get_connection_target* (*int* idx) const
String	*get_connection_method* (*int* idx) const
int	*get_connection_flags* (*int* idx) const
Array	*get_connection_binds* (*int* idx) const

9.261.3 Member Function Description

- *int* **get_node_count** ( ) const
- *String* **get_node_type** (*int* idx) const
- *String* **get_node_name** (*int* idx) const
- *NodePath* **get_node_path** (*int* idx, *bool* for_parent=false) const
- *NodePath* **get_node_owner_path** (*int* idx) const
- *bool* **is_node_instance_placeholder** (*int* idx) const
- *String* **get_node_instance_placeholder** (*int* idx) const
- *PackedScene* **get_node_instance** (*int* idx) const
- *StringArray* **get_node_groups** (*int* idx) const
- *int* **get_node_property_count** (*int* idx) const
- *String* **get_node_property_name** (*int* idx, *int* prop_idx) const
- void **get_node_property_value** (*int* idx, *int* prop_idx) const
- *int* **get_connection_count** ( ) const
- *NodePath* **get_connection_source** (*int* idx) const
- *String* **get_connection_signal** (*int* idx) const
- *NodePath* **get_connection_target** (*int* idx) const
- *String* **get_connection_method** (*int* idx) const

- *int* **get_connection_flags** (*int* idx) const
- *Array* **get_connection_binds** (*int* idx) const

9.262 SceneTree

Inherits: *MainLoop* < *Object*

Category: Core

9.262.1 Brief Description

9.262.2 Member Functions

void	*notify_group* (*int* call_flags, *String* group, *int* notification)
void	*set_group* (*int* call_flags, *String* group, *String* property, var value)
Array	*get_nodes_in_group* (*String* group)
Viewport	*get_root* ( ) const
bool	*has_group* (*String* name) const
void	*set_auto_accept_quit* (*bool* enabled)
void	*set_editor_hint* (*bool* enable)
bool	*is_editor_hint* ( ) const
void	*set_debug_collisions_hint* (*bool* enable)
bool	*is_debugging_collisions_hint* ( ) const
void	*set_debug_navigation_hint* (*bool* enable)
bool	*is_debugging_navigation_hint* ( ) const
void	*set_edited_scene_root* (*Object* scene)
Object	*get_edited_scene_root* ( ) const
void	*set_pause* (*bool* enable)
bool	*is_paused* ( ) const
void	*set_input_as_handled* ( )
int	*get_node_count* ( ) const
int	*get_frame* ( ) const
void	*quit* ( )
void	*set_screen_stretch* (*int* mode, *int* aspect, *Vector2* minsize)
void	*queue_delete* (*Object* obj)
void	*call_group* (*int* flags, *String* group, *String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL)
void	*set_current_scene* (*Node* child_node)
Node	*get_current_scene* ( ) const
int	*change_scene* (*String* path)
int	*change_scene_to* (*PackedScene* packed_scene)
int	*reload_current_scene* ( )

9.262.3 Signals

- **screen_resized** ( )
- **node_removed** (*Object* node)

- **idle_frame** ( )
- **tree_changed** ( )
- **fixed_frame** ( )

9.262.4 Numeric Constants

- **GROUP_CALL_DEFAULT = 0**
- **GROUP_CALL_REVERSE = 1**
- **GROUP_CALL_REALTIME = 2**
- **GROUP_CALL_UNIQUE = 4**
- **STRETCH_MODE_DISABLED = 0**
- **STRETCH_MODE_2D = 1**
- **STRETCH_MODE_VIEWPORT = 2**
- **STRETCH_ASPECT_IGNORE = 0**
- **STRETCH_ASPECT_KEEP = 1**
- **STRETCH_ASPECT_KEEP_WIDTH = 2**
- **STRETCH_ASPECT_KEEP_HEIGHT = 3**

9.262.5 Member Function Description

- void **notify_group** (*int* call_flags, *String* group, *int* notification)
- void **set_group** (*int* call_flags, *String* group, *String* property, var value)
- *Array* **get_nodes_in_group** (*String* group)
- *Viewport* **get_root** ( ) const
- *bool* **has_group** (*String* name) const
- void **set_auto_accept_quit** (*bool* enabled)
- void **set_editor_hint** (*bool* enable)
- *bool* **is_editor_hint** ( ) const
- void **set_debug_collisions_hint** (*bool* enable)
- *bool* **is_debugging_collisions_hint** ( ) const
- void **set_debug_navigation_hint** (*bool* enable)
- *bool* **is_debugging_navigation_hint** ( ) const
- void **set_edited_scene_root** (*Object* scene)
- *Object* **get_edited_scene_root** ( ) const
- void **set_pause** (*bool* enable)
- *bool* **is_paused** ( ) const
- void **set_input_as_handled** ( )
- *int* **get_node_count** ( ) const

- *int* **get_frame** ( ) const
- void **quit** ( )
- void **set_screen_stretch** (*int* mode, *int* aspect, *Vector2* minsize)
- void **queue_delete** (*Object* obj)
- void **call_group** (*int* flags, *String* group, *String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL)
- void **set_current_scene** (*Node* child_node)
- *Node* **get_current_scene** ( ) const
- *int* **change_scene** (*String* path)
- *int* **change_scene_to** (*PackedScene* packed_scene)
- *int* **reload_current_scene** ( )

9.263 Script

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *GDScript*

Category: Core

9.263.1 Brief Description

Base class for scripts.

9.263.2 Member Functions

bool	*can_instance* ( ) const
bool	*instance_has* (*Object* base_object) const
bool	*has_source_code* ( ) const
String	*get_source_code* ( ) const
void	*set_source_code* (*String* source)
int	*reload* ( )

9.263.3 Description

Base class for scripts. Any script that is loaded becomes one of these resources, which can then create instances.

9.263.4 Member Function Description

- *bool* **can_instance** ( ) const

Return true if this script can be instance (ie not a library).

- *bool* **instance_has** (*Object* base_object) const

Return true if a given object uses an instance of this script.

- *bool* **has_source_code** ( ) const

Return true if the script contains source code.

- *String* **get_source_code** ( ) const

Return the script source code (if available).

- void **set_source_code** (*String* source)

Set the script source code.

- *int* **reload** ( )

Reload the script. This will fail if there are existing instances.

9.264 ScrollBar

Inherits: *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *HScrollBar*, *VScrollBar*

Category: Core

9.264.1 Brief Description

Base class for scroll bars.

9.264.2 Member Functions

void	*set_custom_step* (*float* step)
float	*get_custom_step* ( ) const

9.264.3 Description

Scrollbars are a *Range* based *Control*, that display a draggable area (the size of the page). Horizontal (*HScrollBar*) and Vertical (*VScrollBar*) versions are available.

9.264.4 Member Function Description

- void **set_custom_step** (*float* step)
- *float* **get_custom_step** ( ) const

9.265 ScrollContainer

Inherits: *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.265.1 Brief Description

A helper node for displaying scrollable elements (e.g. lists).

9.265.2 Member Functions

void	*set_enable_h_scroll* (*bool* enable)
bool	*is_h_scroll_enabled* ( ) const
void	*set_enable_v_scroll* (*bool* enable)
bool	*is_v_scroll_enabled* ( ) const
void	*set_h_scroll* (*int* val)
int	*get_h_scroll* ( ) const
void	*set_v_scroll* (*int* val)
int	*get_v_scroll* ( ) const

9.265.3 Description

A ScrollContainer node with a *Control* child and scrollbar child (*HScrollbar*, *VScrollBar*, or both) will only draw the Control within the ScrollContainer area. Scrollbars will automatically be drawn at the right (for vertical) or bottom (for horizontal) and will enable dragging to move the viewable Control (and its children) within the ScrollContainer. Scrollbars will also automatically resize the grabber based on the minimum_size of the Control relative to the ScrollContainer. Works great with a *Panel* control.

9.265.4 Member Function Description

- void **set_enable_h_scroll** (*bool* enable)

Set allows horizontal scrool.

- *bool* **is_h_scroll_enabled** ( ) const

Return true if horizontal scrool is allowed.

- void **set_enable_v_scroll** (*bool* enable)

Set allows vertical scrool.

- *bool* **is_v_scroll_enabled** ( ) const

Return true if vertical scrool is allowed.

- void **set_h_scroll** (*int* val)

Set horizontal scroll value.

- *int* **get_h_scroll** ( ) const

Return current horizontal scroll value.

- void **set_v_scroll** (*int* val)

Set vertical scroll value.

- *int* **get_v_scroll** ( ) const

Return current vertical scroll value.

9.266 SegmentShape2D

Inherits: *Shape2D* < *Resource* < *Reference* < *Object*

Category: Core

9.266.1 Brief Description

Segment Shape for 2D Collision Detection.

9.266.2 Member Functions

void	*set_a* (*Vector2* a)
Vector2	*get_a* ( ) const
void	*set_b* (*Vector2* b)
Vector2	*get_b* ( ) const

9.266.3 Description

Segment Shape for 2D Collision Detection, consists of two points, 'a' and 'b'.

9.266.4 Member Function Description

- void **set_a** (*Vector2* a)

Set the first point's position.

- *Vector2* **get_a** ( ) const

Return the first point's position.

- void **set_b** (*Vector2* b)

Set the second point's position.

- *Vector2* **get_b** ( ) const

Return the second point's position.

9.267 Semaphore

Inherits: *Reference* < *Object*

Category: Core

9.267.1 Brief Description

9.267.2 Member Functions

Error	*wait* ( )
Error	*post* ( )

9.267.3 Member Function Description

- Error **wait** ( )
- Error **post** ( )

9.268 Separator

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *VSeparator*, *HSeparator*

Category: Core

9.268.1 Brief Description

Base class for separators.

9.268.2 Description

Separator is a *Control* used for separating other controls. It's purely a visual decoration. Horizontal (*HSeparator*) and Vertical (*VSeparator*) versions are available.

9.269 Shader

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *MaterialShader*, *CanvasItemShader*, *ShaderGraph*

Category: Core

9.269.1 Brief Description

To be changed, ignore.

9.269.2 Member Functions

int	*get_mode* ( ) const
void	*set_code* (*String* vcode, *String* fcode, *String* lcode, *int* fofs=0, *int* lofs=0)
String	*get_vertex_code* ( ) const
String	*get_fragment_code* ( ) const
String	*get_light_code* ( ) const
void	*set_default_texture_param* (*String* param, *Texture* texture)
Texture	*get_default_texture_param* (*String* param) const
bool	*has_param* (*String* name) const

9.269.3 Numeric Constants

- **MODE_MATERIAL = 0**
- **MODE_CANVAS_ITEM = 1**
- **MODE_POST_PROCESS = 2**

9.269.4 Description

To be changed, ignore.

9.269.5 Member Function Description

- *int* **get_mode** ( ) const
- void **set_code** (*String* vcode, *String* fcode, *String* lcode, *int* fofs=0, *int* lofs=0)
- *String* **get_vertex_code** ( ) const
- *String* **get_fragment_code** ( ) const
- *String* **get_light_code** ( ) const
- void **set_default_texture_param** (*String* param, *Texture* texture)
- *Texture* **get_default_texture_param** (*String* param) const
- *bool* **has_param** (*String* name) const

9.270 ShaderGraph

Inherits: *Shader* < *Resource* < *Reference* < *Object*

Inherited By: *MaterialShaderGraph*, *CanvasItemShaderGraph*

Category: Core

9.270.1 Brief Description

9.270.2 Member Functions

void	*node_add* (*int* shader_type, *int* node_type, *int* id)
void	*node_remove* (*int* shader_type, *int* id)
void	*node_set_pos* (*int* shader_type, *int* id, *Vector2* pos)
Vector2	*node_get_pos* (*int* shader_type, *int* id) const
int	*node_get_type* (*int* shader_type, *int* id) const
Array	*get_node_list* (*int* shader_type) const
void	*default_set_value* (*int* shader_type, *int* id, *int* param_id, var value)
void	*default_get_value* (*int* shader_type, *int* id, *int* param_id)
void	*scalar_const_node_set_value* (*int* shader_type, *int* id, *float* value)
float	*scalar_const_node_get_value* (*int* shader_type, *int* id) const
void	*vec_const_node_set_value* (*int* shader_type, *int* id, *Vector3* value)

Continúa en la página siguiente

Tabla 9.26 – proviene de la página anterior

Vector3	*vec_const_node_get_value* (*int* shader_type, *int* id) const
void	*rgb_const_node_set_value* (*int* shader_type, *int* id, *Color* value)
Color	*rgb_const_node_get_value* (*int* shader_type, *int* id) const
void	*xform_const_node_set_value* (*int* shader_type, *int* id, *Transform* value)
Transform	*xform_const_node_get_value* (*int* shader_type, *int* id) const
void	*texture_node_set_filter_size* (*int* shader_type, *int* id, *int* filter_size)
int	*texture_node_get_filter_size* (*int* shader_type, *int* id) const
void	*texture_node_set_filter_strength* (*int* shader_type, *float* id, *float* filter_strength)
float	*texture_node_get_filter_strength* (*int* shader_type, *float* id) const
void	*scalar_op_node_set_op* (*int* shader_type, *float* id, *int* op)
int	*scalar_op_node_get_op* (*int* shader_type, *float* id) const
void	*vec_op_node_set_op* (*int* shader_type, *float* id, *int* op)
int	*vec_op_node_get_op* (*int* shader_type, *float* id) const
void	*vec_scalar_op_node_set_op* (*int* shader_type, *float* id, *int* op)
int	*vec_scalar_op_node_get_op* (*int* shader_type, *float* id) const
void	*rgb_op_node_set_op* (*int* shader_type, *float* id, *int* op)
int	*rgb_op_node_get_op* (*int* shader_type, *float* id) const
void	*xform_vec_mult_node_set_no_translation* (*int* shader_type, *int* id, *bool* disable)
bool	*xform_vec_mult_node_get_no_translation* (*int* shader_type, *int* id) const
void	*scalar_func_node_set_function* (*int* shader_type, *int* id, *int* func)
int	*scalar_func_node_get_function* (*int* shader_type, *int* id) const
void	*vec_func_node_set_function* (*int* shader_type, *int* id, *int* func)
int	*vec_func_node_get_function* (*int* shader_type, *int* id) const
void	*input_node_set_name* (*int* shader_type, *int* id, *String* name)
String	*input_node_get_name* (*int* shader_type, *int* id)
void	*scalar_input_node_set_value* (*int* shader_type, *int* id, *float* value)
float	*scalar_input_node_get_value* (*int* shader_type, *int* id) const
void	*vec_input_node_set_value* (*int* shader_type, *int* id, *Vector3* value)
Vector3	*vec_input_node_get_value* (*int* shader_type, *int* id) const
void	*rgb_input_node_set_value* (*int* shader_type, *int* id, *Color* value)
Color	*rgb_input_node_get_value* (*int* shader_type, *int* id) const
void	*xform_input_node_set_value* (*int* shader_type, *int* id, *Transform* value)
Transform	*xform_input_node_get_value* (*int* shader_type, *int* id) const
void	*texture_input_node_set_value* (*int* shader_type, *int* id, *Texture* value)
Texture	*texture_input_node_get_value* (*int* shader_type, *int* id) const
void	*cubemap_input_node_set_value* (*int* shader_type, *int* id, *CubeMap* value)
CubeMap	*cubemap_input_node_get_value* (*int* shader_type, *int* id) const
void	*comment_node_set_text* (*int* shader_type, *int* id, *String* text)
String	*comment_node_get_text* (*int* shader_type, *int* id) const
void	*color_ramp_node_set_ramp* (*int* shader_type, *int* id, *ColorArray* colors, *RealArray* offsets)
ColorArray	*color_ramp_node_get_colors* (*int* shader_type, *int* id) const
RealArray	*color_ramp_node_get_offsets* (*int* shader_type, *int* id) const
void	*curve_map_node_set_points* (*int* shader_type, *int* id, *Vector2Array* points)
Vector2Array	*curve_map_node_get_points* (*int* shader_type, *int* id) const
Error	*connect_node* (*int* shader_type, *int* src_id, *int* src_slot, *int* dst_id, *int* dst_slot)
bool	*is_node_connected* (*int* shader_type, *int* src_id, *int* src_slot, *int* dst_id, *int* dst_slot) const
void	*disconnect_node* (*int* shader_type, *int* src_id, *int* src_slot, *int* dst_id, *int* dst_slot)
Array	*get_node_connections* (*int* shader_type) const
void	*clear* (*int* shader_type)
void	*node_set_state* (*int* shader_type, *int* id, var state)

Continúa en la página siguiente

Tabla 9.26 – proviene de la página anterior

Variant	*node_get_state* (*int* shader_type, *int* id) const

9.270.3 Signals

- **updated** ( )

9.270.4 Numeric Constants

- **NODE_INPUT = 0**
- **NODE_SCALAR_CONST = 1**
- **NODE_VEC_CONST = 2**
- **NODE_RGB_CONST = 3**
- **NODE_XFORM_CONST = 4**
- **NODE_TIME = 5**
- **NODE_SCREEN_TEX = 6**
- **NODE_SCALAR_OP = 7**
- **NODE_VEC_OP = 8**
- **NODE_VEC_SCALAR_OP = 9**
- **NODE_RGB_OP = 10**
- **NODE_XFORM_MULT = 11**
- **NODE_XFORM_VEC_MULT = 12**
- **NODE_XFORM_VEC_INV_MULT = 13**
- **NODE_SCALAR_FUNC = 14**
- **NODE_VEC_FUNC = 15**
- **NODE_VEC_LEN = 16**
- **NODE_DOT_PROD = 17**
- **NODE_VEC_TO_SCALAR = 18**
- **NODE_SCALAR_TO_VEC = 19**
- **NODE_VEC_TO_XFORM = 21**
- **NODE_XFORM_TO_VEC = 20**
- **NODE_SCALAR_INTERP = 22**
- **NODE_VEC_INTERP = 23**
- **NODE_COLOR_RAMP = 24**
- **NODE_CURVE_MAP = 25**
- **NODE_SCALAR_INPUT = 26**
- **NODE_VEC_INPUT = 27**
- **NODE_RGB_INPUT = 28**

- **NODE_XFORM_INPUT = 29**
- **NODE_TEXTURE_INPUT = 30**
- **NODE_CUBEMAP_INPUT = 31**
- **NODE_DEFAULT_TEXTURE = 32**
- **NODE_OUTPUT = 33**
- **NODE_COMMENT = 34**
- **NODE_TYPE_MAX = 35**
- **SLOT_TYPE_SCALAR = 0**
- **SLOT_TYPE_VEC = 1**
- **SLOT_TYPE_XFORM = 2**
- **SLOT_TYPE_TEXTURE = 3**
- **SLOT_MAX = 4**
- **SHADER_TYPE_VERTEX = 0**
- **SHADER_TYPE_FRAGMENT = 1**
- **SHADER_TYPE_LIGHT = 2**
- **SHADER_TYPE_MAX = 3**
- **SLOT_IN = 0**
- **SLOT_OUT = 1**
- **GRAPH_OK = 0**
- **GRAPH_ERROR_CYCLIC = 1**
- **GRAPH_ERROR_MISSING_CONNECTIONS = 2**
- **SCALAR_OP_ADD = 0**
- **SCALAR_OP_SUB = 1**
- **SCALAR_OP_MUL = 2**
- **SCALAR_OP_DIV = 3**
- **SCALAR_OP_MOD = 4**
- **SCALAR_OP_POW = 5**
- **SCALAR_OP_MAX = 6**
- **SCALAR_OP_MIN = 7**
- **SCALAR_OP_ATAN2 = 8**
- **SCALAR_MAX_OP = 9**
- **VEC_OP_ADD = 0**
- **VEC_OP_SUB = 1**
- **VEC_OP_MUL = 2**
- **VEC_OP_DIV = 3**
- **VEC_OP_MOD = 4**

- **VEC_OP_POW = 5**
- **VEC_OP_MAX = 6**
- **VEC_OP_MIN = 7**
- **VEC_OP_CROSS = 8**
- **VEC_MAX_OP = 9**
- **VEC_SCALAR_OP_MUL = 0**
- **VEC_SCALAR_OP_DIV = 1**
- **VEC_SCALAR_OP_POW = 2**
- **VEC_SCALAR_MAX_OP = 3**
- **RGB_OP_SCREEN = 0**
- **RGB_OP_DIFFERENCE = 1**
- **RGB_OP_DARKEN = 2**
- **RGB_OP_LIGHTEN = 3**
- **RGB_OP_OVERLAY = 4**
- **RGB_OP_DODGE = 5**
- **RGB_OP_BURN = 6**
- **RGB_OP_SOFT_LIGHT = 7**
- **RGB_OP_HARD_LIGHT = 8**
- **RGB_MAX_OP = 9**
- **SCALAR_FUNC_SIN = 0**
- **SCALAR_FUNC_COS = 1**
- **SCALAR_FUNC_TAN = 2**
- **SCALAR_FUNC_ASIN = 3**
- **SCALAR_FUNC_ACOS = 4**
- **SCALAR_FUNC_ATAN = 5**
- **SCALAR_FUNC_SINH = 6**
- **SCALAR_FUNC_COSH = 7**
- **SCALAR_FUNC_TANH = 8**
- **SCALAR_FUNC_LOG = 9**
- **SCALAR_FUNC_EXP = 10**
- **SCALAR_FUNC_SQRT = 11**
- **SCALAR_FUNC_ABS = 12**
- **SCALAR_FUNC_SIGN = 13**
- **SCALAR_FUNC_FLOOR = 14**
- **SCALAR_FUNC_ROUND = 15**
- **SCALAR_FUNC_CEIL = 16**

- **SCALAR_FUNC_FRAC = 17**
- **SCALAR_FUNC_SATURATE = 18**
- **SCALAR_FUNC_NEGATE = 19**
- **SCALAR_MAX_FUNC = 20**
- **VEC_FUNC_NORMALIZE = 0**
- **VEC_FUNC_SATURATE = 1**
- **VEC_FUNC_NEGATE = 2**
- **VEC_FUNC_RECIPROCAL = 3**
- **VEC_FUNC_RGB2HSV = 4**
- **VEC_FUNC_HSV2RGB = 5**
- **VEC_MAX_FUNC = 6**

9.270.5 Member Function Description

- void **node_add** (*int* shader_type, *int* node_type, *int* id)
- void **node_remove** (*int* shader_type, *int* id)
- void **node_set_pos** (*int* shader_type, *int* id, *Vector2* pos)
- *Vector2* **node_get_pos** (*int* shader_type, *int* id) const
- *int* **node_get_type** (*int* shader_type, *int* id) const
- *Array* **get_node_list** (*int* shader_type) const
- void **default_set_value** (*int* shader_type, *int* id, *int* param_id, var value)
- void **default_get_value** (*int* shader_type, *int* id, *int* param_id)
- void **scalar_const_node_set_value** (*int* shader_type, *int* id, *float* value)
- *float* **scalar_const_node_get_value** (*int* shader_type, *int* id) const
- void **vec_const_node_set_value** (*int* shader_type, *int* id, *Vector3* value)
- *Vector3* **vec_const_node_get_value** (*int* shader_type, *int* id) const
- void **rgb_const_node_set_value** (*int* shader_type, *int* id, *Color* value)
- *Color* **rgb_const_node_get_value** (*int* shader_type, *int* id) const
- void **xform_const_node_set_value** (*int* shader_type, *int* id, *Transform* value)
- *Transform* **xform_const_node_get_value** (*int* shader_type, *int* id) const
- void **texture_node_set_filter_size** (*int* shader_type, *int* id, *int* filter_size)
- *int* **texture_node_get_filter_size** (*int* shader_type, *int* id) const
- void **texture_node_set_filter_strength** (*int* shader_type, *float* id, *float* filter_strength)
- *float* **texture_node_get_filter_strength** (*int* shader_type, *float* id) const
- void **scalar_op_node_set_op** (*int* shader_type, *float* id, *int* op)
- *int* **scalar_op_node_get_op** (*int* shader_type, *float* id) const
- void **vec_op_node_set_op** (*int* shader_type, *float* id, *int* op)

- *int* **vec_op_node_get_op** (*int* shader_type, *float* id) const
- void **vec_scalar_op_node_set_op** (*int* shader_type, *float* id, *int* op)
- *int* **vec_scalar_op_node_get_op** (*int* shader_type, *float* id) const
- void **rgb_op_node_set_op** (*int* shader_type, *float* id, *int* op)
- *int* **rgb_op_node_get_op** (*int* shader_type, *float* id) const
- void **xform_vec_mult_node_set_no_translation** (*int* shader_type, *int* id, *bool* disable)
- *bool* **xform_vec_mult_node_get_no_translation** (*int* shader_type, *int* id) const
- void **scalar_func_node_set_function** (*int* shader_type, *int* id, *int* func)
- *int* **scalar_func_node_get_function** (*int* shader_type, *int* id) const
- void **vec_func_node_set_function** (*int* shader_type, *int* id, *int* func)
- *int* **vec_func_node_get_function** (*int* shader_type, *int* id) const
- void **input_node_set_name** (*int* shader_type, *int* id, *String* name)
- *String* **input_node_get_name** (*int* shader_type, *int* id)
- void **scalar_input_node_set_value** (*int* shader_type, *int* id, *float* value)
- *float* **scalar_input_node_get_value** (*int* shader_type, *int* id) const
- void **vec_input_node_set_value** (*int* shader_type, *int* id, *Vector3* value)
- *Vector3* **vec_input_node_get_value** (*int* shader_type, *int* id) const
- void **rgb_input_node_set_value** (*int* shader_type, *int* id, *Color* value)
- *Color* **rgb_input_node_get_value** (*int* shader_type, *int* id) const
- void **xform_input_node_set_value** (*int* shader_type, *int* id, *Transform* value)
- *Transform* **xform_input_node_get_value** (*int* shader_type, *int* id) const
- void **texture_input_node_set_value** (*int* shader_type, *int* id, *Texture* value)
- *Texture* **texture_input_node_get_value** (*int* shader_type, *int* id) const
- void **cubemap_input_node_set_value** (*int* shader_type, *int* id, *CubeMap* value)
- *CubeMap* **cubemap_input_node_get_value** (*int* shader_type, *int* id) const
- void **comment_node_set_text** (*int* shader_type, *int* id, *String* text)
- *String* **comment_node_get_text** (*int* shader_type, *int* id) const
- void **color_ramp_node_set_ramp** (*int* shader_type, *int* id, *ColorArray* colors, *RealArray* offsets)
- *ColorArray* **color_ramp_node_get_colors** (*int* shader_type, *int* id) const
- *RealArray* **color_ramp_node_get_offsets** (*int* shader_type, *int* id) const
- void **curve_map_node_set_points** (*int* shader_type, *int* id, *Vector2Array* points)
- *Vector2Array* **curve_map_node_get_points** (*int* shader_type, *int* id) const
- Error **connect_node** (*int* shader_type, *int* src_id, *int* src_slot, *int* dst_id, *int* dst_slot)
- *bool* **is_node_connected** (*int* shader_type, *int* src_id, *int* src_slot, *int* dst_id, *int* dst_slot) const
- void **disconnect_node** (*int* shader_type, *int* src_id, *int* src_slot, *int* dst_id, *int* dst_slot)
- *Array* **get_node_connections** (*int* shader_type) const

- void **clear** (*int* shader_type)
- void **node_set_state** (*int* shader_type, *int* id, var state)
- Variant **node_get_state** (*int* shader_type, *int* id) const

9.271 ShaderMaterial

Inherits: *Material* < *Resource* < *Reference* < *Object*

Category: Core

9.271.1 Brief Description

9.271.2 Member Functions

void	*set_shader* (*Shader* shader)
Shader	*get_shader* ( ) const
void	*set_shader_param* (*String* param, Variant value)
Variant	*get_shader_param* (*String* param) const

9.271.3 Member Function Description

- void **set_shader** (*Shader* shader)
- *Shader* **get_shader** ( ) const
- void **set_shader_param** (*String* param, Variant value)
- Variant **get_shader_param** (*String* param) const

9.272 Shape

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *SphereShape*, *PlaneShape*, *CapsuleShape*, *BoxShape*, *ConvexPolygonShape*, *RayShape*, *ConcavePolygonShape*

Category: Core

9.272.1 Brief Description

9.273 Shape2D

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *RayShape2D*, *CapsuleShape2D*, *LineShape2D*, *CircleShape2D*, *ConcavePolygonShape2D*, *ConvexPolygonShape2D*, *RectangleShape2D*, *SegmentShape2D*

Category: Core

9.273.1 Brief Description

Base class for all 2D Shapes.

9.273.2 Member Functions

void	*set_custom_solver_bias* (*float* bias)
float	*get_custom_solver_bias* ( ) const
bool	*collide* (*Matrix32* local_xform, *Shape2D* with_shape, *Matrix32* shape_xform)
bool	*collide_with_motion* (*Matrix32* local_xform, *Vector2* local_motion, *Shape2D* with_shape, *Matrix32* shape_xform, *Vector2* shape_motion)
Variant	*collide_and_get_contacts* (*Matrix32* local_xform, *Shape2D* with_shape, *Matrix32* shape_xform)
Variant	*collide_with_motion_and_get_contacts* (*Matrix32* local_xform, *Vector2* local_motion, *Shape2D* with_shape, *Matrix32* shape_xform, *Vector2* shape_motion)

9.273.3 Description

Base class for all 2D Shapes. All 2D shape types inherit from this.

9.273.4 Member Function Description

- void **set_custom_solver_bias** (*float* bias)

Use a custom solver bias. No need to change this unless you really know what you are doing.

The solver bias is a factor controlling how much two objects "rebound" off each other, when colliding, to avoid them getting into each other because of numerical imprecision.

- *float* **get_custom_solver_bias** ( ) const

Return the custom solver bias.

- *bool* **collide** (*Matrix32* local_xform, *Shape2D* with_shape, *Matrix32* shape_xform)

Return whether this shape is colliding with another.

This method needs the transformation matrix for this shape (`local_xform`), the shape to check collisions with (`with_shape`), and the transformation matrix of that shape (`shape_xform`).

- *bool* **collide_with_motion** (*Matrix32* local_xform, *Vector2* local_motion, *Shape2D* with_shape, *Matrix32* shape_xform, *Vector2* shape_motion)

Return whether this shape would collide with another, if a given movement was applied.

This method needs the transformation matrix for this shape (`local_xform`), the movement to test on this shape (`local_motion`), the shape to check collisions with (`with_shape`), the transformation matrix of that shape (`shape_xform`), and the movement to test onto the other object (`shape_motion`).

- Variant **collide_and_get_contacts** (*Matrix32* local_xform, *Shape2D* with_shape, *Matrix32* shape_xform)

Return a list of the points where this shape touches another. If there are no collisions, the list is empty.

This method needs the transformation matrix for this shape (`local_xform`), the shape to check collisions with (`with_shape`), and the transformation matrix of that shape (`shape_xform`).

- Variant **collide_with_motion_and_get_contacts** (*Matrix32* local_xform, *Vector2* local_motion, *Shape2D* with_shape, *Matrix32* shape_xform, *Vector2* shape_motion)

Return a list of the points where this shape would touch another, if a given movement was applied. If there are no collisions, the list is empty.

This method needs the transformation matrix for this shape (`local_xform`), the movement to test on this shape (`local_motion`), the shape to check collisions with (`with_shape`), the transformation matrix of that shape (`shape_xform`), and the movement to test onto the other object (`shape_motion`).

9.274 Skeleton

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.274.1 Brief Description

Skeleton for characters and animated objects.

9.274.2 Member Functions

void	*add_bone* (*String* name)
int	*find_bone* (*String* name) const
String	*get_bone_name* (*int* bone_idx) const
int	*get_bone_parent* (*int* bone_idx) const
void	*set_bone_parent* (*int* bone_idx, *int* parent_idx)
int	*get_bone_count* ( ) const
void	*unparent_bone_and_rest* (*int* bone_idx)
Transform	*get_bone_rest* (*int* bone_idx) const
void	*set_bone_rest* (*int* bone_idx, *Transform* rest)
void	*set_bone_disable_rest* (*int* bone_idx, *bool* disable)
bool	*is_bone_rest_disabled* (*int* bone_idx) const
void	*bind_child_node_to_bone* (*int* bone_idx, *Node* node)
void	*unbind_child_node_from_bone* (*int* bone_idx, *Node* node)
Array	*get_bound_child_nodes_to_bone* (*int* bone_idx) const
void	*clear_bones* ( )
Transform	*get_bone_pose* (*int* bone_idx) const
void	*set_bone_pose* (*int* bone_idx, *Transform* pose)
void	*set_bone_global_pose* (*int* bone_idx, *Transform* pose)
Transform	*get_bone_global_pose* (*int* bone_idx) const
Transform	*get_bone_custom_pose* (*int* bone_idx) const
void	*set_bone_custom_pose* (*int* bone_idx, *Transform* custom_pose)
Transform	*get_bone_transform* (*int* bone_idx) const

9.274.3 Numeric Constants

- **NOTIFICATION_UPDATE_SKELETON = 50**

9.274.4 Description

Skeleton provides a hierarchical interface for managing bones, including pose, rest and animation (see *Animation*). Skeleton will support rag doll dynamics in the future.

9.274.5 Member Function Description

- void **add_bone** (*String* name)

Add a bone, with name "name". *get_bone_count* will become the bone index.

- *int* **find_bone** (*String* name) const

Return the bone index that matches "name" as its name.

- *String* **get_bone_name** (*int* bone_idx) const

Return the name of the bone at index "index"

- *int* **get_bone_parent** (*int* bone_idx) const

Return the bone index which is the parent of the bone at "bone_idx". If -1, then bone has no parent. Note that the parent bone returned will always be less than "bone_idx".

- void **set_bone_parent** (*int* bone_idx, *int* parent_idx)

Set the bone index "parent_idx" as the parent of the bone at "bone_idx". If -1, then bone has no parent. Note: "parent_idx" must be less than "bone_idx".

- *int* **get_bone_count** ( ) const

Return the amount of bones in the skeleton.

- void **unparent_bone_and_rest** (*int* bone_idx)
- *Transform* **get_bone_rest** (*int* bone_idx) const

Return the rest transform for a bone "bone_idx".

- void **set_bone_rest** (*int* bone_idx, *Transform* rest)

Set the rest transform for bone "bone_idx"

- void **set_bone_disable_rest** (*int* bone_idx, *bool* disable)
- *bool* **is_bone_rest_disabled** (*int* bone_idx) const
- void **bind_child_node_to_bone** (*int* bone_idx, *Node* node)

Deprecated soon.

- void **unbind_child_node_from_bone** (*int* bone_idx, *Node* node)

Deprecated soon.

- *Array* **get_bound_child_nodes_to_bone** (*int* bone_idx) const

Deprecated soon.

- void **clear_bones** ( )

Clear all the bones in this skeleton.

- *Transform* **get_bone_pose** (*int* bone_idx) const

Return the pose transform for bone "bone_idx".

- void **set_bone_pose** (*int* bone_idx, *Transform* pose)

Return the pose transform for bone "bone_idx".

- void **set_bone_global_pose** (*int* bone_idx, *Transform* pose)
- *Transform* **get_bone_global_pose** (*int* bone_idx) const
- *Transform* **get_bone_custom_pose** (*int* bone_idx) const
- void **set_bone_custom_pose** (*int* bone_idx, *Transform* custom_pose)
- *Transform* **get_bone_transform** (*int* bone_idx) const

9.275 Slider

Inherits: *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *HSlider*, *VSlider*

Category: Core

9.275.1 Brief Description

Base class for GUI Sliders.

9.275.2 Member Functions

void	*set_ticks* (*int* count)
int	*get_ticks* ( ) const
bool	*get_ticks_on_borders* ( ) const
void	*set_ticks_on_borders* (*bool* ticks_on_border)

9.275.3 Description

Base class for GUI Sliders.

9.275.4 Member Function Description

- void **set_ticks** (*int* count)

Set amount of ticks to display in slider.

- *int* **get_ticks** ( ) const

Return amounts of ticks to display on slider.

- *bool* **get_ticks_on_borders** ( ) const

Return true if ticks are visible on borders.

- void **set_ticks_on_borders** (*bool* ticks_on_border)

Set true if ticks are visible on borders.

9.276 SliderJoint

Inherits: *Joint* < *Spatial* < *Node* < *Object*

Category: Core

9.276.1 Brief Description

9.276.2 Member Functions

void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const

9.276.3 Numeric Constants

- **PARAM_LINEAR_LIMIT_UPPER = 0**
- **PARAM_LINEAR_LIMIT_LOWER = 1**
- **PARAM_LINEAR_LIMIT_SOFTNESS = 2**
- **PARAM_LINEAR_LIMIT_RESTITUTION = 3**
- **PARAM_LINEAR_LIMIT_DAMPING = 4**
- **PARAM_LINEAR_MOTION_SOFTNESS = 5**
- **PARAM_LINEAR_MOTION_RESTITUTION = 6**
- **PARAM_LINEAR_MOTION_DAMPING = 7**
- **PARAM_LINEAR_ORTHOGONAL_SOFTNESS = 8**
- **PARAM_LINEAR_ORTHOGONAL_RESTITUTION = 9**
- **PARAM_LINEAR_ORTHOGONAL_DAMPING = 10**
- **PARAM_ANGULAR_LIMIT_UPPER = 11**
- **PARAM_ANGULAR_LIMIT_LOWER = 12**
- **PARAM_ANGULAR_LIMIT_SOFTNESS = 13**
- **PARAM_ANGULAR_LIMIT_RESTITUTION = 14**
- **PARAM_ANGULAR_LIMIT_DAMPING = 15**
- **PARAM_ANGULAR_MOTION_SOFTNESS = 16**
- **PARAM_ANGULAR_MOTION_RESTITUTION = 17**
- **PARAM_ANGULAR_MOTION_DAMPING = 18**
- **PARAM_ANGULAR_ORTHOGONAL_SOFTNESS = 19**
- **PARAM_ANGULAR_ORTHOGONAL_RESTITUTION = 20**
- **PARAM_ANGULAR_ORTHOGONAL_DAMPING = 21**
- **PARAM_MAX = 22**

9.276.4 Member Function Description

- void **set_param** (*int* param, *float* value)
- *float* **get_param** (*int* param) const

9.277 SoundPlayer2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Inherited By: *SamplePlayer2D*

Category: Core

9.277.1 Brief Description

Base class for playing spatial 2D sound.

9.277.2 Member Functions

void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const

9.277.3 Numeric Constants

- **PARAM_VOLUME_DB = 0**
- **PARAM_PITCH_SCALE = 1**
- **PARAM_ATTENUATION_MIN_DISTANCE = 2**
- **PARAM_ATTENUATION_MAX_DISTANCE = 3**
- **PARAM_ATTENUATION_DISTANCE_EXP = 4**
- **PARAM_MAX = 5**

9.277.4 Description

Base class for playing spatial 2D sound.

9.277.5 Member Function Description

- void **set_param** (*int* param, *float* value)
- *float* **get_param** (*int* param) const

9.278 SoundRoomParams

Inherits: *Node* < *Object*

Category: Core

9.278.1 Brief Description

9.278.2 Member Functions

void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const
void	*set_reverb_mode* (*int* reverb_mode)
int	*get_reverb_mode* ( ) const
void	*set_force_params_to_all_sources* (*bool* enabled)
bool	*is_forcing_params_to_all_sources* ( )

9.278.3 Member Function Description

- void **set_param** (*int* param, *float* value)
- *float* **get_param** (*int* param) const
- void **set_reverb_mode** (*int* reverb_mode)
- *int* **get_reverb_mode** ( ) const
- void **set_force_params_to_all_sources** (*bool* enabled)
- *bool* **is_forcing_params_to_all_sources** ( )

9.279 Spatial

Inherits: *Node* < *Object*

Inherited By: *Joint*, *RayCast*, *Camera*, *BoneAttachment*, *CollisionShape*, *Path*, *VisualInstance*, *VehicleWheel*, *Position3D*, *ProximityGroup*, *SpatialPlayer*, *WorldEnvironment*, *PathFollow*, *NavigationMeshInstance*, *VisibilityNotifier*, *Navigation*, *CollisionPolygon*, *GridMap*, *Skeleton*, *CollisionObject*

Category: Core

9.279.1 Brief Description

Base class for all 3D nodes.

9.279.2 Member Functions

void	*set_transform* (*Transform* local)
Transform	*get_transform* ( ) const
void	*set_translation* (*Vector3* translation)

Continúa en la página siguiente

Tabla 9.27 – proviene de la página anterior

Vector3	*get_translation* ( ) const
void	*set_rotation* (*Vector3* rotation_rad)
Vector3	*get_rotation* ( ) const
void	*set_rotation_deg* (*Vector3* rotation_deg)
Vector3	*get_rotation_deg* ( ) const
void	*set_scale* (*Vector3* scale)
Vector3	*get_scale* ( ) const
void	*set_global_transform* (*Transform* global)
Transform	*get_global_transform* ( ) const
Object	*get_parent_spatial* ( ) const
void	*set_ignore_transform_notification* (*bool* enabled)
void	*set_as_toplevel* (*bool* enable)
bool	*is_set_as_toplevel* ( ) const
World	*get_world* ( ) const
void	*update_gizmo* ( )
void	*set_gizmo* (SpatialGizmo gizmo)
SpatialGizmo	*get_gizmo* ( ) const
void	*show* ( )
void	*hide* ( )
bool	*is_visible* ( ) const
bool	*is_hidden* ( ) const
void	*set_hidden* (*bool* hidden)
void	*set_notify_local_transform* (*bool* enable)
bool	*is_local_transform_notification_enabled* ( ) const
void	*rotate* (*Vector3* normal, *float* radians)
void	*global_rotate* (*Vector3* normal, *float* radians)
void	*rotate_x* (*float* radians)
void	*rotate_y* (*float* radians)
void	*rotate_z* (*float* radians)
void	*translate* (*Vector3* offset)
void	*global_translate* (*Vector3* offset)
void	*orthonormalize* ( )
void	*set_identity* ( )
void	*look_at* (*Vector3* target, *Vector3* up)
void	*look_at_from_pos* (*Vector3* pos, *Vector3* target, *Vector3* up)

9.279.3 Signals

- **visibility_changed** ( )

9.279.4 Numeric Constants

- **NOTIFICATION_TRANSFORM_CHANGED = 29** — Spatial nodes receive this notification with their global transform changes. This means that either the current or a parent node changed its transform.
- **NOTIFICATION_ENTER_WORLD = 41**
- **NOTIFICATION_EXIT_WORLD = 42**
- **NOTIFICATION_VISIBILITY_CHANGED = 43**

9.279.5 Description

Spatial is the base for every type of 3D *Node*. It contains a 3D *Transform* which can be set or get as local or global. If a Spatial *Node* has Spatial children, their transforms will be relative to the parent.

9.279.6 Member Function Description

- void **set_transform** (*Transform* local)

Set the transform locally, relative to the parent spatial node.

- *Transform* **get_transform** ( ) const

Return the local transform, relative to the bone parent.

- void **set_translation** (*Vector3* translation)
- *Vector3* **get_translation** ( ) const
- void **set_rotation** (*Vector3* rotation_rad)
- *Vector3* **get_rotation** ( ) const
- void **set_rotation_deg** (*Vector3* rotation_deg)
- *Vector3* **get_rotation_deg** ( ) const
- void **set_scale** (*Vector3* scale)
- *Vector3* **get_scale** ( ) const
- void **set_global_transform** (*Transform* global)

Set the transform globally, relative to worldspace.

- *Transform* **get_global_transform** ( ) const

Return the global transform, relative to worldspace.

- *Object* **get_parent_spatial** ( ) const

Return the parent *Spatial*, or an empty *Object* if no parent exists or parent is not of type *Spatial*.

- void **set_ignore_transform_notification** (*bool* enabled)
- void **set_as_toplevel** (*bool* enable)
- *bool* **is_set_as_toplevel** ( ) const
- *World* **get_world** ( ) const
- void **update_gizmo** ( )
- void **set_gizmo** (SpatialGizmo gizmo)
- SpatialGizmo **get_gizmo** ( ) const
- void **show** ( )
- void **hide** ( )
- *bool* **is_visible** ( ) const
- *bool* **is_hidden** ( ) const
- void **set_hidden** (*bool* hidden)
- void **set_notify_local_transform** (*bool* enable)

- *bool* **is_local_transform_notification_enabled** ( ) const
- void **rotate** (*Vector3* normal, *float* radians)
- void **global_rotate** (*Vector3* normal, *float* radians)
- void **rotate_x** (*float* radians)
- void **rotate_y** (*float* radians)
- void **rotate_z** (*float* radians)
- void **translate** (*Vector3* offset)
- void **global_translate** (*Vector3* offset)
- void **orthonormalize** ( )
- void **set_identity** ( )
- void **look_at** (*Vector3* target, *Vector3* up)
- void **look_at_from_pos** (*Vector3* pos, *Vector3* target, *Vector3* up)

9.280 SpatialPlayer

Inherits: *Spatial* < *Node* < *Object*

Inherited By: *SpatialStreamPlayer*, *SpatialSamplePlayer*

Category: Core

9.280.1 Brief Description

9.280.2 Member Functions

void	*set_param* (*int* param, *float* value)
float	*get_param* (*int* param) const

9.280.3 Numeric Constants

- **PARAM_VOLUME_DB = 0**
- **PARAM_PITCH_SCALE = 1**
- **PARAM_ATTENUATION_MIN_DISTANCE = 2**
- **PARAM_ATTENUATION_MAX_DISTANCE = 3**
- **PARAM_ATTENUATION_DISTANCE_EXP = 4**
- **PARAM_EMISSION_CONE_DEGREES = 5**
- **PARAM_EMISSION_CONE_ATTENUATION_DB = 6**
- **PARAM_MAX = 7**

9.280.4 Member Function Description

- void **set_param** (*int* param, *float* value)
- *float* **get_param** (*int* param) const

9.281 SpatialSamplePlayer

Inherits: *SpatialPlayer* < *Spatial* < *Node* < *Object*

Category: Core

9.281.1 Brief Description

9.281.2 Member Functions

void	*set_sample_library* (*SampleLibrary* library)
SampleLibrary	*get_sample_library* ( ) const
void	*set_polyphony* (*int* voices)
int	*get_polyphony* ( ) const
int	*play* (*String* sample, *int* voice=-2)
void	*voice_set_pitch_scale* (*int* voice, *float* ratio)
void	*voice_set_volume_scale_db* (*int* voice, *float* db)
bool	*is_voice_active* (*int* voice) const
void	*stop_voice* (*int* voice)
void	*stop_all* ( )

9.281.3 Numeric Constants

- **INVALID_VOICE** = **-1**
- **NEXT_VOICE** = **-2**

9.281.4 Member Function Description

- void **set_sample_library** (*SampleLibrary* library)
- *SampleLibrary* **get_sample_library** ( ) const
- void **set_polyphony** (*int* voices)
- *int* **get_polyphony** ( ) const
- *int* **play** (*String* sample, *int* voice=-2)
- void **voice_set_pitch_scale** (*int* voice, *float* ratio)
- void **voice_set_volume_scale_db** (*int* voice, *float* db)
- *bool* **is_voice_active** (*int* voice) const
- void **stop_voice** (*int* voice)
- void **stop_all** ( )

9.282 SpatialSound2DServer

Inherits: *Object*

Inherited By: *SpatialSound2DServerSW*

Category: Core

9.282.1 Brief Description

Server for Spatial 2D Sound.

9.282.2 Description

Server for Spatial 2D Sound.

9.283 SpatialSound2DServerSW

Inherits: *SpatialSound2DServer* < *Object*

Category: Core

9.283.1 Brief Description

9.284 SpatialSoundServer

Inherits: *Object*

Inherited By: *SpatialSoundServerSW*

Category: Core

9.284.1 Brief Description

9.285 SpatialSoundServerSW

Inherits: *SpatialSoundServer* < *Object*

Category: Core

9.285.1 Brief Description

9.286 SpatialStreamPlayer

Inherits: *SpatialPlayer* < *Spatial* < *Node* < *Object*

Category: Core

9.286.1 Brief Description

9.286.2 Member Functions

void	*set_stream* (*AudioStream* stream)
AudioStream	*get_stream* ( ) const
void	*play* (*float* offset=0)
void	*stop* ( )
bool	*is_playing* ( ) const
void	*set_paused* (*bool* paused)
bool	*is_paused* ( ) const
void	*set_loop* (*bool* enabled)
bool	*has_loop* ( ) const
void	*set_volume* (*float* volume)
float	*get_volume* ( ) const
void	*set_volume_db* (*float* db)
float	*get_volume_db* ( ) const
void	*set_buffering_msec* (*int* msec)
int	*get_buffering_msec* ( ) const
void	*set_loop_restart_time* (*float* secs)
float	*get_loop_restart_time* ( ) const
String	*get_stream_name* ( ) const
int	*get_loop_count* ( ) const
float	*get_pos* ( ) const
void	*seek_pos* (*float* time)
void	*set_autoplay* (*bool* enabled)
bool	*has_autoplay* ( ) const
float	*get_length* ( ) const

9.286.3 Member Function Description

- void **set_stream** (*AudioStream* stream)
- *AudioStream* **get_stream** ( ) const
- void **play** (*float* offset=0)
- void **stop** ( )
- *bool* **is_playing** ( ) const
- void **set_paused** (*bool* paused)
- *bool* **is_paused** ( ) const
- void **set_loop** (*bool* enabled)
- *bool* **has_loop** ( ) const
- void **set_volume** (*float* volume)
- *float* **get_volume** ( ) const
- void **set_volume_db** (*float* db)
- *float* **get_volume_db** ( ) const
- void **set_buffering_msec** (*int* msec)

- *int* **get_buffering_msec** ( ) const
- void **set_loop_restart_time** (*float* secs)
- *float* **get_loop_restart_time** ( ) const
- *String* **get_stream_name** ( ) const
- *int* **get_loop_count** ( ) const
- *float* **get_pos** ( ) const
- void **seek_pos** (*float* time)
- void **set_autoplay** (*bool* enabled)
- *bool* **has_autoplay** ( ) const
- *float* **get_length** ( ) const

9.287 SphereShape

Inherits: *Shape* < *Resource* < *Reference* < *Object*

Category: Core

9.287.1 Brief Description

9.287.2 Member Functions

void	*set_radius* (*float* radius)
float	*get_radius* ( ) const

9.287.3 Member Function Description

- void **set_radius** (*float* radius)
- *float* **get_radius** ( ) const

9.288 SpinBox

Inherits: *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.288.1 Brief Description

Numerical input text field.

9.288.2 Member Functions

void	*set_suffix* (*String* suffix)
String	*get_suffix* ( ) const
void	*set_prefix* (*String* prefix)
String	*get_prefix* ( ) const
void	*set_editable* (*bool* editable)
bool	*is_editable* ( ) const
Object	*get_line_edit* ( )

9.288.3 Description

SpinBox is a numerical input text field. It allows entering integers and floats.

9.288.4 Member Function Description

- void **set_suffix** (*String* suffix)

Set a specific suffix.

- *String* **get_suffix** ( ) const

Return the specific suffix.

- void **set_prefix** (*String* prefix)

Set a prefix.

- *String* **get_prefix** ( ) const
- void **set_editable** (*bool* editable)

Set whether the spinbox is editable.

- *bool* **is_editable** ( ) const

Return if the spinbox is editable.

- *Object* **get_line_edit** ( )

9.289 SplitContainer

Inherits: *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *HSplitContainer*, *VSplitContainer*

Category: Core

9.289.1 Brief Description

Container for splitting and adjusting.

9.289.2 Member Functions

void	*set_split_offset* (*int* offset)
int	*get_split_offset* ( ) const
void	*set_collapsed* (*bool* collapsed)
bool	*is_collapsed* ( ) const
void	*set_dragger_visibility* (*int* mode)
int	*get_dragger_visibility* ( ) const

9.289.3 Signals

- **dragged** (*int* offset)

9.289.4 Numeric Constants

- **DRAGGER_VISIBLE = 0** — The split dragger is visible.
- **DRAGGER_HIDDEN = 1** — The split dragger is invisible.
- **DRAGGER_HIDDEN_COLLAPSED = 2** — The split dragger is invisible and collapsed.

9.289.5 Description

Container for splitting two controls vertically or horizontally, with a grabber that allows adjusting the split offset or ratio.

9.289.6 Member Function Description

- void **set_split_offset** (*int* offset)

Set the split offset.

- *int* **get_split_offset** ( ) const

Return the split offset.

- void **set_collapsed** (*bool* collapsed)

Set if the split must be collapsed.

- *bool* **is_collapsed** ( ) const

Return true if the split is collapsed.

- void **set_dragger_visibility** (*int* mode)

Set visibility of the split dragger(*mode* must be one of DRAGGER_VISIBLE, DRAGGER_HIDDEN or DRAGGER_HIDDEN_COLLAPSED).

- *int* **get_dragger_visibility** ( ) const

Return visibility of the split dragger(One of DRAGGER_VISIBLE, DRAGGER_HIDDEN or DRAGGER_HIDDEN_COLLAPSED).

9.290 SpotLight

Inherits: *Light* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.290.1 Brief Description

Spotlight *Light*, such as a reflector spotlight or a lantern.

9.290.2 Description

A SpotLight light is a type of *Light* node that emits lights in a specific direction, in the shape of a cone. The light is attenuated through the distance and this attenuation can be configured by changing the energy, radius and attenuation parameters of *Light*. TODO: Image of a spotlight.

9.291 Sprite

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.291.1 Brief Description

General purpose Sprite node.

9.291.2 Member Functions

void	*set_texture* (*Texture* texture)
Texture	*get_texture* ( ) const
void	*set_centered* (*bool* centered)
bool	*is_centered* ( ) const
void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const
void	*set_flip_h* (*bool* flip_h)
bool	*is_flipped_h* ( ) const
void	*set_flip_v* (*bool* flip_v)
bool	*is_flipped_v* ( ) const
void	*set_region* (*bool* enabled)
bool	*is_region* ( ) const
void	*set_region_rect* (*Rect2* rect)
Rect2	*get_region_rect* ( ) const
void	*set_frame* (*int* frame)
int	*get_frame* ( ) const
void	*set_vframes* (*int* vframes)
int	*get_vframes* ( ) const
void	*set_hframes* (*int* hframes)
int	*get_hframes* ( ) const
void	*set_modulate* (*Color* modulate)
Color	*get_modulate* ( ) const

9.291.3 Signals

- **frame_changed** ( )

9.291.4 Description

General purpose Sprite node. This Sprite node can show any texture as a sprite. The texture can be used as a spritesheet for animation, or only a region from a bigger texture can referenced, like an atlas.

9.291.5 Member Function Description

- void **set_texture** (*Texture* texture)

Set the base texture for the sprite.

- *Texture* **get_texture** ( ) const

Return the base texture for the sprite.

- void **set_centered** (*bool* centered)

Set whether the sprite should be centered on the origin.

- *bool* **is_centered** ( ) const

Return if the sprite is centered at the local origin.

- void **set_offset** (*Vector2* offset)

Set the sprite draw offset, useful for setting rotation pivots.

- *Vector2* **get_offset** **(** **)** const

Return sprite draw offset.

- void **set_flip_h** **(** *bool* flip_h **)**

Set true to flip the sprite horizontally.

- *bool* **is_flipped_h** **(** **)** const

Return true if the sprite is flipped horizontally.

- void **set_flip_v** **(** *bool* flip_v **)**

Set true to flip the sprite vertically.

- *bool* **is_flipped_v** **(** **)** const

Return true if the sprite is flipped vertically.

- void **set_region** **(** *bool* enabled **)**

Set the sprite as a sub-region of a bigger texture. Useful for texture-atlases.

- *bool* **is_region** **(** **)** const

Return if the sprite reads from a region.

- void **set_region_rect** **(** *Rect2* rect **)**

Set the region rect to read from.

- *Rect2* **get_region_rect** **(** **)** const

Return the region rect to read from.

- void **set_frame** **(** *int* frame **)**

Set the texture frame for a sprite-sheet, works when vframes or hframes are greater than 1.

- *int* **get_frame** **(** **)** const

Return the texture frame for a sprite-sheet, works when vframes or hframes are greater than 1.

- void **set_vframes** **(** *int* vframes **)**

Set the amount of vertical frames and converts the sprite into a sprite-sheet. This is useful for animation.

- *int* **get_vframes** **(** **)** const

Return the amount of vertical frames. See *set_vframes*.

- void **set_hframes** **(** *int* hframes **)**

Set the amount of horizontal frames and converts the sprite into a sprite-sheet. This is useful for animation.

- *int* **get_hframes** **(** **)** const

Return the amount of horizontal frames. See *set_hframes*.

- void **set_modulate** **(** *Color* modulate **)**

Set color modulation for the sprite. All sprite pixels are multiplied by this color. Color may contain rgb values above 1 to achieve a highlight effect.

- *Color* **get_modulate** **(** **)** const

Return color modulation for the sprite. All sprite pixels are multiplied by this color.

9.292 Sprite3D

Inherits: *SpriteBase3D* < *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.292.1 Brief Description

9.292.2 Member Functions

void	*set_texture* (*Texture* texture)
Texture	*get_texture* ( ) const
void	*set_region* (*bool* enabled)
bool	*is_region* ( ) const
void	*set_region_rect* (*Rect2* rect)
Rect2	*get_region_rect* ( ) const
void	*set_frame* (*int* frame)
int	*get_frame* ( ) const
void	*set_vframes* (*int* vframes)
int	*get_vframes* ( ) const
void	*set_hframes* (*int* hframes)
int	*get_hframes* ( ) const

9.292.3 Signals

- **frame_changed** ( )

9.292.4 Member Function Description

- void **set_texture** (*Texture* texture)
- *Texture* **get_texture** ( ) const
- void **set_region** (*bool* enabled)
- *bool* **is_region** ( ) const
- void **set_region_rect** (*Rect2* rect)
- *Rect2* **get_region_rect** ( ) const
- void **set_frame** (*int* frame)
- *int* **get_frame** ( ) const
- void **set_vframes** (*int* vframes)
- *int* **get_vframes** ( ) const
- void **set_hframes** (*int* hframes)
- *int* **get_hframes** ( ) const

9.293 SpriteBase3D

Inherits: *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

Inherited By: *AnimatedSprite3D*, *Sprite3D*

Category: Core

9.293.1 Brief Description

9.293.2 Member Functions

void	*set_centered* (*bool* centered)
bool	*is_centered* ( ) const
void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const
void	*set_flip_h* (*bool* flip_h)
bool	*is_flipped_h* ( ) const
void	*set_flip_v* (*bool* flip_v)
bool	*is_flipped_v* ( ) const
void	*set_modulate* (*Color* modulate)
Color	*get_modulate* ( ) const
void	*set_opacity* (*float* opacity)
float	*get_opacity* ( ) const
void	*set_pixel_size* (*float* pixel_size)
float	*get_pixel_size* ( ) const
void	*set_axis* (*int* axis)
int	*get_axis* ( ) const
void	*set_draw_flag* (*int* flag, *bool* enabled)
bool	*get_draw_flag* (*int* flag) const
void	*set_alpha_cut_mode* (*int* mode)
int	*get_alpha_cut_mode* ( ) const
Rect2	*get_item_rect* ( ) const

9.293.3 Numeric Constants

- **FLAG_TRANSPARENT = 0**
- **FLAG_SHADED = 1**
- **FLAG_MAX = 2**
- **ALPHA_CUT_DISABLED = 0**
- **ALPHA_CUT_DISCARD = 1**
- **ALPHA_CUT_OPAQUE_PREPASS = 2**

9.293.4 Member Function Description

- void **set_centered** (*bool* centered)
- *bool* **is_centered** ( ) const

- void **set_offset** (*Vector2* offset)
- *Vector2* **get_offset** ( ) const
- void **set_flip_h** (*bool* flip_h)
- *bool* **is_flipped_h** ( ) const
- void **set_flip_v** (*bool* flip_v)
- *bool* **is_flipped_v** ( ) const
- void **set_modulate** (*Color* modulate)
- *Color* **get_modulate** ( ) const
- void **set_opacity** (*float* opacity)
- *float* **get_opacity** ( ) const
- void **set_pixel_size** (*float* pixel_size)
- *float* **get_pixel_size** ( ) const
- void **set_axis** (*int* axis)
- *int* **get_axis** ( ) const
- void **set_draw_flag** (*int* flag, *bool* enabled)
- *bool* **get_draw_flag** (*int* flag) const
- void **set_alpha_cut_mode** (*int* mode)
- *int* **get_alpha_cut_mode** ( ) const
- *Rect2* **get_item_rect** ( ) const

9.294 SpriteFrames

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.294.1 Brief Description

Sprite frame library for AnimatedSprite.

9.294.2 Member Functions

void	*add_frame* (*Object* frame, *int* atpos=-1)
int	*get_frame_count* ( ) const
Object	*get_frame* (*int* idx) const
void	*set_frame* (*int* idx, *Object* txt)
void	*remove_frame* (*int* idx)
void	*clear* ( )

9.294.3 Description

Sprite frame library for *AnimatedSprite*.

9.294.4 Member Function Description

- void **add_frame** (*Object* frame, *int* atpos=-1)

Add a frame (texture).

- *int* **get_frame_count** ( ) const

Return the amount of frames.

- *Object* **get_frame** (*int* idx) const

Return a texture (frame).

- void **set_frame** (*int* idx, *Object* txt)
- void **remove_frame** (*int* idx)

Remove a frame

- void **clear** ( )

Clear the frames.

9.295 StaticBody

Inherits: *PhysicsBody* < *CollisionObject* < *Spatial* < *Node* < *Object*

Category: Core

9.295.1 Brief Description

Static body for 3D Physics.

9.295.2 Member Functions

void	*set_constant_linear_velocity* (*Vector3* vel)
void	*set_constant_angular_velocity* (*Vector3* vel)
Vector3	*get_constant_linear_velocity* ( ) const
Vector3	*get_constant_angular_velocity* ( ) const
void	*set_friction* (*float* friction)
float	*get_friction* ( ) const
void	*set_bounce* (*float* bounce)
float	*get_bounce* ( ) const

9.295.3 Description

Static body for 3D Physics. A static body is a simple body that is not intended to move. They don't consume any CPU resources in contrast to a RigidBody3D so they are great for scenario collision.

A static body can also be animated by using simulated motion mode. This is useful for implementing functionalities such as moving platforms. When this mode is active the body can be animated and automatically computes linear and angular velocity to apply in that frame and to influence other bodies.

Alternatively, a constant linear or angular velocity can be set for the static body, so even if it doesn't move, it affects other bodies as if it was moving (this is useful for simulating conveyor belts or conveyor wheels).

9.295.4 Member Function Description

- void **set_constant_linear_velocity** **(** *Vector3* vel **)**

Set a constant linear velocity for the body. This does not move the body, but affects other bodies touching it, as if it was moving.

- void **set_constant_angular_velocity** **(** *Vector3* vel **)**

Set a constant angular velocity for the body. This does not rotate the body, but affects other bodies touching it, as if it was rotating.

- *Vector3* **get_constant_linear_velocity** **(** **)** const

Return the constant linear velocity for the body.

- *Vector3* **get_constant_angular_velocity** **(** **)** const

Return the constant angular velocity for the body.

- void **set_friction** **(** *float* friction **)**

Set the body friction, from 0 (frictionless) to 1 (full friction).

- *float* **get_friction** **(** **)** const

Return the body friction.

- void **set_bounce** **(** *float* bounce **)**

Set the body bounciness, from 0 (not bouncy) to 1 (bouncy).

- *float* **get_bounce** **(** **)** const

Return the body bounciness.

9.296 StaticBody2D

Inherits: *PhysicsBody2D* **<** *CollisionObject2D* **<** *Node2D* **<** *CanvasItem* **<** *Node* **<** *Object*

Category: Core

9.296.1 Brief Description

Static body for 2D Physics.

9.296.2 Member Functions

void	*set_constant_linear_velocity* (*Vector2* vel)
void	*set_constant_angular_velocity* (*float* vel)
Vector2	*get_constant_linear_velocity* ( ) const
float	*get_constant_angular_velocity* ( ) const
void	*set_friction* (*float* friction)
float	*get_friction* ( ) const
void	*set_bounce* (*float* bounce)
float	*get_bounce* ( ) const

9.296.3 Description

Static body for 2D Physics. A static body is a simple body that is not intended to move. They don't consume any CPU resources in contrast to a *RigidBody2D* so they are great for scenario collision.

A static body can also be animated by using simulated motion mode. This is useful for implementing functionalities such as moving platforms. When this mode is active the body can be animated and automatically computes linear and angular velocity to apply in that frame and to influence other bodies.

Alternatively, a constant linear or angular velocity can be set for the static body, so even if it doesn't move, it affects other bodies as if it was moving (this is useful for simulating conveyor belts or conveyor wheels).

9.296.4 Member Function Description

- void **set_constant_linear_velocity** (*Vector2* vel)

Set a constant linear velocity for the body. This does not move the body, but affects other bodies touching it, as if it was moving.

- void **set_constant_angular_velocity** (*float* vel)

Set a constant angular velocity for the body. This does not rotate the body, but affects other bodies touching it, as if it was rotating.

- *Vector2* **get_constant_linear_velocity** ( ) const

Return the constant linear velocity for the body.

- *float* **get_constant_angular_velocity** ( ) const

Return the constant angular velocity for the body.

- void **set_friction** (*float* friction)

Set the body friction, from 0 (frictionless) to 1 (full friction).

- *float* **get_friction** ( ) const

Return the body friction.

- void **set_bounce** (*float* bounce)

Set the body bounciness, from 0 (not bouncy) to 1 (bouncy).

- *float* **get_bounce** ( ) const

Return the body bounciness.

9.297 StreamPeer

Inherits: *Reference* < *Object*

Inherited By: *StreamPeerSSL*, *StreamPeerTCP*

Category: Core

9.297.1 Brief Description

Abstraction and base class for stream-based protocols.

9.297.2 Member Functions

int	*put_data* (*RawArray* data)
Array	*put_partial_data* (*RawArray* data)
Array	*get_data* (*int* bytes)
Array	*get_partial_data* (*int* bytes)
int	*get_available_bytes* ( ) const
void	*set_big_endian* (*bool* enable)
bool	*is_big_endian_enabled* ( ) const
void	*put_8* (*int* val)
void	*put_u8* (*int* val)
void	*put_16* (*int* val)
void	*put_u16* (*int* val)
void	*put_32* (*int* val)
void	*put_u32* (*int* val)
void	*put_64* (*int* val)
void	*put_u64* (*int* val)
void	*put_float* (*float* val)
void	*put_double* (*float* val)
void	*put_utf8_string* (*String* val)
void	*put_var* (Variant val)
int	*get_8* ( )
int	*get_u8* ( )
int	*get_16* ( )
int	*get_u16* ( )
int	*get_32* ( )
int	*get_u32* ( )
int	*get_64* ( )
int	*get_u64* ( )
float	*get_float* ( )
float	*get_double* ( )
String	*get_string* (*int* bytes)
String	*get_utf8_string* (*int* bytes)
Variant	*get_var* ( )

9.297.3 Description

StreamPeer is an abstraction and base class for stream-based protocols (such as TCP or Unix Sockets). It provides an API for sending and receiving data through streams as raw data or strings.

9.297.4 Member Function Description

- *int* **put_data** (*RawArray* data)

Send a chunk of data through the connection, blocking if necessary until the data is done sending. This function returns an Error code.

- *Array* **put_partial_data** (*RawArray* data)

Send a chunk of data through the connection, if all the data could not be sent at once, only part of it will. This function returns two values, an Error code and an integer, describing how much data was actually sent.

- *Array* **get_data** (*int* bytes)

Return a chunk data with the received bytes. The amount of bytes to be received can be requested in the "bytes" argument. If not enough bytes are available, the function will block until the desired amount is received. This function returns two values, an Error code and a data array.

- *Array* **get_partial_data** (*int* bytes)

Return a chunk data with the received bytes. The amount of bytes to be received can be requested in the "bytes" argument. If not enough bytes are available, the function will return how many were actually received. This function returns two values, an Error code, and a data array.

- *int* **get_available_bytes** ( ) const
- void **set_big_endian** (*bool* enable)
- *bool* **is_big_endian_enabled** ( ) const
- void **put_8** (*int* val)
- void **put_u8** (*int* val)
- void **put_16** (*int* val)
- void **put_u16** (*int* val)
- void **put_32** (*int* val)
- void **put_u32** (*int* val)
- void **put_64** (*int* val)
- void **put_u64** (*int* val)
- void **put_float** (*float* val)
- void **put_double** (*float* val)
- void **put_utf8_string** (*String* val)
- void **put_var** (Variant val)
- *int* **get_8** ( )
- *int* **get_u8** ( )
- *int* **get_16** ( )
- *int* **get_u16** ( )

- *int* **get_32** ( )
- *int* **get_u32** ( )
- *int* **get_64** ( )
- *int* **get_u64** ( )
- *float* **get_float** ( )
- *float* **get_double** ( )
- *String* **get_string** (*int* bytes)
- *String* **get_utf8_string** (*int* bytes)
- Variant **get_var** ( )

9.298 StreamPeerSSL

Inherits: *StreamPeer* < *Reference* < *Object*

Category: Core

9.298.1 Brief Description

9.298.2 Member Functions

Error	*accept* (*StreamPeer* stream)
Error	*connect* (*StreamPeer* stream, *bool* validate_certs=false, *String* for_hostname="")
int	*get_status* ( ) const
void	*disconnect* ( )

9.298.3 Numeric Constants

- **STATUS_DISCONNECTED = 0**
- **STATUS_CONNECTED = 1**
- **STATUS_ERROR_NO_CERTIFICATE = 2**
- **STATUS_ERROR_HOSTNAME_MISMATCH = 3**

9.298.4 Member Function Description

- Error **accept** (*StreamPeer* stream)
- Error **connect** (*StreamPeer* stream, *bool* validate_certs=false, *String* for_hostname="")
- *int* **get_status** ( ) const
- void **disconnect** ( )

9.299 StreamPeerTCP

Inherits: *StreamPeer* < *Reference* < *Object*

Category: Core

9.299.1 Brief Description

TCP Stream peer.

9.299.2 Member Functions

int	*connect* (*String* host, *int* port)
bool	*is_connected* ( ) const
int	*get_status* ( ) const
String	*get_connected_host* ( ) const
int	*get_connected_port* ( ) const
void	*disconnect* ( )

9.299.3 Numeric Constants

- **STATUS_NONE = 0**
- **STATUS_CONNECTING = 1**
- **STATUS_CONNECTED = 2**
- **STATUS_ERROR = 3**

9.299.4 Description

TCP Stream peer. This object can be used to connect to TCP servers, or also is returned by a tcp server.

9.299.5 Member Function Description

- *int* **connect** (*String* host, *int* port)
- *bool* **is_connected** ( ) const
- *int* **get_status** ( ) const
- *String* **get_connected_host** ( ) const
- *int* **get_connected_port** ( ) const
- void **disconnect** ( )

9.300 StreamPlayer

Inherits: *Node* < *Object*

Category: Core

9.300.1 Brief Description

Base class for audio stream playback.

9.300.2 Member Functions

void	*set_stream* (*AudioStream* stream)
AudioStream	*get_stream* ( ) const
void	*play* (*float* offset=0)
void	*stop* ( )
bool	*is_playing* ( ) const
void	*set_paused* (*bool* paused)
bool	*is_paused* ( ) const
void	*set_loop* (*bool* enabled)
bool	*has_loop* ( ) const
void	*set_volume* (*float* volume)
float	*get_volume* ( ) const
void	*set_volume_db* (*float* db)
float	*get_volume_db* ( ) const
void	*set_buffering_msec* (*int* msec)
int	*get_buffering_msec* ( ) const
void	*set_loop_restart_time* (*float* secs)
float	*get_loop_restart_time* ( ) const
String	*get_stream_name* ( ) const
int	*get_loop_count* ( ) const
float	*get_pos* ( ) const
void	*seek_pos* (*float* time)
void	*set_autoplay* (*bool* enabled)
bool	*has_autoplay* ( ) const
float	*get_length* ( ) const

9.300.3 Signals

- **finished** ( )

9.300.4 Description

Base class for audio stream playback. Audio stream players inherit from it.

9.300.5 Member Function Description

- void **set_stream** (*AudioStream* stream)

Set the *EventStream* this player will play.

- *AudioStream* **get_stream** ( ) const

Return the currently assigned stream.

- void **play** (*float* offset=0)

Play the currently assigned stream, starting from a given position (in seconds).

- void **stop** ( )

Stop the playback.

- *bool* **is_playing** ( ) const

Return whether this player is playing.

- void **set_paused** (*bool* paused)

Pause stream playback.

- *bool* **is_paused** ( ) const

Return whether the playback is currently paused.

- void **set_loop** (*bool* enabled)

Set whether the stream will be restarted at the end.

- *bool* **has_loop** ( ) const

Return whether the stream will be restarted at the end.

- void **set_volume** (*float* volume)

Set the playback volume for this player. This is a float between 0.0 (silent) and 1.0 (full volume). Values over 1.0 will amplify sound even more, but may introduce distortion. Negative values will just invert the output waveform, which produces no audible difference.

- *float* **get_volume** ( ) const

Return the playback volume for this player.

- void **set_volume_db** (*float* db)

Set the playback volume for this player, in decibels. This is a float between -80.0 (silent) and 0.0 (full volume). Values under -79.0 get truncated to -80, but values over 0.0 do not, so the warnings for overamplifying (see *set_volume*) still apply.

- *float* **get_volume_db** ( ) const

Return the playback volume for this player, in decibels.

- void **set_buffering_msec** (*int* msec)

Set the size (in milliseconds) of the audio buffer. A long audio buffer protects better against slowdowns, but responds worse to changes (in volume, stream played...). A shorter buffer takes less time to respond to changes, but may stutter if the application suffers some slowdown.

Default is 500 milliseconds.

- *int* **get_buffering_msec** ( ) const

Return the size of the audio buffer.

- void **set_loop_restart_time** (*float* secs)

Set the point in time the stream will rewind to, when looping.

- *float* **get_loop_restart_time** ( ) const

Return the point in time the stream will rewind to, when looping.

- *String* **get_stream_name** ( ) const

Return the name of the currently assigned stream. This is not the file name, but a field inside the file. If no stream is assigned, if returns "<No Stream>".

- *int* **get_loop_count** ( ) const

Return the number of times the playback has looped.

- *float* **get_pos** ( ) const

Return the playback position, in seconds.

- void **seek_pos** (*float* time)

Set the playback position, in seconds.

- void **set_autoplay** (*bool* enabled)

Set whether this player will start playing as soon as it enters the scene tree.

- *bool* **has_autoplay** ( ) const

Return whether this player will start playing as soon as it enters the scene tree.

- *float* **get_length** ( ) const

Return the length of the stream, in seconds.

9.301 String

Category: Built-In Types

9.301.1 Brief Description

Built-in string class.

9.301.2 Member Functions

String	*basename* ( )
bool	*begins_with* (*String* text)
String	*c_escape* ( )
String	*c_unescape* ( )
String	*capitalize* ( )
int	*casecmp_to* (*String* to)
bool	*empty* ( )
String	*extension* ( )
int	*find* (*String* what, *int* from=0)
int	*find_last* (*String* what)
int	*findn* (*String* what, *int* from=0)
String	*get_base_dir* ( )
String	*get_file* ( )
int	*hash* ( )
int	*hex_to_int* ( )
String	*insert* (*int* pos, *String* what)
bool	*is_abs_path* ( )
bool	*is_rel_path* ( )
bool	*is_valid_float* ( )

Continúa en la página siguiente

Tabla 9.29 – proviene de la página anterior

bool	*is_valid_html_color* ( )
bool	*is_valid_identifier* ( )
bool	*is_valid_integer* ( )
bool	*is_valid_ip_address* ( )
String	*json_escape* ( )
String	*left* (*int* pos)
int	*length* ( )
bool	*match* (*String* expr)
bool	*matchn* (*String* expr)
RawArray	*md5_buffer* ( )
String	*md5_text* ( )
int	*nocasecmp_to* (*String* to)
String	*ord_at* (*int* at)
String	*pad_decimals* (*int* digits)
String	*pad_zeros* (*int* digits)
String	*percent_decode* ( )
String	*percent_encode* ( )
String	*plus_file* (*String* file)
String	*replace* (*String* what, *String* forwhat)
String	*replacen* (*String* what, *String* forwhat)
int	*rfind* (*String* what, *int* from=-1)
int	*rfindn* (*String* what, *int* from=-1)
String	*right* (*int* pos)
StringArray	*split* (*String* divisor, *bool* allow_empty=True)
RealArray	*split_floats* (*String* divisor, *bool* allow_empty=True)
String	*strip_edges* (*bool* left=True, *bool* right=True)
String	*substr* (*int* from, *int* len)
RawArray	*to_ascii* ( )
float	*to_float* ( )
int	*to_int* ( )
String	*to_lower* ( )
String	*to_upper* ( )
RawArray	*to_utf8* ( )
String	*xml_escape* ( )
String	*xml_unescape* ( )

9.301.3 Description

This is the built-in string class (and the one used by GDScript). It supports Unicode and provides all necessary means for string handling. Strings are reference counted and use a copy-on-write approach, so passing them around is cheap in resources.

9.301.4 Member Function Description

- *String* **basename** ( )

If the string is a path to a file, return the path to the file without the extension.

- *bool* **begins_with** (*String* text)

Return true if the strings begins with the given string.

- *String* **c_escape ( )**
- *String* **c_unescape ( )**
- *String* **capitalize ( )**

Return the string in uppercase.

- *int* **casecmp_to (** *String* to **)**

Perform a case-sensitive comparison to another string, return -1 if less, 0 if equal and +1 if greater.

- *bool* **empty ( )**

Return true if the string is empty.

- *String* **extension ( )**

If the string is a path to a file, return the extension.

- *int* **find (** *String* what, *int* from=0 **)**

Find the first occurrence of a substring, return the starting position of the substring or -1 if not found. Optionally, the initial search index can be passed.

- *int* **find_last (** *String* what **)**

Find the last occurrence of a substring, return the starting position of the substring or -1 if not found. Optionally, the initial search index can be passed.

- *int* **findn (** *String* what, *int* from=0 **)**

Find the first occurrence of a substring but search as case-insensitive, return the starting position of the substring or -1 if not found. Optionally, the initial search index can be passed.

- *String* **get_base_dir ( )**

If the string is a path to a file, return the base directory.

- *String* **get_file ( )**

If the string is a path to a file, return the file and ignore the base directory.

- *int* **hash ( )**

Hash the string and return a 32 bits integer.

- *int* **hex_to_int ( )**

Convert a string containing an hexadecimal number into an int.

- *String* **insert (** *int* pos, *String* what **)**

Insert a substring at a given position.

- *bool* **is_abs_path ( )**

If the string is a path to a file or directory, return true if the path is absolute.

- *bool* **is_rel_path ( )**

If the string is a path to a file or directory, return true if the path is relative.

- *bool* **is_valid_float ( )**

Check whether the string contains a valid float.

- *bool* **is_valid_html_color ( )**

Check whether the string contains a valid color in HTML notation.

- *bool* **is_valid_identifier** **(** **)**
- *bool* **is_valid_integer** **(** **)**

Check whether the string contains a valid integer.

- *bool* **is_valid_ip_address** **(** **)**

Check whether the string contains a valid IP address.

- *String* **json_escape** **(** **)**
- *String* **left** **(** *int* pos **)**

Return an amount of characters from the left of the string.

- *int* **length** **(** **)**

Return the length of the string in characters.

- *bool* **match** **(** *String* expr **)**

Do a simple expression match, where '*' matches zero or more arbitrary characters and '?' matches any single character except '.'.

- *bool* **matchn** **(** *String* expr **)**

Do a simple case insensitive expression match, using ? and * wildcards (see *match*).

- *RawArray* **md5_buffer** **(** **)**
- *String* **md5_text** **(** **)**
- *int* **nocasecmp_to** **(** *String* to **)**

Perform a case-insensitive comparison to another string, return -1 if less, 0 if equal and +1 if greater.

- *String* **ord_at** **(** *int* at **)**

Return the character code at position "at".

- *String* **pad_decimals** **(** *int* digits **)**
- *String* **pad_zeros** **(** *int* digits **)**
- *String* **percent_decode** **(** **)**
- *String* **percent_encode** **(** **)**
- *String* **plus_file** **(** *String* file **)**
- *String* **replace** **(** *String* what, *String* forwhat **)**

Replace occurrences of a substring for different ones inside the string.

- *String* **replacen** **(** *String* what, *String* forwhat **)**

Replace occurrences of a substring for different ones inside the string, but search case-insensitive.

- *int* **rfind** **(** *String* what, *int* from=-1 **)**

Perform a search for a substring, but start from the end of the string instead of the beginning.

- *int* **rfindn** **(** *String* what, *int* from=-1 **)**

Perform a search for a substring, but start from the end of the string instead of the beginning. Also search case-insensitive.

- *String* **right** **(** *int* pos **)**

Return the right side of the string from a given position.

- *StringArray* **split** (*String* divisor, *bool* allow_empty=True)

Split the string by a divisor string, return an array of the substrings. Example "One,Two,Three" will return :ref:*"One","Two","Three"*<class_"one","two","three"> if split by ",".

- *RealArray* **split_floats** (*String* divisor, *bool* allow_empty=True)

Split the string in floats by using a divisor string, return an array of the substrings. Example "1,2.5,3" will return :ref:*1,2.5,3*<class_1,2.5,3> if split by ",".

- *String* **strip_edges** (*bool* left=True, *bool* right=True)

Return a copy of the string stripped of any non-printable character at the beginning and the end. The optional arguments are used to toggle stripping on the left and right edges respectively.

- *String* **substr** (*int* from, *int* len)

Return part of the string from "from", with length "len".

- *RawArray* **to_ascii** ( )

Convert the String (which is a character array) to RawArray (which is an array of bytes). The conversion is speeded up in comparison to to_utf8() with the assumption that all the characters the String contains are only ASCII characters.

- *float* **to_float** ( )

Convert a string, containing a decimal number, into a float.

- *int* **to_int** ( )

Convert a string, containing an integer number, into an int.

- *String* **to_lower** ( )

Return the string converted to lowercase.

- *String* **to_upper** ( )

Return the string converted to uppercase.

- *RawArray* **to_utf8** ( )

Convert the String (which is an array of characters) to RawArray (which is an array of bytes). The conversion is a bit slower than to_ascii(), but supports all UTF-8 characters. Therefore, you should prefer this function over to_ascii().

- *String* **xml_escape** ( )

Perform XML escaping on the string.

- *String* **xml_unescape** ( )

Perform XML un-escaping of the string.

9.302 StringArray

Category: Built-In Types

9.302.1 Brief Description

String Array.

9.302.2 Member Functions

void	*push_back* (*String* string)
void	*resize* (*int* idx)
void	*set* (*int* idx, *String* string)
int	*size* ( )
StringArray	*StringArray* (*Array* from)

9.302.3 Description

String Array. Array of strings. Can only contain strings. Optimized for memory usage, can't fragment the memory.

9.302.4 Member Function Description

- void **push_back** (*String* string)

Append a string element at end of the array.

- void **resize** (*int* idx)

Reset the size of the array.

- void **set** (*int* idx, *String* string)
- *int* **size** ( )

Return the size of the array.

- *StringArray* **StringArray** (*Array* from)

9.303 StyleBox

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *StyleBoxImageMask*, *StyleBoxFlat*, *StyleBoxTexture*, *StyleBoxEmpty*

Category: Core

9.303.1 Brief Description

Base class for drawing stylized boxes for the UI.

9.303.2 Member Functions

bool	*test_mask* (*Vector2* point, *Rect2* rect) const
void	*set_default_margin* (*int* margin, *float* offset)
float	*get_default_margin* (*int* margin) const
float	*get_margin* (*int* margin) const
Vector2	*get_minimum_size* ( ) const
Vector2	*get_center_size* ( ) const
Vector2	*get_offset* ( ) const
void	*draw* (*RID* canvas_item, *Rect2* rect) const

9.303.3 Description

StyleBox is *Resource* that provides an abstract base class for drawing stylized boxes for the UI. StyleBoxes are used for drawing the styles of buttons, line edit backgrounds, tree backgrounds, etc. and also for testing a transparency mask for pointer signals. If mask test fails on a StyleBox assigned as mask to a control, clicks and motion signals will go through it to the one below.

9.303.4 Member Function Description

- *bool* **test_mask** (*Vector2* point, *Rect2* rect) const

Test a position in a rectangle, return whether it passes the mask test.

- void **set_default_margin** (*int* margin, *float* offset)

Set the default offset "offset" of the margin "margin" (see MARGIN_* enum) for a StyleBox, Controls that draw styleboxes with context inside need to know the margin, so the border of the stylebox is not occluded.

- *float* **get_default_margin** (*int* margin) const

Return the default offset of the margin "margin" (see MARGIN_* enum) of a StyleBox, Controls that draw styleboxes with context inside need to know the margin, so the border of the stylebox is not occluded.

- *float* **get_margin** (*int* margin) const

Return the offset of margin "margin" (see MARGIN_* enum).

- *Vector2* **get_minimum_size** ( ) const

Return the minimum size that this stylebox can be shrunk to.

- *Vector2* **get_center_size** ( ) const
- *Vector2* **get_offset** ( ) const

Return the "offset" of a stylebox, this is a helper function, like writing `Vector2(style.get_margin(MARGIN_LEFT), style.get_margin(MARGIN_TOP))`.

- void **draw** (*RID* canvas_item, *Rect2* rect) const

9.304 StyleBoxEmpty

Inherits: *StyleBox* < *Resource* < *Reference* < *Object*

Category: Core

9.304.1 Brief Description

Empty stylebox (does not display anything).

9.304.2 Description

Empty stylebox (really does not display anything).

9.305 StyleBoxFlat

Inherits: *StyleBox* < *Resource* < *Reference* < *Object*

Category: Core

9.305.1 Brief Description

Stylebox of a single color.

9.305.2 Member Functions

void	*set_bg_color* (*Color* color)
Color	*get_bg_color* ( ) const
void	*set_light_color* (*Color* color)
Color	*get_light_color* ( ) const
void	*set_dark_color* (*Color* color)
Color	*get_dark_color* ( ) const
void	*set_border_size* (*int* size)
int	*get_border_size* ( ) const
void	*set_border_blend* (*bool* blend)
bool	*get_border_blend* ( ) const
void	*set_draw_center* (*bool* size)
bool	*get_draw_center* ( ) const

9.305.3 Description

Stylebox of a single color. Displays the stylebox of a single color, alternatively a border with light/dark colors can be assigned.

9.305.4 Member Function Description

- void **set_bg_color** (*Color* color)
- *Color* **get_bg_color** ( ) const
- void **set_light_color** (*Color* color)
- *Color* **get_light_color** ( ) const
- void **set_dark_color** (*Color* color)
- *Color* **get_dark_color** ( ) const
- void **set_border_size** (*int* size)
- *int* **get_border_size** ( ) const
- void **set_border_blend** (*bool* blend)
- *bool* **get_border_blend** ( ) const
- void **set_draw_center** (*bool* size)
- *bool* **get_draw_center** ( ) const

9.306 StyleBoxImageMask

Inherits: *StyleBox* < *Resource* < *Reference* < *Object*

Category: Core

9.306.1 Brief Description

Image mask based StyleBox, for mask test.

9.306.2 Member Functions

void	*set_image* (*Image* image)
Image	*get_image* ( ) const
void	*set_expand* (*bool* expand)
bool	*get_expand* ( ) const
void	*set_expand_margin_size* (*int* margin, *float* size)
float	*get_expand_margin_size* (*int* margin) const

9.306.3 Description

This StyleBox is similar to *StyleBoxTexture*, but only meant to be used for mask testing. It takes an image and applies stretch rules to determine if the point clicked is masked or not.

9.306.4 Member Function Description

- void **set_image** (*Image* image)

Set the image used for mask testing. Pixels (converted to grey) that have a value, less than 0.5 will fail the test.

- *Image* **get_image** ( ) const

Return the image used for mask testing. (see *set_image*).

- void **set_expand** (*bool* expand)

Set the expand property (default). When expanding, the image will use the same rules as *StyleBoxTexture* for expand. If not expanding, the image will always be tested at its original size.

- *bool* **get_expand** ( ) const

Return whether the expand property is set(default). When expanding, the image will use the same rules as *StyleBoxTexture* for expand. If not expanding, the image will always be tested at its original size.

- void **set_expand_margin_size** (*int* margin, *float* size)

Set an expand margin size (from enum MARGIN_*). Parts of the image below the size of the margin (and in the direction of the margin) will not expand.

- *float* **get_expand_margin_size** (*int* margin) const

Return the expand margin size (from enum MARGIN_*). Parts of the image below the size of the margin (and in the direction of the margin) will not expand.

9.307 StyleBoxTexture

Inherits: *StyleBox* < *Resource* < *Reference* < *Object*

Category: Core

9.307.1 Brief Description

Texture Based 3x3 scale style.

9.307.2 Member Functions

void	*set_texture* (*Texture* texture)
Texture	*get_texture* ( ) const
void	*set_margin_size* (*int* margin, *float* size)
float	*get_margin_size* (*int* margin) const
void	*set_expand_margin_size* (*int* margin, *float* size)
float	*get_expand_margin_size* (*int* margin) const
void	*set_draw_center* (*bool* enable)
bool	*get_draw_center* ( ) const

9.307.3 Description

Texture Based 3x3 scale style. This stylebox performs a 3x3 scaling of a texture, where only the center cell is fully stretched. This allows for the easy creation of bordered styles.

9.307.4 Member Function Description

- void **set_texture** (*Texture* texture)
- *Texture* **get_texture** ( ) const
- void **set_margin_size** (*int* margin, *float* size)
- *float* **get_margin_size** (*int* margin) const
- void **set_expand_margin_size** (*int* margin, *float* size)
- *float* **get_expand_margin_size** (*int* margin) const
- void **set_draw_center** (*bool* enable)
- *bool* **get_draw_center** ( ) const

9.308 SurfaceTool

Inherits: *Reference* < *Object*

Category: Core

9.308.1 Brief Description

Helper tool to create geometry.

9.308.2 Member Functions

void	*begin* (*int* primitive)
void	*add_vertex* (*Vector3* vertex)
void	*add_color* (*Color* color)
void	*add_normal* (*Vector3* normal)
void	*add_tangent* (*Plane* tangent)
void	*add_uv* (*Vector2* uv)
void	*add_uv2* (*Vector2* uv2)
void	*add_bones* (*IntArray* bones)
void	*add_weights* (*RealArray* weights)
void	*add_smooth_group* (*bool* smooth)
void	*set_material* (*Material* material)
void	*index* ( )
void	*deindex* ( )
void	*generate_normals* ( )
Mesh	*commit* (*Mesh* existing=NULL)
void	*clear* ( )

9.308.3 Description

Helper tool to create geometry.

9.308.4 Member Function Description

- void **begin** (*int* primitive)
- void **add_vertex** (*Vector3* vertex)
- void **add_color** (*Color* color)
- void **add_normal** (*Vector3* normal)
- void **add_tangent** (*Plane* tangent)
- void **add_uv** (*Vector2* uv)
- void **add_uv2** (*Vector2* uv2)
- void **add_bones** (*IntArray* bones)
- void **add_weights** (*RealArray* weights)
- void **add_smooth_group** (*bool* smooth)
- void **set_material** (*Material* material)
- void **index** ( )
- void **deindex** ( )
- void **generate_normals** ( )
- *Mesh* **commit** (*Mesh* existing=NULL)

- void **clear** ( )

9.309 TabContainer

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.309.1 Brief Description

Tabbed Container.

9.309.2 Member Functions

int	*get_tab_count* ( ) const
void	*set_current_tab* (*int* tab_idx)
int	*get_current_tab* ( ) const
Control	*get_current_tab_control* ( ) const
Control	*get_tab_control* (*int* idx) const
void	*set_tab_align* (*int* align)
int	*get_tab_align* ( ) const
void	*set_tabs_visible* (*bool* visible)
bool	*are_tabs_visible* ( ) const
void	*set_tab_title* (*int* tab_idx, *String* title)
String	*get_tab_title* (*int* tab_idx) const
void	*set_tab_icon* (*int* tab_idx, *Texture* icon)
Texture	*get_tab_icon* (*int* tab_idx) const
void	*set_popup* (*Popup* popup)
Popup	*get_popup* ( ) const

9.309.3 Signals

- **pre_popup_pressed** ( )
- **tab_changed** (*int* tab)

9.309.4 Description

Tabbed Container. Contains several children controls, but shows only one at the same time. Clicking on the top tabs allows to change the currently visible one.

Children controls of this one automatically.

9.309.5 Member Function Description

- *int* **get_tab_count** ( ) const

Return the amount of tabs.

- void **set_current_tab** (*int* tab_idx)

Bring a tab (and the Control it represents) to the front, and hide the rest.

- *int* **get_current_tab** ( ) const

Return the current tab that is being showed.

- *Control* **get_current_tab_control** ( ) const
- *Control* **get_tab_control** (*int* idx) const
- void **set_tab_align** (*int* align)

Set tab alignment, from the ALIGN_* enum. Moves tabs to the left, right or center.

- *int* **get_tab_align** ( ) const

Return tab alignment, from the ALIGN_* enum.

- void **set_tabs_visible** (*bool* visible)

Set whether the tabs should be visible or hidden.

- *bool* **are_tabs_visible** ( ) const

Return whether the tabs should be visible or hidden.

- void **set_tab_title** (*int* tab_idx, *String* title)

Set a title for the tab. Tab titles are by default the children node name, but this can be overridden.

- *String* **get_tab_title** (*int* tab_idx) const

Return the title for the tab. Tab titles are by default the children node name, but this can be overridden.

- void **set_tab_icon** (*int* tab_idx, *Texture* icon)

Set an icon for a tab.

- *Texture* **get_tab_icon** (*int* tab_idx) const
- void **set_popup** (*Popup* popup)
- *Popup* **get_popup** ( ) const

9.310 Tabs

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.310.1 Brief Description

Tabs Control.

9.310.2 Member Functions

int	*get_tab_count* ( ) const
void	*set_current_tab* (*int* tab_idx)
int	*get_current_tab* ( ) const
void	*set_tab_title* (*int* tab_idx, *String* title)
String	*get_tab_title* (*int* tab_idx) const
void	*set_tab_icon* (*int* tab_idx, *Texture* icon)
Texture	*get_tab_icon* (*int* tab_idx) const
void	*remove_tab* (*int* tab_idx)
void	*add_tab* (*String* title, *Texture* icon)
void	*set_tab_align* (*int* align)
int	*get_tab_align* ( ) const
void	*ensure_tab_visible* (*int* idx)

9.310.3 Signals

- **tab_close** (*int* tab)
- **right_button_pressed** (*int* tab)
- **tab_changed** (*int* tab)

9.310.4 Numeric Constants

- **ALIGN_LEFT = 0**
- **ALIGN_CENTER = 1**
- **ALIGN_RIGHT = 2**
- **CLOSE_BUTTON_SHOW_ACTIVE_ONLY = 1**
- **CLOSE_BUTTON_SHOW_ALWAYS = 2**
- **CLOSE_BUTTON_SHOW_NEVER = 0**

9.310.5 Description

Simple tabs control, similar to *TabContainer* but is only in charge of drawing tabs, not interact with children.

9.310.6 Member Function Description

- *int* **get_tab_count** ( ) const
- void **set_current_tab** (*int* tab_idx)
- *int* **get_current_tab** ( ) const
- void **set_tab_title** (*int* tab_idx, *String* title)
- *String* **get_tab_title** (*int* tab_idx) const
- void **set_tab_icon** (*int* tab_idx, *Texture* icon)
- *Texture* **get_tab_icon** (*int* tab_idx) const

- void **remove_tab** (*int* tab_idx)
- void **add_tab** (*String* title, *Texture* icon)
- void **set_tab_align** (*int* align)
- *int* **get_tab_align** ( ) const
- void **ensure_tab_visible** (*int* idx)

9.311 TCP_Server

Inherits: *Reference* < *Object*

Category: Core

9.311.1 Brief Description

TCP Server.

9.311.2 Member Functions

int	*listen* (*int* port, *StringArray* accepted_hosts=StringArray())
bool	*is_connection_available* ( ) const
Object	*take_connection* ( )
void	*stop* ( )

9.311.3 Description

TCP Server class. Listens to connections on a port and returns a *StreamPeerTCP* when got a connection.

9.311.4 Member Function Description

- *int* **listen** (*int* port, *StringArray* accepted_hosts=StringArray())

Listen on a port, alternatively give a white-list of accepted hosts.

- *bool* **is_connection_available** ( ) const

Return true if a connection is available for taking.

- *Object* **take_connection** ( )

If a connection is available, return a StreamPeerTCP with the connection/

- void **stop** ( )

Stop listening.

9.312 TestCube

Inherits: *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

Category: Core

9.312.1 Brief Description

9.313 TextEdit

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.313.1 Brief Description

Multiline text editing control.

9.313.2 Member Functions

void	*set_text* (*String* text)
void	*insert_text_at_cursor* (*String* text)
int	*get_line_count* ( ) const
String	*get_text* ( )
String	*get_line* (*int* line) const
void	*cursor_set_column* (*int* column, *bool* adjust_viewport=false)
void	*cursor_set_line* (*int* line, *bool* adjust_viewport=false)
int	*cursor_get_column* ( ) const
int	*cursor_get_line* ( ) const
void	*cursor_set_blink_enabled* (*bool* enable)
bool	*cursor_get_blink_enabled* ( ) const
void	*cursor_set_blink_speed* (*float* blink_speed)
float	*cursor_get_blink_speed* ( ) const
void	*set_readonly* (*bool* enable)
void	*set_wrap* (*bool* enable)
void	*set_max_chars* (*int* amount)
void	*cut* ( )
void	*copy* ( )
void	*paste* ( )
void	*select_all* ( )
void	*select* (*int* from_line, *int* from_column, *int* to_line, *int* to_column)
bool	*is_selection_active* ( ) const
int	*get_selection_from_line* ( ) const
int	*get_selection_from_column* ( ) const
int	*get_selection_to_line* ( ) const
int	*get_selection_to_column* ( ) const
String	*get_selection_text* ( ) const
String	*get_word_under_cursor* ( ) const
IntArray	*search* (*String* flags, *int* from_line, *int* from_column, *int* to_line) const
void	*undo* ( )
void	*redo* ( )
void	*clear_undo_history* ( )
void	*set_syntax_coloring* (*bool* enable)
bool	*is_syntax_coloring_enabled* ( ) const
void	*add_keyword_color* (*String* keyword, *Color* color)

Continúa en la página siguiente

Tabla 9.30 – proviene de la página anterior

void	*add_color_region* (*String* begin_key, *String* end_key, *Color* color, *bool* line_only=false)
void	*set_symbol_color* (*Color* color)
void	*set_custom_bg_color* (*Color* color)
void	*clear_colors* ( )

9.313.3 Signals

- **text_changed** ( )
- **cursor_changed** ( )
- **request_completion** ( )

9.313.4 Numeric Constants

- **SEARCH_MATCH_CASE = 1** — Match case when searching.
- **SEARCH_WHOLE_WORDS = 2** — Match whole words when searching.
- **SEARCH_BACKWARDS = 4** — Search from end to beginning.

9.313.5 Description

TextEdit is meant for editing large, multiline text. It also has facilities for editing code, such as syntax highlighting support and multiple levels of undo/redo.

9.313.6 Member Function Description

- void **set_text** (*String* text)

Set the entire text.

- void **insert_text_at_cursor** (*String* text)

Insert a given text at the cursor position.

- *int* **get_line_count** ( ) const

Return the amount of total lines in the text.

- *String* **get_text** ( )

Return the whole text.

- *String* **get_line** (*int* line) const

Return the text of a specific line.

- void **cursor_set_column** (*int* column, *bool* adjust_viewport=false)
- void **cursor_set_line** (*int* line, *bool* adjust_viewport=false)
- *int* **cursor_get_column** ( ) const

Return the column the editing cursor is at.

- *int* **cursor_get_line** ( ) const

Return the line the editing cursor is at.

- void **cursor_set_blink_enabled** (*bool* enable)

Set the text editor caret to blink.

- *bool* **cursor_get_blink_enabled** ( ) const

Gets whether the text editor caret is blinking.

- void **cursor_set_blink_speed** (*float* blink_speed)

Set the text editor caret blink speed. Cannot be less then or equal to 0.

- *float* **cursor_get_blink_speed** ( ) const

Gets the text editor caret blink speed.

- void **set_readonly** (*bool* enable)

Set the text editor as read-only. Text can be displayed but not edited.

- void **set_wrap** (*bool* enable)

Enable text wrapping when it goes beyond he edge of what is visible.

- void **set_max_chars** (*int* amount)

Set the maximum amount of characters editable.

- void **cut** ( )

Cut the current selection.

- void **copy** ( )

Copy the current selection.

- void **paste** ( )

Paste the current selection.

- void **select_all** ( )

Select all the text.

- void **select** (*int* from_line, *int* from_column, *int* to_line, *int* to_column)

Perform selection, from line/column to line/column.

- *bool* **is_selection_active** ( ) const

Return true if the selection is active.

- *int* **get_selection_from_line** ( ) const

Return the selection begin line.

- *int* **get_selection_from_column** ( ) const

Return the selection begin column.

- *int* **get_selection_to_line** ( ) const

Return the selection end line.

- *int* **get_selection_to_column** ( ) const

Return the selection end column.

- *String* **get_selection_text** ( ) const

Return the text inside the selection.

- *String* **get_word_under_cursor** ( ) const
- *IntArray* **search** (*String* flags, *int* from_line, *int* from_column, *int* to_line) const

Perform a search inside the text. Search flags can be specified in the SEARCH_* enum.

- void **undo** ( )

Perform undo operation.

- void **redo** ( )

Perform redo operation.

- void **clear_undo_history** ( )

Clear the undo history.

- void **set_syntax_coloring** (*bool* enable)

Set to enable the syntax coloring.

- *bool* **is_syntax_coloring_enabled** ( ) const

Return true if the syntax coloring is enabled.

- void **add_keyword_color** (*String* keyword, *Color* color)

Add a keyword and its color.

- void **add_color_region** (*String* begin_key, *String* end_key, *Color* color, *bool* line_only=false)

Add color region (given the delimiters) and its colors.

- void **set_symbol_color** (*Color* color)

Set the color for symbols.

- void **set_custom_bg_color** (*Color* color)

Set a custom background color. A background color with alpha==0 disables this.

- void **clear_colors** ( )

Clear all the syntax coloring information.

9.314 Texture

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *RenderTargetTexture*, *AtlasTexture*, *ImageTexture*, *LargeTexture*

Category: Core

9.314.1 Brief Description

Texture for 2D and 3D.

9.314.2 Member Functions

int	*get_width* ( ) const
int	*get_height* ( ) const
Vector2	*get_size* ( ) const
RID	*get_rid* ( ) const
bool	*has_alpha* ( ) const
void	*set_flags* (*int* flags)
int	*get_flags* ( ) const
void	*draw* (*RID* canvas_item, *Vector2* pos, *Color* modulate=Color(1,1,1,1), *bool* transpose=false) const
void	*draw_rect* (*RID* canvas_item, *Rect2* rect, *bool* tile, *Color* modulate=Color(1,1,1,1), *bool* transpose=false) const
void	*draw_rect_region* (*RID* canvas_item, *Rect2* rect, *Rect2* src_rect, *Color* modulate=Color(1,1,1,1), *bool* transpose=false) const

9.314.3 Numeric Constants

- **FLAG_MIPMAPS = 1** — Generate mipmaps, to enable smooth zooming out of the texture.
- **FLAG_REPEAT = 2** — Repeat (instead of clamp to edge).
- **FLAG_FILTER = 4** — Turn on magnifying filter, to enable smooth zooming in of the texture.
- **FLAG_VIDEO_SURFACE = 4096** — Texture is a video surface.
- **FLAGS_DEFAULT = 7** — Default flags. Generate mipmaps, repeat, and filter are enabled.
- **FLAG_ANISOTROPIC_FILTER = 8**
- **FLAG_CONVERT_TO_LINEAR = 16**
- **FLAG_MIRRORED_REPEAT = 32**

9.314.4 Description

A texture works by registering an image in the video hardware, which then can be used in 3D models or 2D *Sprite* or GUI *Control*.

9.314.5 Member Function Description

- *int* **get_width** ( ) const

Return the texture width.

- *int* **get_height** ( ) const

Return the texture height.

- *Vector2* **get_size** ( ) const

Return the texture size.

- *RID* **get_rid** ( ) const

Return the texture RID as used in the *VisualServer*.

- *bool* **has_alpha** ( ) const

- void **set_flags** (*int* flags)

Change the texture flags.

- *int* **get_flags** ( ) const

Return the current texture flags.

- void **draw** (*RID* canvas_item, *Vector2* pos, *Color* modulate=Color(1,1,1,1), *bool* transpose=false) const
- void **draw_rect** (*RID* canvas_item, *Rect2* rect, *bool* tile, *Color* modulate=Color(1,1,1,1), *bool* transpose=false) const
- void **draw_rect_region** (*RID* canvas_item, *Rect2* rect, *Rect2* src_rect, *Color* modulate=Color(1,1,1,1), *bool* transpose=false) const

9.315 TextureButton

Inherits: *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.315.1 Brief Description

Button that can be themed with textures.

9.315.2 Member Functions

void	*set_normal_texture* (*Texture* texture)
void	*set_pressed_texture* (*Texture* texture)
void	*set_hover_texture* (*Texture* texture)
void	*set_disabled_texture* (*Texture* texture)
void	*set_focused_texture* (*Texture* texture)
void	*set_click_mask* (*BitMap* mask)
void	*set_texture_scale* (*Vector2* scale)
void	*set_modulate* (*Color* color)
Texture	*get_normal_texture* ( ) const
Texture	*get_pressed_texture* ( ) const
Texture	*get_hover_texture* ( ) const
Texture	*get_disabled_texture* ( ) const
Texture	*get_focused_texture* ( ) const
BitMap	*get_click_mask* ( ) const
Vector2	*get_texture_scale* ( ) const
Color	*get_modulate* ( ) const

9.315.3 Description

Button that can be themed with textures. This is like a regular *Button* but can be themed by assigning textures to it. This button is intended to be easy to theme, however a regular button can expand (that uses styleboxes) and still be better if the interface is expect to have internationalization of texts.

Only the normal texture is required, the others are optional.

9.315.4 Member Function Description

- void **set_normal_texture** (*Texture* texture)
- void **set_pressed_texture** (*Texture* texture)
- void **set_hover_texture** (*Texture* texture)
- void **set_disabled_texture** (*Texture* texture)
- void **set_focused_texture** (*Texture* texture)
- void **set_click_mask** (*BitMap* mask)
- void **set_texture_scale** (*Vector2* scale)
- void **set_modulate** (*Color* color)
- *Texture* **get_normal_texture** ( ) const
- *Texture* **get_pressed_texture** ( ) const
- *Texture* **get_hover_texture** ( ) const
- *Texture* **get_disabled_texture** ( ) const
- *Texture* **get_focused_texture** ( ) const
- *BitMap* **get_click_mask** ( ) const
- *Vector2* **get_texture_scale** ( ) const
- *Color* **get_modulate** ( ) const

9.316 TextureFrame

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.316.1 Brief Description

Control Frame that draws a texture.

9.316.2 Member Functions

void	*set_texture* (*Object* texture)
Object	*get_texture* ( ) const
void	*set_modulate* (*Color* modulate)
Color	*get_modulate* ( ) const
void	*set_expand* (*bool* enable)
bool	*has_expand* ( ) const

9.316.3 Description

Control frame that simply draws an assigned texture. It can stretch or not. It's a simple way to just show an image in a UI.

9.316.4 Member Function Description

- void **set_texture** (*Object* texture)
- *Object* **get_texture** ( ) const
- void **set_modulate** (*Color* modulate)
- *Color* **get_modulate** ( ) const
- void **set_expand** (*bool* enable)
- *bool* **has_expand** ( ) const

9.317 TextureProgress

Inherits: *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.317.1 Brief Description

Textured progress bar implementation.

9.317.2 Member Functions

void	*set_under_texture* (*Object* tex)
Object	*get_under_texture* ( ) const
void	*set_progress_texture* (*Object* tex)
Object	*get_progress_texture* ( ) const
void	*set_over_texture* (*Object* tex)
Object	*get_over_texture* ( ) const
void	*set_fill_mode* (*int* mode)
int	*get_fill_mode* ( )
void	*set_radial_initial_angle* (*float* mode)
float	*get_radial_initial_angle* ( )
void	*set_radial_center_offset* (*Vector2* mode)
Vector2	*get_radial_center_offset* ( )
void	*set_fill_degrees* (*float* mode)
float	*get_fill_degrees* ( )

9.317.3 Numeric Constants

- **FILL_LEFT_TO_RIGHT = 0**
- **FILL_RIGHT_TO_LEFT = 1**
- **FILL_TOP_TO_BOTTOM = 2**
- **FILL_BOTTOM_TO_TOP = 3**
- **FILL_CLOCKWISE = 4**
- **FILL_COUNTER_CLOCKWISE = 5**

9.317.4 Description

ProgressBar implementation that is easier to theme (by just passing a few textures).

9.317.5 Member Function Description

- void **set_under_texture** (*Object* tex)
- *Object* **get_under_texture** ( ) const
- void **set_progress_texture** (*Object* tex)
- *Object* **get_progress_texture** ( ) const
- void **set_over_texture** (*Object* tex)
- *Object* **get_over_texture** ( ) const
- void **set_fill_mode** (*int* mode)
- *int* **get_fill_mode** ( )
- void **set_radial_initial_angle** (*float* mode)
- *float* **get_radial_initial_angle** ( )
- void **set_radial_center_offset** (*Vector2* mode)
- *Vector2* **get_radial_center_offset** ( )
- void **set_fill_degrees** (*float* mode)
- *float* **get_fill_degrees** ( )

9.318 Theme

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.318.1 Brief Description

Theme for controls.

9.318.2 Member Functions

void	*set_icon* (*String* name, *String* type, *Texture* texture)
Texture	*get_icon* (*String* name, *String* type) const
bool	*has_icon* (*String* name, *String* type) const
void	*clear_icon* (*String* name, *String* type)
StringArray	*get_icon_list* (*String* type) const
void	*set_stylebox* (*String* name, *String* type, *StyleBox* texture)
StyleBox	*get_stylebox* (*String* name, *String* type) const
bool	*has_stylebox* (*String* name, *String* type) const
void	*clear_stylebox* (*String* name, *String* type)
StringArray	*get_stylebox_list* (*String* type) const
void	*set_font* (*String* name, *String* type, *Font* font)
Font	*get_font* (*String* name, *String* type) const
bool	*has_font* (*String* name, *String* type) const
void	*clear_font* (*String* name, *String* type)
StringArray	*get_font_list* (*String* type) const
void	*set_color* (*String* name, *String* type, *Color* color)
Color	*get_color* (*String* name, *String* type) const
bool	*has_color* (*String* name, *String* type) const
void	*clear_color* (*String* name, *String* type)
StringArray	*get_color_list* (*String* type) const
void	*set_constant* (*String* name, *String* type, *int* constant)
int	*get_constant* (*String* name, *String* type) const
bool	*has_constant* (*String* name, *String* type) const
void	*clear_constant* (*String* name, *String* type)
StringArray	*get_constant_list* (*String* type) const
void	*set_default_font* (*Object* font)
Object	*get_default_font* ( ) const
StringArray	*get_type_list* (*String* type) const
void	*copy_default_theme* ( )

9.318.3 Description

Theme for skinning controls. Controls can be skinned individually, but for complex applications it's more efficient to just create a global theme that defines everything. This theme can be applied to any *Control*, and it and its children will automatically use it.

Theme resources can be alternatively loaded by writing them in a .theme file, see wiki for more info.

9.318.4 Member Function Description

- void **set_icon** (*String* name, *String* type, *Texture* texture)
- *Texture* **get_icon** (*String* name, *String* type) const
- *bool* **has_icon** (*String* name, *String* type) const
- void **clear_icon** (*String* name, *String* type)
- *StringArray* **get_icon_list** (*String* type) const
- void **set_stylebox** (*String* name, *String* type, *StyleBox* texture)

- *StyleBox* **get_stylebox** (*String* name, *String* type) const
- *bool* **has_stylebox** (*String* name, *String* type) const
- void **clear_stylebox** (*String* name, *String* type)
- *StringArray* **get_stylebox_list** (*String* type) const
- void **set_font** (*String* name, *String* type, *Font* font)
- *Font* **get_font** (*String* name, *String* type) const
- *bool* **has_font** (*String* name, *String* type) const
- void **clear_font** (*String* name, *String* type)
- *StringArray* **get_font_list** (*String* type) const
- void **set_color** (*String* name, *String* type, *Color* color)
- *Color* **get_color** (*String* name, *String* type) const
- *bool* **has_color** (*String* name, *String* type) const
- void **clear_color** (*String* name, *String* type)
- *StringArray* **get_color_list** (*String* type) const
- void **set_constant** (*String* name, *String* type, *int* constant)
- *int* **get_constant** (*String* name, *String* type) const
- *bool* **has_constant** (*String* name, *String* type) const
- void **clear_constant** (*String* name, *String* type)
- *StringArray* **get_constant_list** (*String* type) const
- void **set_default_font** (*Object* font)
- *Object* **get_default_font** ( ) const
- *StringArray* **get_type_list** (*String* type) const
- void **copy_default_theme** ( )

9.319 Thread

Inherits: *Reference* < *Object*

Category: Core

9.319.1 Brief Description

9.319.2 Member Functions

Error	*start* (*Object* instance, *String* method, var userdata=NULL, *int* priority=1)
String	*get_id* ( ) const
bool	*is_active* ( ) const
Variant	*wait_to_finish* ( )

9.319.3 Numeric Constants

- **PRIORITY_LOW = 0**
- **PRIORITY_NORMAL = 1**
- **PRIORITY_HIGH = 2**

9.319.4 Member Function Description

- Error **start** (*Object* instance, *String* method, var userdata=NULL, *int* priority=1)
- *String* **get_id** ( ) const
- *bool* **is_active** ( ) const
- Variant **wait_to_finish** ( )

9.320 TileMap

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.320.1 Brief Description

Node for 2D tile-based games.

9.320.2 Member Functions

void	*set_tileset* (*TileSet* tileset)
TileSet	*get_tileset* ( ) const
void	*set_mode* (*int* mode)
int	*get_mode* ( ) const
void	*set_half_offset* (*int* half_offset)
int	*get_half_offset* ( ) const
void	*set_custom_transform* (*Matrix32* custom_transform)
Matrix32	*get_custom_transform* ( ) const
void	*set_cell_size* (*Vector2* size)
Vector2	*get_cell_size* ( ) const
void	*set_quadrant_size* (*int* size)
int	*get_quadrant_size* ( ) const
void	*set_tile_origin* (*int* origin)
int	*get_tile_origin* ( ) const
void	*set_center_x* (*bool* enable)
bool	*get_center_x* ( ) const
void	*set_center_y* (*bool* enable)
bool	*get_center_y* ( ) const
void	*set_y_sort_mode* (*bool* enable)
bool	*is_y_sort_mode_enabled* ( ) const

Continúa en la página siguiente

Tabla 9.31 – proviene de la página anterior

void	*set_collision_use_kinematic* (*bool* use_kinematic)
bool	*get_collision_use_kinematic* ( ) const
void	*set_collision_layer* (*int* mask)
int	*get_collision_layer* ( ) const
void	*set_collision_mask* (*int* mask)
int	*get_collision_mask* ( ) const
void	*set_collision_friction* (*float* value)
float	*get_collision_friction* ( ) const
void	*set_collision_bounce* (*float* value)
float	*get_collision_bounce* ( ) const
void	*set_occluder_light_mask* (*int* mask)
int	*get_occluder_light_mask* ( ) const
void	*set_cell* (*int* x, *int* y, *int* tile, *bool* flip_x=false, *bool* flip_y=false, *bool* transpose=false)
void	*set_cellv* (*Vector2* pos, *int* tile, *bool* flip_x=false, *bool* flip_y=false, *bool* transpose=false)
int	*get_cell* (*int* x, *int* y) const
int	*get_cellv* (*Vector2* pos) const
bool	*is_cell_x_flipped* (*int* x, *int* y) const
bool	*is_cell_y_flipped* (*int* x, *int* y) const
bool	*is_cell_transposed* (*int* x, *int* y) const
void	*clear* ( )
Array	*get_used_cells* ( ) const
Vector2	*map_to_world* (*Vector2* mappos, *bool* ignore_half_ofs=false) const
Vector2	*world_to_map* (*Vector2* worldpos) const

9.320.3 Signals

- **settings_changed** ( )

9.320.4 Numeric Constants

- **INVALID_CELL = -1** — Returned when a cell doesn't exist.
- **MODE_SQUARE = 0** — Orthogonal orientation mode.
- **MODE_ISOMETRIC = 1** — Isometric orientation mode.
- **MODE_CUSTOM = 2** — Custom orientation mode.
- **HALF_OFFSET_X = 0** — Half offset on the X coordinate.
- **HALF_OFFSET_Y = 1** — Half offset on the Y coordinate.
- **HALF_OFFSET_DISABLED = 2** — Half offset disabled.
- **TILE_ORIGIN_TOP_LEFT = 0** — Tile origin at its top-left corner.
- **TILE_ORIGIN_CENTER = 1** — Tile origin at its center.

9.320.5 Description

Node for 2D tile-based games. Tilemaps use a *TileSet* which contain a list of tiles (textures, their rect and a collision) and are used to create complex grid-based maps.

To optimize drawing and culling (sort of like *GridMap*), you can specify a quadrant size, so chunks of the map will be batched together at drawing time.

9.320.6 Member Function Description

- void **set_tileset** (*TileSet* tileset)

Set the current tileset.

- *TileSet* **get_tileset** ( ) const

Return the current tileset.

- void **set_mode** (*int* mode)

Set the orientation mode as square, isometric or custom (use MODE_* constants as argument).

- *int* **get_mode** ( ) const

Return the orientation mode.

- void **set_half_offset** (*int* half_offset)

Set an half offset on the X coordinate, Y coordinate, or none (use HALF_OFFSET_* constants as argument).

Half offset sets every other tile off by a half tile size in the specified direction.

- *int* **get_half_offset** ( ) const

Return the current half offset configuration.

- void **set_custom_transform** (*Matrix32* custom_transform)

Set custom transform matrix, to use in combination with the custom orientation mode.

- *Matrix32* **get_custom_transform** ( ) const

Return the custom transform matrix.

- void **set_cell_size** (*Vector2* size)

Set the cell size.

- *Vector2* **get_cell_size** ( ) const

Return the cell size.

- void **set_quadrant_size** (*int* size)

Set the quadrant size, this optimizes drawing by batching chunks of map at draw/cull time.

Allowed values are integers ranging from 1 to 128.

- *int* **get_quadrant_size** ( ) const

Return the quadrant size.

- void **set_tile_origin** (*int* origin)

Set the tile origin to the tile center or its top-left corner (use TILE_ORIGIN_* constants as argument).

- *int* **get_tile_origin** ( ) const

Return the tile origin configuration.

- void **set_center_x** (*bool* enable)

Set tiles to be centered in x coordinate. (by default this is false and they are drawn from upper left cell corner).

- *bool* **get_center_x** **(** **)** const

Return true if tiles are to be centered in x coordinate (by default this is false and they are drawn from upper left cell corner).

- void **set_center_y** **(** *bool* enable **)**

Set tiles to be centered in y coordinate. (by default this is false and they are drawn from upper left cell corner).

- *bool* **get_center_y** **(** **)** const

Return true if tiles are to be centered in y coordinate (by default this is false and they are drawn from upper left cell corner).

- void **set_y_sort_mode** **(** *bool* enable **)**

Set the Y sort mode. Enabled Y sort mode means that children of the tilemap will be drawn in the order defined by their Y coordinate.

A tile with a higher Y coordinate will therefore be drawn later, potentially covering up the tile(s) above it if its sprite is higher than its cell size.

- *bool* **is_y_sort_mode_enabled** **(** **)** const

Return the Y sort mode.

- void **set_collision_use_kinematic** **(** *bool* use_kinematic **)**

Set the tilemap to handle collisions as a kinematic body (enabled) or a static body (disabled).

- *bool* **get_collision_use_kinematic** **(** **)** const

Return whether the tilemap handles collisions as a kinematic body.

- void **set_collision_layer** **(** *int* mask **)**

Set the collision layer.

Layers are referenced by binary indexes, so allowable values to describe the 20 available layers range from 0 to 2^20-1.

- *int* **get_collision_layer** **(** **)** const

Return the collision layer.

- void **set_collision_mask** **(** *int* mask **)**

Set the collision masks.

Masks are referenced by binary indexes, so allowable values to describe the 20 available masks range from 0 to 2^20-1.

- *int* **get_collision_mask** **(** **)** const

Return the collision mask.

- void **set_collision_friction** **(** *float* value **)**

Set the collision friction parameter.

Allowable values range from 0 to 1.

- *float* **get_collision_friction** **(** **)** const

Return the collision friction parameter.

- void **set_collision_bounce** **(** *float* value **)**

Set the collision bounce parameter.

Allowable values range from 0 to 1.

- *float* **get_collision_bounce ( )** const

Return the collision bounce parameter.

- void **set_occluder_light_mask (*int* mask)**
- *int* **get_occluder_light_mask ( )** const
- void **set_cell (*int* x, *int* y, *int* tile, *bool* flip_x=false, *bool* flip_y=false, *bool* transpose=false)**

Set the tile index for the cell referenced by its grid-based X and Y coordinates.

A tile index of -1 clears the cell.

Optionally, the tile can also be flipped over the X and Y coordinates or transposed.

- void **set_cellv (*Vector2* pos, *int* tile, *bool* flip_x=false, *bool* flip_y=false, *bool* transpose=false)**

Set the tile index for the cell referenced by a Vector2 of grid-based coordinates.

A tile index of -1 clears the cell.

Optionally, the tile can also be flipped over the X and Y axes or transposed.

- *int* **get_cell (*int* x, *int* y)** const

Return the tile index of the referenced cell.

- *int* **get_cellv (*Vector2* pos)** const

Return the tile index of the cell referenced by a Vector2.

- *bool* **is_cell_x_flipped (*int* x, *int* y)** const

Return whether the referenced cell is flipped over the X axis.

- *bool* **is_cell_y_flipped (*int* x, *int* y)** const

Return whether the referenced cell is flipped over the Y axis.

- *bool* **is_cell_transposed (*int* x, *int* y)** const

Return whether the referenced cell is transposed, i.e. the X and Y axes are swapped (mirroring with regard to the (1,1) vector).

- void **clear ( )**

Clear all cells.

- *Array* **get_used_cells ( )** const

Return an array of all cells containing a tile from the tileset (i.e. a tile index different from -1).

- *Vector2* **map_to_world (*Vector2* mappos, *bool* ignore_half_ofs=false)** const

Return the absolute world position corresponding to the tilemap (grid-based) coordinates given as an argument.

Optionally, the tilemap's potential half offset can be ignored.

- *Vector2* **world_to_map (*Vector2* worldpos)** const

Return the tilemap (grid-based) coordinates corresponding to the absolute world position given as an argument.

9.321 TileSet

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.321.1 Brief Description

Tile library for tilemaps.

9.321.2 Member Functions

void	*create_tile* (*int* id)
void	*tile_set_name* (*int* id, *String* name)
String	*tile_get_name* (*int* id) const
void	*tile_set_texture* (*int* id, *Texture* texture)
Texture	*tile_get_texture* (*int* id) const
void	*tile_set_material* (*int* id, *CanvasItemMaterial* material)
CanvasItemMaterial	*tile_get_material* (*int* id) const
void	*tile_set_texture_offset* (*int* id, *Vector2* texture_offset)
Vector2	*tile_get_texture_offset* (*int* id) const
void	*tile_set_shape_offset* (*int* id, *Vector2* shape_offset)
Vector2	*tile_get_shape_offset* (*int* id) const
void	*tile_set_region* (*int* id, *Rect2* region)
Rect2	*tile_get_region* (*int* id) const
void	*tile_set_shape* (*int* id, *Shape2D* shape)
Shape2D	*tile_get_shape* (*int* id) const
void	*tile_set_shapes* (*int* id, *Array* shapes)
Array	*tile_get_shapes* (*int* id) const
void	*tile_set_navigation_polygon* (*int* id, *NavigationPolygon* navigation_polygon)
NavigationPolygon	*tile_get_navigation_polygon* (*int* id) const
void	*tile_set_navigation_polygon_offset* (*int* id, *Vector2* navigation_polygon_offset)
Vector2	*tile_get_navigation_polygon_offset* (*int* id) const
void	*tile_set_light_occluder* (*int* id, *OccluderPolygon2D* light_occluder)
OccluderPolygon2D	*tile_get_light_occluder* (*int* id) const
void	*tile_set_occluder_offset* (*int* id, *Vector2* occluder_offset)
Vector2	*tile_get_occluder_offset* (*int* id) const
void	*remove_tile* (*int* id)
void	*clear* ( )
int	*get_last_unused_tile_id* ( ) const
int	*find_tile_by_name* (*String* name) const
Array	*get_tiles_ids* ( ) const

9.321.3 Description

A TileSet is a library of tiles for a *TileMap*. It contains a list of tiles, each consisting of a sprite and optional collision shapes.

Tiles are referenced by a unique integer ID.

9.321.4 Member Function Description

- void **create_tile** (*int* id)

Create a new tile which will be referenced by the given ID.

- void **tile_set_name** (*int* id, *String* name)

Set the name of the tile, for descriptive purposes.

- *String* **tile_get_name** (*int* id) const

Return the name of the tile.

- void **tile_set_texture** (*int* id, *Texture* texture)

Set the texture of the tile.

- *Texture* **tile_get_texture** (*int* id) const

Return the texture of the tile.

- void **tile_set_material** (*int* id, *CanvasItemMaterial* material)

Set the material of the tile.

- *CanvasItemMaterial* **tile_get_material** (*int* id) const

Return the material of the tile.

- void **tile_set_texture_offset** (*int* id, *Vector2* texture_offset)

Set the texture offset of the tile.

- *Vector2* **tile_get_texture_offset** (*int* id) const

Return the texture offset of the tile.

- void **tile_set_shape_offset** (*int* id, *Vector2* shape_offset)

Set the shape offset of the tile.

- *Vector2* **tile_get_shape_offset** (*int* id) const

Return the shape offset of the tile.

- void **tile_set_region** (*int* id, *Rect2* region)

Set the tile sub-region in the texture. This is common in texture atlases.

- *Rect2* **tile_get_region** (*int* id) const

Return the tile sub-region in the texture.

- void **tile_set_shape** (*int* id, *Shape2D* shape)

Set a shape for the tile, enabling physics to collide with it.

- *Shape2D* **tile_get_shape** (*int* id) const

Return the shape of the tile.

- void **tile_set_shapes** (*int* id, *Array* shapes)

Set an array of shapes for the tile, enabling physics to collide with it.

- *Array* **tile_get_shapes** (*int* id) const

Return the array of shapes of the tile.

- void **tile_set_navigation_polygon** (*int* id, *NavigationPolygon* navigation_polygon)

Set a navigation polygon for the tile.

- *NavigationPolygon* **tile_get_navigation_polygon** (*int* id) const

Return the navigation polygon of the tile.

- void **tile_set_navigation_polygon_offset** (*int* id, *Vector2* navigation_polygon_offset)

Set an offset for the tile's navigation polygon.

- *Vector2* **tile_get_navigation_polygon_offset** (*int* id) const

Return the offset of the tile's navigation polygon.

- void **tile_set_light_occluder** (*int* id, *OccluderPolygon2D* light_occluder)

Set a light occluder for the tile.

- *OccluderPolygon2D* **tile_get_light_occluder** (*int* id) const

Return the light occluder of the tile.

- void **tile_set_occluder_offset** (*int* id, *Vector2* occluder_offset)

Set an offset for the tile's light occluder.

- *Vector2* **tile_get_occluder_offset** (*int* id) const

Return the offset of the tile's light occluder.

- void **remove_tile** (*int* id)

Remove the tile referenced by the given ID.

- void **clear** ( )

Clear all tiles.

- *int* **get_last_unused_tile_id** ( ) const

Return the ID following the last currently used ID, useful when creating a new tile.

- *int* **find_tile_by_name** (*String* name) const

Find the first tile matching the given name.

- *Array* **get_tiles_ids** ( ) const

Return an array of all currently used tile IDs.

9.322 Timer

Inherits: *Node* < *Object*

Category: Core

9.322.1 Brief Description

9.322.2 Member Functions

void	*set_wait_time* (*float* time_sec)
float	*get_wait_time* ( ) const
void	*set_one_shot* (*bool* enable)
bool	*is_one_shot* ( ) const
void	*set_autostart* (*bool* enable)
bool	*has_autostart* ( ) const
void	*start* ( )
void	*stop* ( )
float	*get_time_left* ( ) const
void	*set_timer_process_mode* (*int* mode)
int	*get_timer_process_mode* ( ) const

9.322.3 Signals

- **timeout** ( )

9.322.4 Numeric Constants

- **TIMER_PROCESS_FIXED = 0** — Update the timer at fixed intervals (framerate processing).
- **TIMER_PROCESS_IDLE = 1** — Update the timer during the idle time at each frame.

9.322.5 Description

Timer node. This is a simple node that will emit a timeout callback when the timer runs out. It can optionally be set to loop.

9.322.6 Member Function Description

- void **set_wait_time** (*float* time_sec)

Set wait time in seconds. When the time is over, it will emit the timeout signal.

- *float* **get_wait_time** ( ) const

Return the wait time in seconds.

- void **set_one_shot** (*bool* enable)

Set as one-shot. If enabled, the timer will stop after timeout, otherwise it will automatically restart.

- *bool* **is_one_shot** ( ) const

Return true if configured as one-shot.

- void **set_autostart** (*bool* enable)

Set to automatically start when entering the scene.

- *bool* **has_autostart** ( ) const

Return true if set to automatically start when entering the scene.

- void **start** ( )

Start the timer.

- void **stop** ( )

Stop (cancel) the timer.

- *float* **get_time_left** ( ) const

Return the time left for timeout in seconds if the timer is active, 0 otherwise.

- void **set_timer_process_mode** (*int* mode)

Set the timer's processing mode (fixed or idle, use TIMER_PROCESS_* constants as argument).

- *int* **get_timer_process_mode** ( ) const

Return the timer's processing mode.

9.323 ToolButton

Inherits: *Button* < *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.323.1 Brief Description

9.324 TouchScreenButton

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.324.1 Brief Description

9.324.2 Member Functions

void	*set_texture* (*Object* texture)
Object	*get_texture* ( ) const
void	*set_texture_pressed* (*Object* texture_pressed)
Object	*get_texture_pressed* ( ) const
void	*set_bitmask* (*Object* bitmask)
Object	*get_bitmask* ( ) const
void	*set_action* (*String* action)
String	*get_action* ( ) const
void	*set_visibility_mode* (*int* mode)
int	*get_visibility_mode* ( ) const
void	*set_passby_press* (*bool* enabled)
bool	*is_passby_press_enabled* ( ) const
bool	*is_pressed* ( ) const

9.324.3 Signals

- **released** ( )
- **pressed** ( )

9.324.4 Member Function Description

- void **set_texture** (*Object* texture)
- *Object* **get_texture** ( ) const
- void **set_texture_pressed** (*Object* texture_pressed)
- *Object* **get_texture_pressed** ( ) const
- void **set_bitmask** (*Object* bitmask)
- *Object* **get_bitmask** ( ) const
- void **set_action** (*String* action)
- *String* **get_action** ( ) const
- void **set_visibility_mode** (*int* mode)
- *int* **get_visibility_mode** ( ) const
- void **set_passby_press** (*bool* enabled)
- *bool* **is_passby_press_enabled** ( ) const
- *bool* **is_pressed** ( ) const

9.325 Transform

Category: Built-In Types

9.325.1 Brief Description

3D Transformation.

9.325.2 Member Functions

Transform	*affine_inverse* ( )
Transform	*inverse* ( )
Transform	*looking_at* (*Vector3* target, *Vector3* up)
Transform	*orthonormalized* ( )
Transform	*rotated* (*Vector3* axis, *float* phi)
Transform	*scaled* (*Vector3* scale)
Transform	*translated* (*Vector3* ofs)
var	*xform* (var v)
var	*xform_inv* (var v)
Transform	Transform (*Vector3* x_axis, *Vector3* y_axis, *Vector3* z_axis, *Vector3* origin)
Transform	Transform (*Matrix3* basis, *Vector3* origin)
Transform	Transform (*Matrix32* from)
Transform	Transform (*Quat* from)
Transform	Transform (*Matrix3* from)

9.325.3 Member Variables

- *Matrix3* **basis** - The basis contains 3 [Vector3]. X axis, Y axis, and Z axis.
- *Vector3* **origin** - The origin of the transform. Which is the translation offset.

9.325.4 Description

Transform is used to store transformations, including translations. It consists of a Matrix3 “basis” and Vector3 “origin”. Transform is used to represent transformations of any object in space. It is similar to a 4x3 matrix.

9.325.5 Member Function Description

- *Transform* **affine_inverse** ( )

Returns the inverse of the transfrom, even if the transform has scale or the axis vectors are not orthogonal.

- *Transform* **inverse** ( )

Returns the inverse of the transform.

- *Transform* **looking_at** (*Vector3* target, *Vector3* up)

Rotate the transform around the up vector to face the target.

- *Transform* **orthonormalized** ( )

Returns a transfrom with the basis orthogonal (90 degrees), and normalized axis vectors.

- *Transform* **rotated** (*Vector3* axis, *float* phi)

Rotate the transform locally.

- *Transform* **scaled** (*Vector3* scale)

Scale the transform locally.

- *Transform* **translated** (*Vector3* ofs)

Translate the transform locally.

- var **xform** (var v)

Transforms vector "v" by this transform.

- var **xform_inv** (var v)

Inverse-transforms vector "v" by this transform.

- *Transform* **Transform** (*Vector3* x_axis, *Vector3* y_axis, *Vector3* z_axis, *Vector3* origin)

Construct the Transform from four Vector3. Each axis creates the basis.

- *Transform* **Transform** (*Matrix3* basis, *Vector3* origin)

Construct the Transform from a Matrix3 and Vector3.

- *Transform* **Transform** (*Matrix32* from)

Construct the Transform from a Matrix32.

- *Transform* **Transform** (*Quat* from)

Construct the Transform from a Quat. The origin will be Vector3(0, 0, 0)

- *Transform* **Transform** (*Matrix3* from)

Construct the Transform from a Matrix3. The origin will be Vector3(0, 0, 0)

9.326 Translation

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *PHashTranslation*

Category: Core

9.326.1 Brief Description

Language Translation.

9.326.2 Member Functions

void	*set_locale* (*String* locale)
String	*get_locale* ( ) const
void	*add_message* (*String* src_message, *String* xlated_message)
String	*get_message* (*String* src_message) const
void	*erase_message* (*String* src_message)
StringArray	*get_message_list* ( ) const
int	*get_message_count* ( ) const

9.326.3 Description

Translations are resources that can be loaded/unloaded on demand. They map a string to another string.

9.326.4 Member Function Description

- void **set_locale** (*String* locale)

Set the locale of the translation.

- *String* **get_locale** ( ) const

Return the locale of the translation.

- void **add_message** (*String* src_message, *String* xlated_message)

Add a message for translation.

- *String* **get_message** (*String* src_message) const

Return a message for translation.

- void **erase_message** (*String* src_message)

Erase a message.

- *StringArray* **get_message_list** ( ) const

Return all the messages (keys).

- *int* **get_message_count** ( ) const

9.327 TranslationServer

Inherits: *Object*

Category: Core

9.327.1 Brief Description

Server that manages all translations. Translations can be set to it and removed from it.

9.327.2 Member Functions

void	*set_locale* (*String* locale)
String	*get_locale* ( ) const
String	*translate* (*String* message) const
void	*add_translation* (*Translation* translation)
void	*remove_translation* (*Translation* translation)
void	*clear* ( )

9.327.3 Member Function Description

- void **set_locale** (*String* locale)
- *String* **get_locale** ( ) const
- *String* **translate** (*String* message) const
- void **add_translation** (*Translation* translation)
- void **remove_translation** (*Translation* translation)

- void **clear** ( )

9.328 Tree

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.328.1 Brief Description

9.328.2 Member Functions

void	*clear* ( )
TreeItem	*create_item* (*TreeItem* parent=NULL)
TreeItem	*get_root* ( )
void	*set_column_min_width* (*int* column, *int* min_width)
void	*set_column_expand* (*int* column, *bool* expand)
int	*get_column_width* (*int* column) const
void	*set_hide_root* (*bool* enable)
TreeItem	*get_next_selected* (*TreeItem* from)
TreeItem	*get_selected* ( ) const
int	*get_selected_column* ( ) const
int	*get_pressed_button* ( ) const
void	*set_select_mode* (*int* mode)
void	*set_columns* (*int* amount)
int	*get_columns* ( ) const
TreeItem	*get_edited* ( ) const
int	*get_edited_column* ( ) const
Rect2	*get_custom_popup_rect* ( ) const
Rect2	*get_item_area_rect* (*TreeItem* item, *int* column=-1) const
void	*ensure_cursor_is_visible* ( )
void	*set_column_titles_visible* (*bool* visible)
bool	*are_column_titles_visible* ( ) const
void	*set_column_title* (*int* column, *String* title)
String	*get_column_title* (*int* column) const
Vector2	*get_scroll* ( ) const
void	*set_hide_folding* (*bool* hide)
bool	*is_folding_hidden* ( ) const

9.328.3 Signals

- **item_activated** ( )
- **multi_selected** (*Object* item, *int* column, *bool* selected)
- **custom_popup_edited** (*bool* arrow_clicked)
- **item_collapsed** (*Object* item)
- **item_edited** ( )
- **item_selected** ( )

- **cell_selected** ( )
- **button_pressed** (*Object* item, *int* column, *int* id)

9.328.4 Numeric Constants

- **SELECT_SINGLE = 0**
- **SELECT_ROW = 1**
- **SELECT_MULTI = 2**

9.328.5 Member Function Description

- void **clear** ( )
- *TreeItem* **create_item** (*TreeItem* parent=NULL)
- *TreeItem* **get_root** ( )
- void **set_column_min_width** (*int* column, *int* min_width)
- void **set_column_expand** (*int* column, *bool* expand)
- *int* **get_column_width** (*int* column) const
- void **set_hide_root** (*bool* enable)
- *TreeItem* **get_next_selected** (*TreeItem* from)
- *TreeItem* **get_selected** ( ) const
- *int* **get_selected_column** ( ) const
- *int* **get_pressed_button** ( ) const
- void **set_select_mode** (*int* mode)
- void **set_columns** (*int* amount)
- *int* **get_columns** ( ) const
- *TreeItem* **get_edited** ( ) const
- *int* **get_edited_column** ( ) const
- *Rect2* **get_custom_popup_rect** ( ) const
- *Rect2* **get_item_area_rect** (*TreeItem* item, *int* column=-1) const
- void **ensure_cursor_is_visible** ( )
- void **set_column_titles_visible** (*bool* visible)
- *bool* **are_column_titles_visible** ( ) const
- void **set_column_title** (*int* column, *String* title)
- *String* **get_column_title** (*int* column) const
- *Vector2* **get_scroll** ( ) const
- void **set_hide_folding** (*bool* hide)
- *bool* **is_folding_hidden** ( ) const

9.329 TreeItem

Inherits: *Object*

Category: Core

9.329.1 Brief Description

9.329.2 Member Functions

void	*set_cell_mode* (*int* column, *int* mode)
int	*get_cell_mode* (*int* column) const
void	*set_checked* (*int* column, *bool* checked)
bool	*is_checked* (*int* column) const
void	*set_text* (*int* column, *String* text)
String	*get_text* (*int* column) const
void	*set_icon* (*int* column, *Texture* texture)
Texture	*get_icon* (*int* column) const
void	*set_icon_region* (*int* column, *Rect2* region)
Rect2	*get_icon_region* (*int* column) const
void	*set_icon_max_width* (*int* column, *int* width)
int	*get_icon_max_width* (*int* column) const
void	*set_range* (*int* column, *float* value)
float	*get_range* (*int* column) const
void	*set_range_config* (*int* column, *float* min, *float* max, *float* step, *bool* expr=false)
Dictionary	*get_range_config* (*int* column)
void	*set_metadata* (*int* column, var meta)
void	*get_metadata* (*int* column) const
void	*set_custom_draw* (*int* column, *Object* object, *String* callback)
void	*set_collapsed* (*bool* enable)
bool	*is_collapsed* ( )
TreeItem	*get_next* ( )
TreeItem	*get_prev* ( )
TreeItem	*get_parent* ( )
TreeItem	*get_children* ( )
TreeItem	*get_next_visible* ( )
TreeItem	*get_prev_visible* ( )
TreeItem	*remove_child* (*Object* child)
void	*set_selectable* (*int* column, *bool* selectable)
bool	*is_selectable* (*int* column) const
bool	*is_selected* (*int* column)
void	*select* (*int* column)
void	*deselect* (*int* column)
void	*set_editable* (*int* column, *bool* enabled)
bool	*is_editable* (*int* column)
void	*set_custom_color* (*int* column, *Color* color)
void	*clear_custom_color* (*int* column)
void	*set_custom_bg_color* (*int* column, *Color* color)
void	*clear_custom_bg_color* (*int* column)
Color	*get_custom_bg_color* (*int* column) const

Continúa en la página siguiente

Tabla 9.32 – proviene de la página anterior

void	*add_button* (*int* column, *Texture* button, *int* button_idx=-1, *bool* disabled=false)
int	*get_button_count* (*int* column) const
Texture	*get_button* (*int* column, *int* button_idx) const
void	*erase_button* (*int* column, *int* button_idx)
bool	*is_button_disabled* (*int* column, *int* button_idx) const
void	*set_tooltip* (*int* column, *String* tooltip)
String	*get_tooltip* (*int* column) const
void	*move_to_top* ( )
void	*move_to_bottom* ( )

9.329.3 Numeric Constants

- **CELL_MODE_STRING = 0**
- **CELL_MODE_CHECK = 1**
- **CELL_MODE_RANGE = 2**
- **CELL_MODE_ICON = 3**
- **CELL_MODE_CUSTOM = 4**

9.329.4 Member Function Description

- void **set_cell_mode** (*int* column, *int* mode)
- *int* **get_cell_mode** (*int* column) const
- void **set_checked** (*int* column, *bool* checked)
- *bool* **is_checked** (*int* column) const
- void **set_text** (*int* column, *String* text)
- *String* **get_text** (*int* column) const
- void **set_icon** (*int* column, *Texture* texture)
- *Texture* **get_icon** (*int* column) const
- void **set_icon_region** (*int* column, *Rect2* region)
- *Rect2* **get_icon_region** (*int* column) const
- void **set_icon_max_width** (*int* column, *int* width)
- *int* **get_icon_max_width** (*int* column) const
- void **set_range** (*int* column, *float* value)
- *float* **get_range** (*int* column) const
- void **set_range_config** (*int* column, *float* min, *float* max, *float* step, *bool* expr=false)
- *Dictionary* **get_range_config** (*int* column)
- void **set_metadata** (*int* column, var meta)
- void **get_metadata** (*int* column) const
- void **set_custom_draw** (*int* column, *Object* object, *String* callback)

- void **set_collapsed** (*bool* enable)
- *bool* **is_collapsed** ( )
- *TreeItem* **get_next** ( )
- *TreeItem* **get_prev** ( )
- *TreeItem* **get_parent** ( )
- *TreeItem* **get_children** ( )
- *TreeItem* **get_next_visible** ( )
- *TreeItem* **get_prev_visible** ( )
- *TreeItem* **remove_child** (*Object* child)
- void **set_selectable** (*int* column, *bool* selectable)
- *bool* **is_selectable** (*int* column) const
- *bool* **is_selected** (*int* column)
- void **select** (*int* column)
- void **deselect** (*int* column)
- void **set_editable** (*int* column, *bool* enabled)
- *bool* **is_editable** (*int* column)
- void **set_custom_color** (*int* column, *Color* color)
- void **clear_custom_color** (*int* column)
- void **set_custom_bg_color** (*int* column, *Color* color)
- void **clear_custom_bg_color** (*int* column)
- *Color* **get_custom_bg_color** (*int* column) const
- void **add_button** (*int* column, *Texture* button, *int* button_idx=-1, *bool* disabled=false)
- *int* **get_button_count** (*int* column) const
- *Texture* **get_button** (*int* column, *int* button_idx) const
- void **erase_button** (*int* column, *int* button_idx)
- *bool* **is_button_disabled** (*int* column, *int* button_idx) const
- void **set_tooltip** (*int* column, *String* tooltip)
- *String* **get_tooltip** (*int* column) const
- void **move_to_top** ( )
- void **move_to_bottom** ( )

9.330 Tween

Inherits: *Node* < *Object*

Category: Core

9.330.1 Brief Description

Node useful for animations with unknown start and end points.

9.330.2 Member Functions

bool	*is_active* ( ) const
void	*set_active* (*bool* active)
bool	*is_repeat* ( ) const
void	*set_repeat* (*bool* repeat)
void	*set_speed* (*float* speed)
float	*get_speed* ( ) const
void	*set_tween_process_mode* (*int* mode)
int	*get_tween_process_mode* ( ) const
bool	*start* ( )
bool	*reset* (*Object* object, *String* key)
bool	*reset_all* ( )
bool	*stop* (*Object* object, *String* key)
bool	*stop_all* ( )
bool	*resume* (*Object* object, *String* key)
bool	*resume_all* ( )
bool	*remove* (*Object* object, *String* key)
bool	*remove_all* ( )
bool	*seek* (*float* time)
float	*tell* ( ) const
float	*get_runtime* ( ) const
bool	*interpolate_property* (*Object* object, *String* property, var initial_val, var final_val, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)
bool	*interpolate_method* (*Object* object, *String* method, var initial_val, var final_val, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)
bool	*interpolate_callback* (*Object* object, *float* times_in_sec, *String* callback, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL)
bool	*interpolate_deferred_callback* (*Object* object, *float* times_in_sec, *String* callback, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL)
bool	*follow_property* (*Object* object, *String* property, var initial_val, *Object* target, *String* target_property, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)
bool	*follow_method* (*Object* object, *String* method, var initial_val, *Object* target, *String* target_method, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)
bool	*targeting_property* (*Object* object, *String* property, *Object* initial, *String* initial_val, var final_val, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)
bool	*targeting_method* (*Object* object, *String* method, *Object* initial, *String* initial_method, var final_val, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)

9.330.3 Signals

- **tween_complete** (*Object* object, *String* key)
- **tween_step** (*Object* object, *String* key, *float* elapsed, *Object* value)
- **tween_start** (*Object* object, *String* key)

9.330.4 Numeric Constants

- **TWEEN_PROCESS_FIXED = 0** — The *Tween* should use `_fixed_process` for timekeeping when this is enabled.
- **TWEEN_PROCESS_IDLE = 1** — The *Tween* should use `_process` for timekeeping when this is enabled (default).
- **TRANS_LINEAR = 0** — Means that the animation is interpolated linearly.
- **TRANS_SINE = 1** — Means that the animation is interpolated using a sine wave.
- **TRANS_QUINT = 2** — Means that the animation is interpolated with a quinary (to the power of 5) function.
- **TRANS_QUART = 3** — Means that the animation is interpolated with a quartic (to the power of 4) function.
- **TRANS_QUAD = 4** — Means that the animation is interpolated with a quadratic (to the power of 2) function.
- **TRANS_EXPO = 5** — Means that the animation is interpolated with a exponential (some number to the power of x) function.
- **TRANS_ELASTIC = 6** — Means that the animation is interpolated with elasticity, wiggling around the edges.
- **TRANS_CUBIC = 7** — Means that the animation is interpolated with a cubic (to the power of 3) function.
- **TRANS_CIRC = 8** — Means that the animation is interpolated with a function using square roots.
- **TRANS_BOUNCE = 9** — Means that the animation is interpolated by bouncing at, but never surpassing, the end.
- **TRANS_BACK = 10** — Means that the animation is interpolated backing out at edges.
- **EASE_IN = 0** — Signifies that the interpolation should be focused in the beginning.
- **EASE_OUT = 1** — Signifies that the interpolation should be focused in the end.
- **EASE_IN_OUT = 2** — Signifies that the interpolation should be focused in both ends.
- **EASE_OUT_IN = 3** — Signifies that the interpolation should be focused in both ends, but they should be switched (a bit hard to explain, try it for yourself to be sure).

9.330.5 Description

Node useful for animations with unknown start and end points, procedural animations, making one node follow another, and other simple behavior.

Because it is easy to get it wrong, here is a quick usage example:

```
var tween = get_node("Tween")
tween.interpolate_property(get_node("Node2D_to_move"), "transform/pos", Vector2(0,0),␣
↪Vector2(100,100), Tween.TRANS_LINEAR, Tween.EASE_IN_OUT)
tween.start()
```

Some of the methods of this class require a property name. You can get the property name by hovering over the property in the inspector of the editor.

Many of the methods accept `trans_type` and `ease_type`. The first accepts an TRANS_* constant, and refers to the way the timing of the animation is handled (you might want to see `http://easings.net/` for some examples). The second accepts an EASE_* constant, and controls the where `trans_type` is applied to the interpolation (in the begining, the end, or both). If you don't know which transision and easing to pick, you can try different TRANS_* constants with EASE_IN_OUT, and use the one that looks best.

9.330.6 Member Function Description

- *bool* **is_active** ( ) const

Returns true if any tweens are currently running, and false otherwise. Note that this method doesn't consider tweens that have ended.

- void **set_active** (*bool* active)

Activate/deactivate the tween. You can use this for pausing animations, though *stop_all* and *resume_all* might be more fit for this.

- *bool* **is_repeat** ( ) const

Returns true if repeat has been set from editor GUI or *set_repeat*.

- void **set_repeat** (*bool* repeat)

Make the tween repeat after all tweens have finished.

- void **set_speed** (*float* speed)

Set the speed multiplier of the tween. Set it to 1 for normal speed, 2 for two times nromal speed, and 0.5 for half of the normal speed. Setting it to 0 would pause the animation, but you might consider using *set_active* or *stop_all* and *resume_all* for this.

- *float* **get_speed** ( ) const

Returns the speed that has been set from editor GUI or *set_repeat*.

- void **set_tween_process_mode** (*int* mode)

Set whether the Tween uses `_process` or `_fixed_process` (accepts TWEEN_PROCESS_IDLE and TWEEN_PROCESS_FIXED constants, respectively).

- *int* **get_tween_process_mode** ( ) const

Returns the process mode that has been set from editor GUI or *set_tween_process_mode*

- *bool* **start** ( )

Start the tween node. You can define tweens both before and after this.

- *bool* **reset** (*Object* object, *String* key)

Resets a tween to the initial value (the one given, not the one before the tween), given its object and property/method pair.

- *bool* **reset_all** ( )

Resets all tweens to their initial values (the ones given, not those before the tween).

- *bool* **stop** (*Object* object, *String* key)

Stop animating a tween, given its object and property/method pair.

- *bool* **stop_all** ( )

Stop animating all tweens.

- *bool* **resume** (*Object* object, *String* key)

Continue animating a stopped tween, given its object and property/method pair.

- *bool* **resume_all** ( )

Continue animating all stopped tweens.

- *bool* **remove** (*Object* object, *String* key)

Stop animating and completely remove a tween, given its object and property/method pair.

- *bool* **remove_all** ( )

Stop animating and completely remove all tweens.

- *bool* **seek** (*float* time)

Seek the animation to the given `time` in seconds.

- *float* **tell** ( ) const

Returns the current time of the tween.

- *float* **get_runtime** ( ) const

Returns the time needed for all tweens to end in seconds, measured from the start. Thus, if you have two tweens, one ending 10 seconds after the start and the other - 20 seconds, it would return 20 seconds, as by that time all tweens would have finished.

- *bool* **interpolate_property** (*Object* object, *String* property, var initial_val, var final_val, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)

Animate `property` of `object` from `initial_val` to `final_val` for `times_in_sec` seconds, `delay` seconds later.

`trans_type` accepts TRANS_* constants, and is the way the animation is interpolated, while `ease_type` accepts EASE_* constants, and controls the place of the interpolation (the begining, the end, or both). You can read more about them in the class description.

- *bool* **interpolate_method** (*Object* object, *String* method, var initial_val, var final_val, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)

Animate `method` of `object` from `initial_val` to `final_val` for `times_in_sec` seconds, `delay` seconds later. Methods are animated by calling them with consecuitive values.

`trans_type` accepts TRANS_* constants, and is the way the animation is interpolated, while `ease_type` accepts EASE_* constants, and controls the place of the interpolation (the begining, the end, or both). You can read more about them in the class description.

- *bool* **interpolate_callback** (*Object* object, *float* times_in_sec, *String* callback, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL)

Call `callback` of `object` after `times_in_sec`. `arg1-arg5` are arguments to be passed to the callback.

- *bool* **interpolate_deferred_callback** (*Object* object, *float* times_in_sec, *String* callback, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL)

Call `callback` of `object` after `times_in_sec` on the main thread (similar to methog Object.call_deferred). [code<class_methog object.call_deferred). [code>‘arg1‘-`arg5` are arguments to be passed to the callback.

- *bool* **follow_property** (*Object* object, *String* property, var initial_val, *Object* target, *String* target_property, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)

Follow `property` of `object` and apply it on `target_property` of `target`, beginning from `initial_val` for `times_in_sec` seconds, `delay` seconds later. Note that `target:target_property` would equal `object:property` at the end of the tween.

`trans_type` accepts TRANS_* constants, and is the way the animation is interpolated, while `ease_type` accepts EASE_* constants, and controls the place of the interpolation (the begining, the end, or both). You can read more about them in the class description.

- *bool* **follow_method** (*Object* object, *String* method, var initial_val, *Object* target, *String* target_method, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)

Follow `method` of `object` and apply the returned value on `target_method` of `target`, beginning from `initial_val` for `times_in_sec` seconds, `delay` later. Methods are animated by calling them with consequitive values.

`trans_type` accepts TRANS_* constants, and is the way the animation is interpolated, while `ease_type` accepts EASE_* constants, and controls the place of the interpolation (the begining, the end, or both). You can read more about them in the class description.

- *bool* **targeting_property** (*Object* object, *String* property, *Object* initial, *String* initial_val, var final_val, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)

Animate `property` of `object` from the current value of the `initial_val` property of `initial` to `final_val` for `times_in_sec` seconds, `delay` seconds later.

`trans_type` accepts TRANS_* constants, and is the way the animation is interpolated, while `ease_type` accepts EASE_* constants, and controls the place of the interpolation (the begining, the end, or both). You can read more about them in the class description.

- *bool* **targeting_method** (*Object* object, *String* method, *Object* initial, *String* initial_method, var final_val, *float* times_in_sec, *int* trans_type, *int* ease_type, *float* delay=0)

Animate `method` of `object` from the value returned by `initial.initial_method` to `final_val` for `times_in_sec` seconds, `delay` seconds later. Methods are animated by calling them with consecuitive values.

`trans_type` accepts TRANS_* constants, and is the way the animation is interpolated, while `ease_type` accepts EASE_* constants, and controls the place of the interpolation (the begining, the end, or both). You can read more about them in the class description.

9.331 UndoRedo

Inherits: *Object*

Category: Core

9.331.1 Brief Description

9.331.2 Member Functions

void	*create_action* (*String* name, *bool* mergeable=false)
void	*commit_action* ( )
void	*add_do_method* (*Object* object, *String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL)
void	*add_undo_method* (*Object* object, *String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL)
void	*add_do_property* (*Object* object, *String* property, Variant value)
void	*add_undo_property* (*Object* object, *String* property, Variant value)
void	*add_do_reference* (*Object* object)
void	*add_undo_reference* (*Object* object)
void	*clear_history* ( )
String	*get_current_action_name* ( ) const
int	*get_version* ( ) const

9.331.3 Member Function Description

- void **create_action** (*String* name, *bool* mergeable=false)
- void **commit_action** ( )
- void **add_do_method** (*Object* object, *String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL)
- void **add_undo_method** (*Object* object, *String* method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL)
- void **add_do_property** (*Object* object, *String* property, Variant value)
- void **add_undo_property** (*Object* object, *String* property, Variant value)
- void **add_do_reference** (*Object* object)
- void **add_undo_reference** (*Object* object)
- void **clear_history** ( )
- *String* **get_current_action_name** ( ) const
- *int* **get_version** ( ) const

9.332 VBoxContainer

Inherits: *BoxContainer* < *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.332.1 Brief Description

Vertical box container.

9.332.2 Description

Vertical box container. See *BoxContainer*.

9.333 VButtonArray

Inherits: *ButtonArray* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.333.1 Brief Description

Vertical button array.

9.333.2 Description

Vertical button array. See *ButtonArray*.

9.334 Vector2

Category: Built-In Types

9.334.1 Brief Description

Vector used for 2D Math.

9.334.2 Member Functions

float	*angle* ( )
float	*angle_to* (*Vector2* to)
float	*angle_to_point* (*Vector2* to)
Vector2	*cubic_interpolate* (*Vector2* b, *Vector2* pre_a, *Vector2* post_b, *float* t)
float	*distance_squared_to* (*Vector2* to)
float	*distance_to* (*Vector2* to)
float	*dot* (*Vector2* with)
Vector2	*floor* ( )
Vector2	*floorf* ( )
float	*get_aspect* ( )
float	*length* ( )
float	*length_squared* ( )
Vector2	*linear_interpolate* (*Vector2* b, *float* t)
Vector2	*normalized* ( )
Vector2	*reflect* (*Vector2* vec)
Vector2	*rotated* (*float* phi)
Vector2	*slide* (*Vector2* vec)
Vector2	*snapped* (*Vector2* by)
Vector2	*tangent* ( )
Vector2	*Vector2* (*float* x, *float* y)

9.334.3 Member Variables

- *float* **x** - X component of the vector.
- *float* **y** - Y component of the vector.
- *float* **width** - Width of the vector (Same as X).
- *float* **height** - Height of the vector (Same as Y).

9.334.4 Description

2-element structure that can be used to represent positions in 2d-space, or any other pair of numeric values.

9.334.5 Member Function Description

- *float* **angle** ( )

Returns the result of atan2 when called with the Vector's x and y as parameters (Math::atan2(x,y)).

Be aware that it therefore returns an angle oriented clockwise with regard to the (0, 1) unit vector, and not an angle oriented counter-clockwise with regard to the (1, 0) unit vector (which would be the typical trigonometric representation of the angle when calling Math::atan2(y,x)).

- *float* **angle_to** (*Vector2* to)

Returns the angle in radians between the two vectors.

- *float* **angle_to_point** (*Vector2* to)

Returns the angle in radians between the line connecting the two points and the x coordinate.

- *Vector2* **cubic_interpolate** (*Vector2* b, *Vector2* pre_a, *Vector2* post_b, *float* t)

Cubicly interpolates between this Vector and "b", using "pre_a" and "post_b" as handles, and returning the result at position "t".

- *float* **distance_squared_to** (*Vector2* to)

Returns the squared distance to vector "b". Prefer this function over "distance_to" if you need to sort vectors or need the squared distance for some formula.

- *float* **distance_to** (*Vector2* to)

Returns the distance to vector "b".

- *float* **dot** (*Vector2* with)

Returns the dot product with vector "b".

- *Vector2* **floor** ( )

Remove the fractional part of x and y.

- *Vector2* **floorf** ( )

Remove the fractional part of x and y.

- *float* **get_aspect** ( )

Returns the ratio of X to Y.

- *float* **length** ( )

Returns the length of the vector.

- *float* **length_squared** ( )

Returns the squared length of the vector. Prefer this function over "length" if you need to sort vectors or need the squared length for some formula.

- *Vector2* **linear_interpolate** (*Vector2* b, *float* t)

Returns the result of the linear interpolation between this vector and "b", by amount "t".

- *Vector2* **normalized** ( )

Returns a normalized vector to unit length.

- *Vector2* **reflect** (*Vector2* vec)

Like "slide", but reflects the Vector instead of continuing along the wall.

- *Vector2* **rotated** (*float* phi)

Rotates the vector by "phi" radians.

- *Vector2* **slide** (*Vector2* vec)

Slides the vector by the other vector.

- *Vector2* **snapped** (*Vector2* by)

Snaps the vector to a grid with the given size.

- *Vector2* **tangent** ( )

Returns a perpendicular vector.

- *Vector2* **Vector2** (*float* x, *float* y)

Constructs a new Vector2 from the given x and y.

9.335 Vector2Array

Category: Built-In Types

9.335.1 Brief Description

An Array of Vector2.

9.335.2 Member Functions

void	*push_back* (*Vector2* vector2)
void	*resize* (*int* idx)
void	*set* (*int* idx, *Vector2* vector2)
int	*size* ( )
Vector2Array	*Vector2Array* (*Array* from)

9.335.3 Description

An Array specifically designed to hold Vector2.

9.335.4 Member Function Description

- void **push_back** (*Vector2* vector2)

Inserts a Vector2 at the end.

- void **resize** (*int* idx)

Sets the size of the Vector2Array. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.

- void **set** (*int* idx, *Vector2* vector2)

Changes the Vector2 at the given index.

- *int* **size** ( )

Returns the size of the array.

- *Vector2Array* **Vector2Array** (*Array* from)

Constructs a new Vector2Array. Optionally, you can pass in an Array that will be converted.

9.336 Vector3

Category: Built-In Types

9.336.1 Brief Description

Vector class, which performs basic 3D vector math operations.

9.336.2 Member Functions

Vector3	*abs* ( )
Vector3	*ceil* ( )
Vector3	*cross* (*Vector3* b)
Vector3	*cubic_interpolate* (*Vector3* b, *Vector3* pre_a, *Vector3* post_b, *float* t)
float	*distance_squared_to* (*Vector3* b)
float	*distance_to* (*Vector3* b)
float	*dot* (*Vector3* b)
Vector3	*floor* ( )
Vector3	*inverse* ( )
float	*length* ( )
float	*length_squared* ( )
Vector3	*linear_interpolate* (*Vector3* b, *float* t)
int	*max_axis* ( )
int	*min_axis* ( )
Vector3	*normalized* ( )
Vector3	*reflect* (*Vector3* by)
Vector3	*rotated* (*Vector3* axis, *float* phi)
Vector3	*slide* (*Vector3* by)
Vector3	*snapped* (*float* by)
Vector3	*Vector3* (*float* x, *float* y, *float* z)

9.336.3 Member Variables

- *float* **x** - X component of the vector.
- *float* **y** - Y component of the vector.
- *float* **z** - Z component of the vector.

9.336.4 Numeric Constants

- **AXIS_X = 0** — Enumerated value for the X axis. Returned by functions like max_axis or min_axis.
- **AXIS_Y = 1** — Enumerated value for the Y axis.
- **AXIS_Z = 2** — Enumerated value for the Z axis.

9.336.5 Description

Vector3 is one of the core classes of the engine, and includes several built-in helper functions to perform basic vector math operations.

9.336.6 Member Function Description

- *Vector3* **abs ( )**

Returns a new vector with all components in absolute values (e.g. positive).

- *Vector3* **ceil ( )**

Returns a new vector with all components rounded up.

- *Vector3* **cross (** *Vector3* b **)**

Return the cross product with b.

- *Vector3* **cubic_interpolate (** *Vector3* b, *Vector3* pre_a, *Vector3* post_b, *float* t **)**

Perform a cubic interpolation between vectors pre_a, a, b, post_b (a is current), by the given amount (t).

- *float* **distance_squared_to (** *Vector3* b **)**

Return the squared distance (distance minus the last square root) to b. Prefer this function over distance_to if you need to sort vectors or need the squared distance for some formula.

- *float* **distance_to (** *Vector3* b **)**

Return the distance to b.

- *float* **dot (** *Vector3* b **)**

Return the dot product with b.

- *Vector3* **floor ( )**

Returns a new vector with all components rounded down.

- *Vector3* **inverse ( )**

Returns the inverse of the vector. This is the same as Vector3(1.0 / v.x, 1.0 / v.y, 1.0 / v.z)

- *float* **length ( )**

Return the length of the vector.

- *float* **length_squared ( )**

Return the length of the vector, squared. Prefer this function over "length" if you need to sort vectors or need the squared length for some formula.

- *Vector3* **linear_interpolate (** *Vector3* b, *float* t **)**

Linearly interpolates the vector to a given one (b), by the given amount (t).

- *int* **max_axis ( )**

Returns AXIS_X, AXIS_Y or AXIS_Z depending on which axis is the largest.

- *int* **min_axis ( )**

Returns AXIS_X, AXIS_Y or AXIS_Z depending on which axis is the smallest.

- *Vector3* **normalized ( )**

Return a copy of the normalized vector to unit length. This is the same as v / v.length().

- *Vector3* **reflect** (*Vector3* by)

Like "slide", but reflects the Vector instead of continuing along the wall.

- *Vector3* **rotated** (*Vector3* axis, *float* phi)

Rotates the vector around some axis by phi radians.

- *Vector3* **slide** (*Vector3* by)

Slides the vector along a wall.

- *Vector3* **snapped** (*float* by)

Return a copy of the vector, snapped to the lowest neared multiple.

- *Vector3* **Vector3** (*float* x, *float* y, *float* z)

Returns a Vector3 with the given components.

9.337 Vector3Array

Category: Built-In Types

9.337.1 Brief Description

An Array of Vector3.

9.337.2 Member Functions

void	*push_back* (*Vector3* vector3)
void	*resize* (*int* idx)
void	*set* (*int* idx, *Vector3* vector3)
int	*size* ( )
Vector3Array	*Vector3Array* (*Array* from)

9.337.3 Description

An Array specifically designed to hold Vector3.

9.337.4 Member Function Description

- void **push_back** (*Vector3* vector3)

Inserts a Vector3 at the end.

- void **resize** (*int* idx)

Sets the size of the Vector3Array. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.

- void **set** (*int* idx, *Vector3* vector3)

Changes the Vector3 at the given index.

- *int* **size** ( )

Returns the size of the array.

- *Vector3Array* **Vector3Array** (*Array* from)

Constructs a new Vector3Array. Optionally, you can pass in an Array that will be converted.

9.338 VehicleBody

Inherits: *PhysicsBody* < *CollisionObject* < *Spatial* < *Node* < *Object*

Category: Core

9.338.1 Brief Description

9.338.2 Member Functions

void	*set_mass* (*float* mass)
float	*get_mass* ( ) const
void	*set_friction* (*float* friction)
float	*get_friction* ( ) const
void	*set_engine_force* (*float* engine_force)
float	*get_engine_force* ( ) const
void	*set_brake* (*float* brake)
float	*get_brake* ( ) const
void	*set_steering* (*float* steering)
float	*get_steering* ( ) const

9.338.3 Member Function Description

- void **set_mass** (*float* mass)
- *float* **get_mass** ( ) const
- void **set_friction** (*float* friction)
- *float* **get_friction** ( ) const
- void **set_engine_force** (*float* engine_force)
- *float* **get_engine_force** ( ) const
- void **set_brake** (*float* brake)
- *float* **get_brake** ( ) const
- void **set_steering** (*float* steering)
- *float* **get_steering** ( ) const

9.339 VehicleWheel

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.339.1 Brief Description

9.339.2 Member Functions

void	*set_radius* (*float* length)
float	*get_radius* ( ) const
void	*set_suspension_rest_length* (*float* length)
float	*get_suspension_rest_length* ( ) const
void	*set_suspension_travel* (*float* length)
float	*get_suspension_travel* ( ) const
void	*set_suspension_stiffness* (*float* length)
float	*get_suspension_stiffness* ( ) const
void	*set_suspension_max_force* (*float* length)
float	*get_suspension_max_force* ( ) const
void	*set_damping_compression* (*float* length)
float	*get_damping_compression* ( ) const
void	*set_damping_relaxation* (*float* length)
float	*get_damping_relaxation* ( ) const
void	*set_use_as_traction* (*bool* enable)
bool	*is_used_as_traction* ( ) const
void	*set_use_as_steering* (*bool* enable)
bool	*is_used_as_steering* ( ) const
void	*set_friction_slip* (*float* length)
float	*get_friction_slip* ( ) const

9.339.3 Member Function Description

- void **set_radius** (*float* length)
- *float* **get_radius** ( ) const
- void **set_suspension_rest_length** (*float* length)
- *float* **get_suspension_rest_length** ( ) const
- void **set_suspension_travel** (*float* length)
- *float* **get_suspension_travel** ( ) const
- void **set_suspension_stiffness** (*float* length)
- *float* **get_suspension_stiffness** ( ) const
- void **set_suspension_max_force** (*float* length)
- *float* **get_suspension_max_force** ( ) const
- void **set_damping_compression** (*float* length)
- *float* **get_damping_compression** ( ) const

- void **set_damping_relaxation** (*float* length)
- *float* **get_damping_relaxation** ( ) const
- void **set_use_as_traction** (*bool* enable)
- *bool* **is_used_as_traction** ( ) const
- void **set_use_as_steering** (*bool* enable)
- *bool* **is_used_as_steering** ( ) const
- void **set_friction_slip** (*float* length)
- *float* **get_friction_slip** ( ) const

9.340 VideoPlayer

Inherits: *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.340.1 Brief Description

9.340.2 Member Functions

void	*set_stream* (*VideoStream* stream)
VideoStream	*get_stream* ( ) const
void	*play* ( )
void	*stop* ( )
bool	*is_playing* ( ) const
void	*set_paused* (*bool* paused)
bool	*is_paused* ( ) const
void	*set_volume* (*float* volume)
float	*get_volume* ( ) const
void	*set_volume_db* (*float* db)
float	*get_volume_db* ( ) const
void	*set_audio_track* (*int* track)
int	*get_audio_track* ( ) const
String	*get_stream_name* ( ) const
float	*get_stream_pos* ( ) const
void	*set_autoplay* (*bool* enabled)
bool	*has_autoplay* ( ) const
void	*set_expand* (*bool* enable)
bool	*has_expand* ( ) const
void	*set_buffering_msec* (*int* msec)
int	*get_buffering_msec* ( ) const
Texture	*get_video_texture* ( )

9.340.3 Member Function Description

- void **set_stream** (*VideoStream* stream)
- *VideoStream* **get_stream** ( ) const

- void **play** ( )
- void **stop** ( )
- *bool* **is_playing** ( ) const
- void **set_paused** (*bool* paused)
- *bool* **is_paused** ( ) const
- void **set_volume** (*float* volume)
- *float* **get_volume** ( ) const
- void **set_volume_db** (*float* db)
- *float* **get_volume_db** ( ) const
- void **set_audio_track** (*int* track)
- *int* **get_audio_track** ( ) const
- *String* **get_stream_name** ( ) const
- *float* **get_stream_pos** ( ) const
- void **set_autoplay** (*bool* enabled)
- *bool* **has_autoplay** ( ) const
- void **set_expand** (*bool* enable)
- *bool* **has_expand** ( ) const
- void **set_buffering_msec** (*int* msec)
- *int* **get_buffering_msec** ( ) const
- *Texture* **get_video_texture** ( )

9.341 VideoStream

Inherits: *Resource* < *Reference* < *Object*

Inherited By: *VideoStreamTheora*

Category: Core

9.341.1 Brief Description

9.342 VideoStreamTheora

Inherits: *VideoStream* < *Resource* < *Reference* < *Object*

Category: Core

9.342.1 Brief Description

9.343 Viewport

Inherits: *Node* < *Object*

Category: Core

9.343.1 Brief Description

Creates a sub-view into the screen.

9.343.2 Member Functions

void	*set_rect* (*Rect2* rect)
Rect2	*get_rect* ( ) const
World2D	*find_world_2d* ( ) const
void	*set_world* (*World* world)
World	*get_world* ( ) const
World	*find_world* ( ) const
void	*set_canvas_transform* (*Matrix32* xform)
Matrix32	*get_canvas_transform* ( ) const
void	*set_global_canvas_transform* (*Matrix32* xform)
Matrix32	*get_global_canvas_transform* ( ) const
Matrix32	*get_final_transform* ( ) const
Rect2	*get_visible_rect* ( ) const
void	*set_transparent_background* (*bool* enable)
bool	*has_transparent_background* ( ) const
void	*set_size_override* (*bool* enable, *Vector2* size=Vector2(-1,-1), *Vector2* margin=Vector2(0,0))
Vector2	*get_size_override* ( ) const
bool	*is_size_override_enabled* ( ) const
void	*set_size_override_stretch* (*bool* enabled)
bool	*is_size_override_stretch_enabled* ( ) const
void	*queue_screen_capture* ( )
Image	*get_screen_capture* ( ) const
void	*set_as_render_target* (*bool* enable)
bool	*is_set_as_render_target* ( ) const
void	*set_render_target_vflip* (*bool* enable)
bool	*get_render_target_vflip* ( ) const
void	*set_render_target_clear_on_new_frame* (*bool* enable)
bool	*get_render_target_clear_on_new_frame* ( ) const
void	*render_target_clear* ( )
void	*set_render_target_filter* (*bool* enable)
bool	*get_render_target_filter* ( ) const
void	*set_render_target_gen_mipmaps* (*bool* enable)
bool	*get_render_target_gen_mipmaps* ( ) const
void	*set_render_target_update_mode* (*int* mode)
int	*get_render_target_update_mode* ( ) const
RenderTargetTexture	*get_render_target_texture* ( ) const

Continúa en la página siguiente

Tabla 9.33 – proviene de la página anterior

void	*set_physics_object_picking* (*bool* enable)
bool	*get_physics_object_picking* ( )
RID	*get_viewport* ( ) const
void	*input* (*InputEvent* local_event)
void	*unhandled_input* (*InputEvent* local_event)
void	*update_worlds* ( )
void	*set_use_own_world* (*bool* enable)
bool	*is_using_own_world* ( ) const
Camera	*get_camera* ( ) const
void	*set_as_audio_listener* (*bool* enable)
bool	*is_audio_listener* ( ) const
void	*set_as_audio_listener_2d* (*bool* enable)
bool	*is_audio_listener_2d* ( ) const
void	*set_render_target_to_screen_rect* (*Rect2* rect)
Vector2	*get_mouse_pos* ( ) const
void	*warp_mouse* (*Vector2* to_pos)
bool	*gui_has_modal_stack* ( ) const
void	*set_disable_input* (*bool* disable)
bool	*is_input_disabled* ( ) const

9.343.3 Signals

- **size_changed** ( )

9.343.4 Numeric Constants

- **RENDER_TARGET_UPDATE_DISABLED = 0**
- **RENDER_TARGET_UPDATE_ONCE = 1**
- **RENDER_TARGET_UPDATE_WHEN_VISIBLE = 2**
- **RENDER_TARGET_UPDATE_ALWAYS = 3**

9.343.5 Description

A Viewport creates a different view into the screen, or a sub-view inside another viewport. Children 2D Nodes will display on it, and children Camera 3D nodes will render on it too.

Optionally, a viewport can have its own 2D or 3D world, so they don't share what they draw with other viewports.

If a viewport is a child of a *Control*, it will automatically take up its same rect and position, otherwise they must be set manually.

Viewports can also choose to be audio listeners, so they generate positional audio depending on a 2D or 3D camera child of it.

Also, viewports can be assigned to different screens in case the devices have multiple screens.

Finally, viewports can also behave as render targets, in which case they will not be visible unless the associated texture is used to draw.

9.343.6 Member Function Description

- void **set_rect** (*Rect2* rect)

Set the viewport rect. If the viewport is child of a control, it will use the same rect as the parent.

- *Rect2* **get_rect** ( ) const

Return the viewport rect. If the viewport is child of a control, it will use the same rect as the parent. Otherwise, if the rect is empty, the viewport will use all the allowed space.

- *World2D* **find_world_2d** ( ) const
- void **set_world** (*World* world)
- *World* **get_world** ( ) const
- *World* **find_world** ( ) const
- void **set_canvas_transform** (*Matrix32* xform)
- *Matrix32* **get_canvas_transform** ( ) const
- void **set_global_canvas_transform** (*Matrix32* xform)
- *Matrix32* **get_global_canvas_transform** ( ) const
- *Matrix32* **get_final_transform** ( ) const
- *Rect2* **get_visible_rect** ( ) const

Return the final, visible rect in global screen coordinates.

- void **set_transparent_background** (*bool* enable)

If this viewport is a child of another viewport, keep the previously drawn background visible.

- *bool* **has_transparent_background** ( ) const

Return whether the viewport lets whatever is behind it to show.

- void **set_size_override** (*bool* enable, *Vector2* size=Vector2(-1,-1), *Vector2* margin=Vector2(0,0))
- *Vector2* **get_size_override** ( ) const
- *bool* **is_size_override_enabled** ( ) const
- void **set_size_override_stretch** (*bool* enabled)
- *bool* **is_size_override_stretch_enabled** ( ) const
- void **queue_screen_capture** ( )
- *Image* **get_screen_capture** ( ) const
- void **set_as_render_target** (*bool* enable)
- *bool* **is_set_as_render_target** ( ) const
- void **set_render_target_vflip** (*bool* enable)
- *bool* **get_render_target_vflip** ( ) const
- void **set_render_target_clear_on_new_frame** (*bool* enable)
- *bool* **get_render_target_clear_on_new_frame** ( ) const
- void **render_target_clear** ( )
- void **set_render_target_filter** (*bool* enable)

- *bool* **get_render_target_filter** ( ) const
- void **set_render_target_gen_mipmaps** (*bool* enable)
- *bool* **get_render_target_gen_mipmaps** ( ) const
- void **set_render_target_update_mode** (*int* mode)
- *int* **get_render_target_update_mode** ( ) const
- *RenderTargetTexture* **get_render_target_texture** ( ) const
- void **set_physics_object_picking** (*bool* enable)
- *bool* **get_physics_object_picking** ( )
- *RID* **get_viewport** ( ) const

Get the viewport RID from the visual server.

- void **input** (*InputEvent* local_event)
- void **unhandled_input** (*InputEvent* local_event)
- void **update_worlds** ( )
- void **set_use_own_world** (*bool* enable)
- *bool* **is_using_own_world** ( ) const
- *Camera* **get_camera** ( ) const
- void **set_as_audio_listener** (*bool* enable)
- *bool* **is_audio_listener** ( ) const
- void **set_as_audio_listener_2d** (*bool* enable)
- *bool* **is_audio_listener_2d** ( ) const
- void **set_render_target_to_screen_rect** (*Rect2* rect)
- *Vector2* **get_mouse_pos** ( ) const
- void **warp_mouse** (*Vector2* to_pos)
- *bool* **gui_has_modal_stack** ( ) const
- void **set_disable_input** (*bool* disable)
- *bool* **is_input_disabled** ( ) const

9.344 ViewportSprite

Inherits: *Node2D* **<** *CanvasItem* **<** *Node* **<** *Object*

Category: Core

9.344.1 Brief Description

9.344.2 Member Functions

void	*set_viewport_path* (*NodePath* path)
NodePath	*get_viewport_path* ( ) const
void	*set_centered* (*bool* centered)
bool	*is_centered* ( ) const
void	*set_offset* (*Vector2* offset)
Vector2	*get_offset* ( ) const
void	*set_modulate* (*Color* modulate)
Color	*get_modulate* ( ) const

9.344.3 Member Function Description

- void **set_viewport_path** (*NodePath* path)
- *NodePath* **get_viewport_path** ( ) const
- void **set_centered** (*bool* centered)
- *bool* **is_centered** ( ) const
- void **set_offset** (*Vector2* offset)
- *Vector2* **get_offset** ( ) const
- void **set_modulate** (*Color* modulate)
- *Color* **get_modulate** ( ) const

9.345 VisibilityEnabler

Inherits: *VisibilityNotifier* < *Spatial* < *Node* < *Object*

Category: Core

9.345.1 Brief Description

Enable certain nodes only when visible.

9.345.2 Member Functions

void	*set_enabler* (*int* enabler, *bool* enabled)
bool	*is_enabler_enabled* (*int* enabler) const

9.345.3 Numeric Constants

- **ENABLER_FREEZE_BODIES = 1** — This enabler will freeze *RigidBody* nodes.
- **ENABLER_PAUSE_ANIMATIONS = 0** — This enabler will pause *AnimationPlayer* nodes.
- **ENABLER_MAX = 2**

9.345.4 Description

The VisibilityEnabler will disable *RigidBody* and *AnimationPlayer* nodes when they are not visible. It will only affect other nodes within the same scene as the VisibilityEnabler itself.

9.345.5 Member Function Description

- void **set_enabler** (*int* enabler, *bool* enabled)

Set an enabler to true for all nodes of its type to be disabled when the VisibilityEnabler is not in view. See the constants for enablers and what they affect.

- *bool* **is_enabler_enabled** (*int* enabler) const

Returns whether the specified enabler was set to true or not.

9.346 VisibilityEnabler2D

Inherits: *VisibilityNotifier2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.346.1 Brief Description

Enable certain nodes only when visible.

9.346.2 Member Functions

void	*set_enabler* (*int* enabler, *bool* enabled)
bool	*is_enabler_enabled* (*int* enabler) const

9.346.3 Numeric Constants

- **ENABLER_FREEZE_BODIES = 1** — This enabler will freeze *RigidBody2D* nodes.
- **ENABLER_PAUSE_ANIMATIONS = 0** — This enabler will pause *AnimationPlayer* nodes.
- **ENABLER_PAUSE_PARTICLES = 2** — This enabler will stop *Particles2D* nodes.
- **ENABLER_PARENT_PROCESS = 3** — This enabler will stop the parent's _process function.
- **ENABLER_PARENT_FIXED_PROCESS = 4** — This enabler will stop the parent's _fixed_process function.
- **ENABLER_MAX = 5**

9.346.4 Description

The VisibilityEnabler2D will disable *RigidBody2D*, *AnimationPlayer*, and other nodes when they are not visible. It will only affect other nodes within the same scene as the VisibilityEnabler2D itself.

9.346.5 Member Function Description

- void **set_enabler** (*int* enabler, *bool* enabled)

Set an enabler to true for all nodes of its type to be disabled when the VisibilityEnabler2D is not in view. See the constants for enablers and what they affect.

- *bool* **is_enabler_enabled** (*int* enabler) const

Returns whether the specified enabler was set to true or not.

9.347 VisibilityNotifier

Inherits: *Spatial* < *Node* < *Object*

Inherited By: *VisibilityEnabler*

Category: Core

9.347.1 Brief Description

Detect when the node is visible on screen.

9.347.2 Member Functions

void	*set_aabb* (*AABB* rect)
AABB	*get_aabb* ( ) const
bool	*is_on_screen* ( ) const

9.347.3 Signals

- **enter_screen** ( )
- **enter_camera** (*Object* camera)
- **exit_screen** ( )
- **exit_camera** (*Object* camera)

9.347.4 Description

The VisibilityNotifier is used to notify when its bounding box enters the screen, is visible on the screen, or when it exits the screen.

9.347.5 Member Function Description

- void **set_aabb** (*AABB* rect)

Set the visibility bounding box of the VisibilityNotifier.

- *AABB* **get_aabb** ( ) const

Return the visibility bounding box of the VisibilityNotifier.

- *bool* **is_on_screen** ( ) const

Return true if any part of the bounding box is on the screen.

9.348 VisibilityNotifier2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Inherited By: *VisibilityEnabler2D*

Category: Core

9.348.1 Brief Description

Detect when the node is visible on screen.

9.348.2 Member Functions

void	*set_rect* (*Rect2* rect)
Rect2	*get_rect* ( ) const
bool	*is_on_screen* ( ) const

9.348.3 Signals

- **enter_screen** ( )
- **enter_viewport** (*Object* viewport)
- **exit_screen** ( )
- **exit_viewport** (*Object* viewport)

9.348.4 Description

The VisibilityNotifier2D is used to notify when its bounding rectangle enters the screen, is visible on the screen, or when it exits the screen.

9.348.5 Member Function Description

- void **set_rect** (*Rect2* rect)

Set the visibility bounding rectangle of the VisibilityNotifier2D.

- *Rect2* **get_rect** ( ) const

Return the visibility bounding rectangle of the VisibilityNotifier2D.

- *bool* **is_on_screen** ( ) const

Return true if any part of the bounding rectangle is on the screen.

9.349 VisualInstance

Inherits: *Spatial* < *Node* < *Object*

Inherited By: *BakedLightInstance*, *Light*, *Room*, *BakedLightSampler*, *Portal*, *GeometryInstance*

Category: Core

9.349.1 Brief Description

9.349.2 Member Functions

void	*set_base* (*RID* base)
void	*set_layer_mask* (*int* mask)
int	*get_layer_mask* ( ) const

9.349.3 Member Function Description

- void **set_base** (*RID* base)
- void **set_layer_mask** (*int* mask)
- *int* **get_layer_mask** ( ) const

9.350 VisualServer

Inherits: *Object*

Category: Core

9.350.1 Brief Description

Server for anything visible.

9.350.2 Member Functions

RID	*texture_create* ( )
RID	*texture_create_from_image* (*Image* arg0, *int* arg1=7)
void	*texture_set_flags* (*RID* arg0, *int* arg1)
int	*texture_get_flags* (*RID* arg0) const
int	*texture_get_width* (*RID* arg0) const
int	*texture_get_height* (*RID* arg0) const
void	*texture_set_shrink_all_x2_on_set_data* (*bool* shrink)
RID	*shader_create* (*int* mode=0)
void	*shader_set_mode* (*RID* shader, *int* mode)
RID	*material_create* ( )
void	*material_set_shader* (*RID* shader, *RID* arg1)
RID	*material_get_shader* (*RID* arg0) const

Continúa en la págin

Tabla 9.34 – proviene de la página anterior

void	*material_set_param* (*RID* arg0, *String* arg1, var arg2)
void	*material_get_param* (*RID* arg0, *String* arg1) const
void	*material_set_flag* (*RID* arg0, *int* arg1, *bool* arg2)
bool	*material_get_flag* (*RID* arg0, *int* arg1) const
void	*material_set_blend_mode* (*RID* arg0, *int* arg1)
int	*material_get_blend_mode* (*RID* arg0) const
void	*material_set_line_width* (*RID* arg0, *float* arg1)
float	*material_get_line_width* (*RID* arg0) const
RID	*mesh_create* ( )
void	*mesh_add_surface* (*RID* arg0, *int* arg1, *Array* arg2, *Array* arg3, *bool* arg4=-1)
void	*mesh_surface_set_material* (*RID* arg0, *int* arg1, *RID* arg2, *bool* arg3=false)
RID	*mesh_surface_get_material* (*RID* arg0, *int* arg1) const
int	*mesh_surface_get_array_len* (*RID* arg0, *int* arg1) const
int	*mesh_surface_get_array_index_len* (*RID* arg0, *int* arg1) const
int	*mesh_surface_get_format* (*RID* arg0, *int* arg1) const
int	*mesh_surface_get_primitive_type* (*RID* arg0, *int* arg1) const
void	*mesh_remove_surface* (*RID* arg0, *int* arg1)
int	*mesh_get_surface_count* (*RID* arg0) const
RID	*multimesh_create* ( )
void	*multimesh_set_mesh* (*RID* arg0, *RID* arg1)
void	*multimesh_set_aabb* (*RID* arg0, *AABB* arg1)
void	*multimesh_instance_set_transform* (*RID* arg0, *int* arg1, *Transform* arg2)
void	*multimesh_instance_set_color* (*RID* arg0, *int* arg1, *Color* arg2)
RID	*multimesh_get_mesh* (*RID* arg0) const
AABB	*multimesh_get_aabb* (*RID* arg0, *AABB* arg1) const
Transform	*multimesh_instance_get_transform* (*RID* arg0, *int* arg1) const
Color	*multimesh_instance_get_color* (*RID* arg0, *int* arg1) const
RID	*particles_create* ( )
void	*particles_set_amount* (*RID* arg0, *int* arg1)
int	*particles_get_amount* (*RID* arg0) const
void	*particles_set_emitting* (*RID* arg0, *bool* arg1)
bool	*particles_is_emitting* (*RID* arg0) const
void	*particles_set_visibility_aabb* (*RID* arg0, *AABB* arg1)
AABB	*particles_get_visibility_aabb* (*RID* arg0) const
void	*particles_set_variable* (*RID* arg0, *int* arg1, *float* arg2)
float	*particles_get_variable* (*RID* arg0, *int* arg1) const
void	*particles_set_randomness* (*RID* arg0, *int* arg1, *float* arg2)
float	*particles_get_randomness* (*RID* arg0, *int* arg1) const
void	*particles_set_color_phases* (*RID* arg0, *int* arg1)
int	*particles_get_color_phases* (*RID* arg0) const
void	*particles_set_color_phase_pos* (*RID* arg0, *int* arg1, *float* arg2)
float	*particles_get_color_phase_pos* (*RID* arg0, *int* arg1) const
void	*particles_set_color_phase_color* (*RID* arg0, *int* arg1, *Color* arg2)
Color	*particles_get_color_phase_color* (*RID* arg0, *int* arg1) const
void	*particles_set_attractors* (*RID* arg0, *int* arg1)
int	*particles_get_attractors* (*RID* arg0) const
void	*particles_set_attractor_pos* (*RID* arg0, *int* arg1, *Vector3* arg2)
Vector3	*particles_get_attractor_pos* (*RID* arg0, *int* arg1) const
void	*particles_set_attractor_strength* (*RID* arg0, *int* arg1, *float* arg2)
float	*particles_get_attractor_strength* (*RID* arg0, *int* arg1) const

Continúa en la págin

Tabla 9.34 – proviene de la página anterior

void	*particles_set_material* (*RID* arg0, *RID* arg1, *bool* arg2=false)
void	*particles_set_height_from_velocity* (*RID* arg0, *bool* arg1)
bool	*particles_has_height_from_velocity* (*RID* arg0) const
RID	*light_create* (*int* arg0)
int	*light_get_type* (*RID* arg0) const
void	*light_set_color* (*RID* arg0, *int* arg1, *Color* arg2)
Color	*light_get_color* (*RID* arg0, *int* arg1) const
void	*light_set_shadow* (*RID* arg0, *bool* arg1)
bool	*light_has_shadow* (*RID* arg0) const
void	*light_set_volumetric* (*RID* arg0, *bool* arg1)
bool	*light_is_volumetric* (*RID* arg0) const
void	*light_set_projector* (*RID* arg0, *RID* arg1)
RID	*light_get_projector* (*RID* arg0) const
void	*light_set_var* (*RID* arg0, *int* arg1, *float* arg2)
float	*light_get_var* (*RID* arg0, *int* arg1) const
RID	*skeleton_create* ( )
void	*skeleton_resize* (*RID* arg0, *int* arg1)
int	*skeleton_get_bone_count* (*RID* arg0) const
void	*skeleton_bone_set_transform* (*RID* arg0, *int* arg1, *Transform* arg2)
Transform	*skeleton_bone_get_transform* (*RID* arg0, *int* arg1)
RID	*room_create* ( )
void	*room_set_bounds* (*RID* arg0, *Dictionary* arg1)
Dictionary	*room_get_bounds* (*RID* arg0) const
RID	*portal_create* ( )
void	*portal_set_shape* (*RID* arg0, *Vector2Array* arg1)
Vector2Array	*portal_get_shape* (*RID* arg0) const
void	*portal_set_enabled* (*RID* arg0, *bool* arg1)
bool	*portal_is_enabled* (*RID* arg0) const
void	*portal_set_disable_distance* (*RID* arg0, *float* arg1)
float	*portal_get_disable_distance* (*RID* arg0) const
void	*portal_set_disabled_color* (*RID* arg0, *Color* arg1)
Color	*portal_get_disabled_color* (*RID* arg0) const
RID	*camera_create* ( )
void	*camera_set_perspective* (*RID* arg0, *float* arg1, *float* arg2, *float* arg3)
void	*camera_set_orthogonal* (*RID* arg0, *float* arg1, *float* arg2, *float* arg3)
void	*camera_set_transform* (*RID* arg0, *Transform* arg1)
RID	*viewport_create* ( )
void	*viewport_set_rect* (*RID* arg0, *Rect2* arg1)
Rect2	*viewport_get_rect* (*RID* arg0) const
void	*viewport_attach_camera* (*RID* arg0, *RID* arg1=RID())
RID	*viewport_get_attached_camera* (*RID* arg0) const
RID	*viewport_get_scenario* (*RID* arg0) const
void	*viewport_attach_canvas* (*RID* arg0, *RID* arg1)
void	*viewport_remove_canvas* (*RID* arg0, *RID* arg1)
void	*viewport_set_global_canvas_transform* (*RID* arg0, *Matrix32* arg1)
RID	*scenario_create* ( )
void	*scenario_set_debug* (*RID* arg0, *int* arg1)
RID	*instance_create* ( )
RID	*instance_get_base* (*RID* arg0) const
RID	*instance_get_base_aabb* (*RID* arg0) const

Continúa en la págin

Tabla 9.34 – proviene de la página anterior

void	*instance_set_transform* (*RID* arg0, *Transform* arg1)
Transform	*instance_get_transform* (*RID* arg0) const
void	*instance_attach_object_instance_ID* (*RID* arg0, *int* arg1)
int	*instance_get_object_instance_ID* (*RID* arg0) const
void	*instance_attach_skeleton* (*RID* arg0, *RID* arg1)
RID	*instance_get_skeleton* (*RID* arg0) const
void	*instance_set_room* (*RID* arg0, *RID* arg1)
RID	*instance_get_room* (*RID* arg0) const
void	*instance_set_exterior* (*RID* arg0, *bool* arg1)
bool	*instance_is_exterior* (*RID* arg0) const
Array	*instances_cull_aabb* (*AABB* arg0, *RID* arg1) const
Array	*instances_cull_ray* (*Vector3* arg0, *Vector3* arg1, *RID* arg2) const
Array	*instances_cull_convex* (*Array* arg0, *RID* arg1) const
RID	*instance_geometry_override_material_param* (*RID* arg0) const
RID	*instance_geometry_get_material_param* (*RID* arg0) const
RID	*get_test_cube* ( )
RID	*canvas_create* ( )
RID	*canvas_item_create* ( )
void	*canvas_item_set_parent* (*RID* arg0, *RID* arg1)
RID	*canvas_item_get_parent* (*RID* arg0) const
void	*canvas_item_set_transform* (*RID* arg0, *Matrix32* arg1)
void	*canvas_item_set_custom_rect* (*RID* arg0, *bool* arg1, *Rect2* arg2)
void	*canvas_item_set_clip* (*RID* arg0, *bool* arg1)
void	*canvas_item_set_opacity* (*RID* arg0, *float* arg1)
float	*canvas_item_get_opacity* (*RID* arg0, *float* arg1) const
void	*canvas_item_set_self_opacity* (*RID* arg0, *float* arg1)
float	*canvas_item_get_self_opacity* (*RID* arg0, *float* arg1) const
void	*canvas_item_set_z* (*RID* arg0, *int* arg1)
void	*canvas_item_add_line* (*RID* arg0, *Vector2* arg1, *Vector2* arg2, *Color* arg3, *float* arg4=1)
void	*canvas_item_add_rect* (*RID* arg0, *Rect2* arg1, *Color* arg2)
void	*canvas_item_add_texture_rect* (*RID* arg0, *Rect2* arg1, *RID* arg2, *bool* arg3, *Color* arg4=Color(1,1,1,1), *bool* arg5=fal
void	*canvas_item_add_texture_rect_region* (*RID* arg0, *Rect2* arg1, *RID* arg2, *Rect2* arg3, *Color* arg4=Color(1,1,1,1), *bool*
void	*canvas_item_add_style_box* (*RID* arg0, *Rect2* arg1, *RID* arg2, *RealArray* arg3, *Color* arg4=Color(1,1,1,1))
void	*canvas_item_add_circle* (*RID* arg0, *Vector2* arg1, *float* arg2, *Color* arg3)
void	*viewport_set_canvas_transform* (*RID* arg0, *RID* arg1, *Matrix32* arg2)
void	*canvas_item_clear* (*RID* arg0)
void	*canvas_item_raise* (*RID* arg0)
void	*cursor_set_rotation* (*float* arg0, *int* arg1)
void	*cursor_set_texture* (*RID* arg0, *Vector2* arg1, *int* arg2)
void	*cursor_set_visible* (*bool* arg0, *int* arg1)
void	*cursor_set_pos* (*Vector2* arg0, *int* arg1)
void	*black_bars_set_margins* (*int* left, *int* top, *int* right, *int* bottom)
void	*black_bars_set_images* (*RID* left, *RID* top, *RID* right, *RID* bottom)
RID	*make_sphere_mesh* (*int* arg0, *int* arg1, *float* arg2)
void	*mesh_add_surface_from_planes* (*RID* arg0, *Array* arg1)
void	*draw* ( )
void	*sync* ( )
void	*free* (*RID* arg0)
void	*set_default_clear_color* (*Color* arg0)
int	*get_render_info* (*int* arg0)

9.350.3 Numeric Constants

- **NO_INDEX_ARRAY = -1**
- **CUSTOM_ARRAY_SIZE = 8**
- **ARRAY_WEIGHTS_SIZE = 4**
- **MAX_PARTICLE_COLOR_PHASES = 4**
- **MAX_PARTICLE_ATTRACTORS = 4**
- **MAX_CURSORS = 8**
- **TEXTURE_FLAG_MIPMAPS = 1**
- **TEXTURE_FLAG_REPEAT = 2**
- **TEXTURE_FLAG_FILTER = 4**
- **TEXTURE_FLAG_CUBEMAP = 2048**
- **TEXTURE_FLAGS_DEFAULT = 7**
- **CUBEMAP_LEFT = 0**
- **CUBEMAP_RIGHT = 1**
- **CUBEMAP_BOTTOM = 2**
- **CUBEMAP_TOP = 3**
- **CUBEMAP_FRONT = 4**
- **CUBEMAP_BACK = 5**
- **SHADER_MATERIAL = 0**
- **SHADER_POST_PROCESS = 2**
- **MATERIAL_FLAG_VISIBLE = 0**
- **MATERIAL_FLAG_DOUBLE_SIDED = 1**
- **MATERIAL_FLAG_INVERT_FACES = 2**
- **MATERIAL_FLAG_UNSHADED = 3**
- **MATERIAL_FLAG_ONTOP = 4**
- **MATERIAL_FLAG_MAX = 7**
- **MATERIAL_BLEND_MODE_MIX = 0**
- **MATERIAL_BLEND_MODE_ADD = 1**
- **MATERIAL_BLEND_MODE_SUB = 2**
- **MATERIAL_BLEND_MODE_MUL = 3**
- **FIXED_MATERIAL_PARAM_DIFFUSE = 0**
- **FIXED_MATERIAL_PARAM_DETAIL = 1**
- **FIXED_MATERIAL_PARAM_SPECULAR = 2**
- **FIXED_MATERIAL_PARAM_EMISSION = 3**
- **FIXED_MATERIAL_PARAM_SPECULAR_EXP = 4**
- **FIXED_MATERIAL_PARAM_GLOW = 5**

- **FIXED_MATERIAL_PARAM_NORMAL = 6**
- **FIXED_MATERIAL_PARAM_SHADE_PARAM = 7**
- **FIXED_MATERIAL_PARAM_MAX = 8**
- **FIXED_MATERIAL_TEXCOORD_SPHERE = 3**
- **FIXED_MATERIAL_TEXCOORD_UV = 0**
- **FIXED_MATERIAL_TEXCOORD_UV_TRANSFORM = 1**
- **FIXED_MATERIAL_TEXCOORD_UV2 = 2**
- **ARRAY_VERTEX = 0**
- **ARRAY_NORMAL = 1**
- **ARRAY_TANGENT = 2**
- **ARRAY_COLOR = 3**
- **ARRAY_TEX_UV = 4**
- **ARRAY_BONES = 6**
- **ARRAY_WEIGHTS = 7**
- **ARRAY_INDEX = 8**
- **ARRAY_MAX = 9**
- **ARRAY_FORMAT_VERTEX = 1**
- **ARRAY_FORMAT_NORMAL = 2**
- **ARRAY_FORMAT_TANGENT = 4**
- **ARRAY_FORMAT_COLOR = 8**
- **ARRAY_FORMAT_TEX_UV = 16**
- **ARRAY_FORMAT_BONES = 64**
- **ARRAY_FORMAT_WEIGHTS = 128**
- **ARRAY_FORMAT_INDEX = 256**
- **PRIMITIVE_POINTS = 0**
- **PRIMITIVE_LINES = 1**
- **PRIMITIVE_LINE_STRIP = 2**
- **PRIMITIVE_LINE_LOOP = 3**
- **PRIMITIVE_TRIANGLES = 4**
- **PRIMITIVE_TRIANGLE_STRIP = 5**
- **PRIMITIVE_TRIANGLE_FAN = 6**
- **PRIMITIVE_MAX = 7**
- **PARTICLE_LIFETIME = 0**
- **PARTICLE_SPREAD = 1**
- **PARTICLE_GRAVITY = 2**
- **PARTICLE_LINEAR_VELOCITY = 3**

- **PARTICLE_ANGULAR_VELOCITY = 4**
- **PARTICLE_LINEAR_ACCELERATION = 5**
- **PARTICLE_RADIAL_ACCELERATION = 6**
- **PARTICLE_TANGENTIAL_ACCELERATION = 7**
- **PARTICLE_INITIAL_SIZE = 9**
- **PARTICLE_FINAL_SIZE = 10**
- **PARTICLE_INITIAL_ANGLE = 11**
- **PARTICLE_HEIGHT = 12**
- **PARTICLE_HEIGHT_SPEED_SCALE = 13**
- **PARTICLE_VAR_MAX = 14**
- **LIGHT_DIRECTIONAL = 0**
- **LIGHT_OMNI = 1**
- **LIGHT_SPOT = 2**
- **LIGHT_COLOR_DIFFUSE = 0**
- **LIGHT_COLOR_SPECULAR = 1**
- **LIGHT_PARAM_SPOT_ATTENUATION = 0**
- **LIGHT_PARAM_SPOT_ANGLE = 1**
- **LIGHT_PARAM_RADIUS = 2**
- **LIGHT_PARAM_ENERGY = 3**
- **LIGHT_PARAM_ATTENUATION = 4**
- **LIGHT_PARAM_MAX = 10**
- **SCENARIO_DEBUG_DISABLED = 0**
- **SCENARIO_DEBUG_WIREFRAME = 1**
- **SCENARIO_DEBUG_OVERDRAW = 2**
- **INSTANCE_MESH = 1**
- **INSTANCE_MULTIMESH = 2**
- **INSTANCE_PARTICLES = 4**
- **INSTANCE_LIGHT = 5**
- **INSTANCE_ROOM = 6**
- **INSTANCE_PORTAL = 7**
- **INSTANCE_GEOMETRY_MASK = 30**
- **INFO_OBJECTS_IN_FRAME = 0**
- **INFO_VERTICES_IN_FRAME = 1**
- **INFO_MATERIAL_CHANGES_IN_FRAME = 2**
- **INFO_SHADER_CHANGES_IN_FRAME = 3**
- **INFO_SURFACE_CHANGES_IN_FRAME = 4**

- **INFO_DRAW_CALLS_IN_FRAME = 5**
- **INFO_USAGE_VIDEO_MEM_TOTAL = 6**
- **INFO_VIDEO_MEM_USED = 7**
- **INFO_TEXTURE_MEM_USED = 8**
- **INFO_VERTEX_MEM_USED = 9**

9.350.4 Description

Server for anything visible. The visual server is the API backend for everything visible. The whole scene system mounts on it to display.

The visual server is completely opaque, the internals are entirely implementation specific and cannot be accessed.

9.350.5 Member Function Description

- *RID* **texture_create** ( )
- *RID* **texture_create_from_image** (*Image* arg0, *int* arg1=7)
- void **texture_set_flags** (*RID* arg0, *int* arg1)
- *int* **texture_get_flags** (*RID* arg0) const
- *int* **texture_get_width** (*RID* arg0) const
- *int* **texture_get_height** (*RID* arg0) const
- void **texture_set_shrink_all_x2_on_set_data** (*bool* shrink)
- *RID* **shader_create** (*int* mode=0)
- void **shader_set_mode** (*RID* shader, *int* mode)
- *RID* **material_create** ( )
- void **material_set_shader** (*RID* shader, *RID* arg1)
- *RID* **material_get_shader** (*RID* arg0) const
- void **material_set_param** (*RID* arg0, *String* arg1, var arg2)
- void **material_get_param** (*RID* arg0, *String* arg1) const
- void **material_set_flag** (*RID* arg0, *int* arg1, *bool* arg2)
- *bool* **material_get_flag** (*RID* arg0, *int* arg1) const
- void **material_set_blend_mode** (*RID* arg0, *int* arg1)
- *int* **material_get_blend_mode** (*RID* arg0) const
- void **material_set_line_width** (*RID* arg0, *float* arg1)
- *float* **material_get_line_width** (*RID* arg0) const
- *RID* **mesh_create** ( )
- void **mesh_add_surface** (*RID* arg0, *int* arg1, *Array* arg2, *Array* arg3, *bool* arg4=-1)
- void **mesh_surface_set_material** (*RID* arg0, *int* arg1, *RID* arg2, *bool* arg3=false)
- *RID* **mesh_surface_get_material** (*RID* arg0, *int* arg1) const

- *int* **mesh_surface_get_array_len** (*RID* arg0, *int* arg1) const
- *int* **mesh_surface_get_array_index_len** (*RID* arg0, *int* arg1) const
- *int* **mesh_surface_get_format** (*RID* arg0, *int* arg1) const
- *int* **mesh_surface_get_primitive_type** (*RID* arg0, *int* arg1) const
- void **mesh_remove_surface** (*RID* arg0, *int* arg1)
- *int* **mesh_get_surface_count** (*RID* arg0) const
- *RID* **multimesh_create** ( )
- void **multimesh_set_mesh** (*RID* arg0, *RID* arg1)
- void **multimesh_set_aabb** (*RID* arg0, *AABB* arg1)
- void **multimesh_instance_set_transform** (*RID* arg0, *int* arg1, *Transform* arg2)
- void **multimesh_instance_set_color** (*RID* arg0, *int* arg1, *Color* arg2)
- *RID* **multimesh_get_mesh** (*RID* arg0) const
- *AABB* **multimesh_get_aabb** (*RID* arg0, *AABB* arg1) const
- *Transform* **multimesh_instance_get_transform** (*RID* arg0, *int* arg1) const
- *Color* **multimesh_instance_get_color** (*RID* arg0, *int* arg1) const
- *RID* **particles_create** ( )
- void **particles_set_amount** (*RID* arg0, *int* arg1)
- *int* **particles_get_amount** (*RID* arg0) const
- void **particles_set_emitting** (*RID* arg0, *bool* arg1)
- *bool* **particles_is_emitting** (*RID* arg0) const
- void **particles_set_visibility_aabb** (*RID* arg0, *AABB* arg1)
- *AABB* **particles_get_visibility_aabb** (*RID* arg0) const
- void **particles_set_variable** (*RID* arg0, *int* arg1, *float* arg2)
- *float* **particles_get_variable** (*RID* arg0, *int* arg1) const
- void **particles_set_randomness** (*RID* arg0, *int* arg1, *float* arg2)
- *float* **particles_get_randomness** (*RID* arg0, *int* arg1) const
- void **particles_set_color_phases** (*RID* arg0, *int* arg1)
- *int* **particles_get_color_phases** (*RID* arg0) const
- void **particles_set_color_phase_pos** (*RID* arg0, *int* arg1, *float* arg2)
- *float* **particles_get_color_phase_pos** (*RID* arg0, *int* arg1) const
- void **particles_set_color_phase_color** (*RID* arg0, *int* arg1, *Color* arg2)
- *Color* **particles_get_color_phase_color** (*RID* arg0, *int* arg1) const
- void **particles_set_attractors** (*RID* arg0, *int* arg1)
- *int* **particles_get_attractors** (*RID* arg0) const
- void **particles_set_attractor_pos** (*RID* arg0, *int* arg1, *Vector3* arg2)
- *Vector3* **particles_get_attractor_pos** (*RID* arg0, *int* arg1) const

- void **particles_set_attractor_strength** (*RID* arg0, *int* arg1, *float* arg2)
- *float* **particles_get_attractor_strength** (*RID* arg0, *int* arg1) const
- void **particles_set_material** (*RID* arg0, *RID* arg1, *bool* arg2=false)
- void **particles_set_height_from_velocity** (*RID* arg0, *bool* arg1)
- *bool* **particles_has_height_from_velocity** (*RID* arg0) const
- *RID* **light_create** (*int* arg0)
- *int* **light_get_type** (*RID* arg0) const
- void **light_set_color** (*RID* arg0, *int* arg1, *Color* arg2)
- *Color* **light_get_color** (*RID* arg0, *int* arg1) const
- void **light_set_shadow** (*RID* arg0, *bool* arg1)
- *bool* **light_has_shadow** (*RID* arg0) const
- void **light_set_volumetric** (*RID* arg0, *bool* arg1)
- *bool* **light_is_volumetric** (*RID* arg0) const
- void **light_set_projector** (*RID* arg0, *RID* arg1)
- *RID* **light_get_projector** (*RID* arg0) const
- void **light_set_var** (*RID* arg0, *int* arg1, *float* arg2)
- *float* **light_get_var** (*RID* arg0, *int* arg1) const
- *RID* **skeleton_create** ( )
- void **skeleton_resize** (*RID* arg0, *int* arg1)
- *int* **skeleton_get_bone_count** (*RID* arg0) const
- void **skeleton_bone_set_transform** (*RID* arg0, *int* arg1, *Transform* arg2)
- *Transform* **skeleton_bone_get_transform** (*RID* arg0, *int* arg1)
- *RID* **room_create** ( )
- void **room_set_bounds** (*RID* arg0, *Dictionary* arg1)
- *Dictionary* **room_get_bounds** (*RID* arg0) const
- *RID* **portal_create** ( )
- void **portal_set_shape** (*RID* arg0, *Vector2Array* arg1)
- *Vector2Array* **portal_get_shape** (*RID* arg0) const
- void **portal_set_enabled** (*RID* arg0, *bool* arg1)
- *bool* **portal_is_enabled** (*RID* arg0) const
- void **portal_set_disable_distance** (*RID* arg0, *float* arg1)
- *float* **portal_get_disable_distance** (*RID* arg0) const
- void **portal_set_disabled_color** (*RID* arg0, *Color* arg1)
- *Color* **portal_get_disabled_color** (*RID* arg0) const
- *RID* **camera_create** ( )
- void **camera_set_perspective** (*RID* arg0, *float* arg1, *float* arg2, *float* arg3)

- void **camera_set_orthogonal** (*RID* arg0, *float* arg1, *float* arg2, *float* arg3)
- void **camera_set_transform** (*RID* arg0, *Transform* arg1)
- *RID* **viewport_create** ( )
- void **viewport_set_rect** (*RID* arg0, *Rect2* arg1)
- *Rect2* **viewport_get_rect** (*RID* arg0) const
- void **viewport_attach_camera** (*RID* arg0, *RID* arg1=RID())
- *RID* **viewport_get_attached_camera** (*RID* arg0) const
- *RID* **viewport_get_scenario** (*RID* arg0) const
- void **viewport_attach_canvas** (*RID* arg0, *RID* arg1)
- void **viewport_remove_canvas** (*RID* arg0, *RID* arg1)
- void **viewport_set_global_canvas_transform** (*RID* arg0, *Matrix32* arg1)
- *RID* **scenario_create** ( )
- void **scenario_set_debug** (*RID* arg0, *int* arg1)
- *RID* **instance_create** ( )
- *RID* **instance_get_base** (*RID* arg0) const
- *RID* **instance_get_base_aabb** (*RID* arg0) const
- void **instance_set_transform** (*RID* arg0, *Transform* arg1)
- *Transform* **instance_get_transform** (*RID* arg0) const
- void **instance_attach_object_instance_ID** (*RID* arg0, *int* arg1)
- *int* **instance_get_object_instance_ID** (*RID* arg0) const
- void **instance_attach_skeleton** (*RID* arg0, *RID* arg1)
- *RID* **instance_get_skeleton** (*RID* arg0) const
- void **instance_set_room** (*RID* arg0, *RID* arg1)
- *RID* **instance_get_room** (*RID* arg0) const
- void **instance_set_exterior** (*RID* arg0, *bool* arg1)
- *bool* **instance_is_exterior** (*RID* arg0) const
- *Array* **instances_cull_aabb** (*AABB* arg0, *RID* arg1) const
- *Array* **instances_cull_ray** (*Vector3* arg0, *Vector3* arg1, *RID* arg2) const
- *Array* **instances_cull_convex** (*Array* arg0, *RID* arg1) const
- *RID* **instance_geometry_override_material_param** (*RID* arg0) const
- *RID* **instance_geometry_get_material_param** (*RID* arg0) const
- *RID* **get_test_cube** ( )
- *RID* **canvas_create** ( )
- *RID* **canvas_item_create** ( )
- void **canvas_item_set_parent** (*RID* arg0, *RID* arg1)
- *RID* **canvas_item_get_parent** (*RID* arg0) const

- void **canvas_item_set_transform** (*RID* arg0, *Matrix32* arg1)
- void **canvas_item_set_custom_rect** (*RID* arg0, *bool* arg1, *Rect2* arg2)
- void **canvas_item_set_clip** (*RID* arg0, *bool* arg1)
- void **canvas_item_set_opacity** (*RID* arg0, *float* arg1)
- *float* **canvas_item_get_opacity** (*RID* arg0, *float* arg1) const
- void **canvas_item_set_self_opacity** (*RID* arg0, *float* arg1)
- *float* **canvas_item_get_self_opacity** (*RID* arg0, *float* arg1) const
- void **canvas_item_set_z** (*RID* arg0, *int* arg1)
- void **canvas_item_add_line** (*RID* arg0, *Vector2* arg1, *Vector2* arg2, *Color* arg3, *float* arg4=1)
- void **canvas_item_add_rect** (*RID* arg0, *Rect2* arg1, *Color* arg2)
- void **canvas_item_add_texture_rect** (*RID* arg0, *Rect2* arg1, *RID* arg2, *bool* arg3, *Color* arg4=Color(1,1,1,1), *bool* arg5=false)
- void **canvas_item_add_texture_rect_region** (*RID* arg0, *Rect2* arg1, *RID* arg2, *Rect2* arg3, *Color* arg4=Color(1,1,1,1), *bool* arg5=false)
- void **canvas_item_add_style_box** (*RID* arg0, *Rect2* arg1, *RID* arg2, *RealArray* arg3, *Color* arg4=Color(1,1,1,1))
- void **canvas_item_add_circle** (*RID* arg0, *Vector2* arg1, *float* arg2, *Color* arg3)
- void **viewport_set_canvas_transform** (*RID* arg0, *RID* arg1, *Matrix32* arg2)
- void **canvas_item_clear** (*RID* arg0)
- void **canvas_item_raise** (*RID* arg0)
- void **cursor_set_rotation** (*float* arg0, *int* arg1)
- void **cursor_set_texture** (*RID* arg0, *Vector2* arg1, *int* arg2)
- void **cursor_set_visible** (*bool* arg0, *int* arg1)
- void **cursor_set_pos** (*Vector2* arg0, *int* arg1)
- void **black_bars_set_margins** (*int* left, *int* top, *int* right, *int* bottom)
- void **black_bars_set_images** (*RID* left, *RID* top, *RID* right, *RID* bottom)
- *RID* **make_sphere_mesh** (*int* arg0, *int* arg1, *float* arg2)
- void **mesh_add_surface_from_planes** (*RID* arg0, *Array* arg1)
- void **draw** ( )
- void **sync** ( )
- void **free** (*RID* arg0)
- void **set_default_clear_color** (*Color* arg0)
- *int* **get_render_info** (*int* arg0)

9.351 VScrollBar

Inherits: *ScrollBar* < *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.351.1 Brief Description

Vertical version of *ScrollBar*, which goes from left (min) to right (max).

9.352 VSeparator

Inherits: *Separator* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.352.1 Brief Description

Vertical version of *Separator*.

9.352.2 Description

Vertical version of *Separator*. It is used to separate objects horizontally, though (but it looks vertical!).

9.353 VSlider

Inherits: *Slider* < *Range* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.353.1 Brief Description

Vertical slider.

9.353.2 Description

Vertical slider. See *Slider*. This one goes from left (min) to right (max).

9.354 VSplitContainer

Inherits: *SplitContainer* < *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Category: Core

9.354.1 Brief Description

Vertical split container.

9.354.2 Description

Vertical split container. See *SplitContainer*. This goes from left to right.

9.355 WeakRef

Inherits: *Reference* < *Object*

Category: Core

9.355.1 Brief Description

Holds an *Object*, but does not contribute to the reference count if the object is a reference.

9.355.2 Member Functions

Object	*get_ref* ( ) const

9.355.3 Description

A weakref can hold a *Reference*, without contributing to the reference counter. A weakref can be created from an *Object* using *@GDScript.weakref*. If this object is not a reference, weakref still works, however, it does not have any effect on the object. Weakrefs are useful in cases where multiple classes have variables that refer to eachother. Without weakrefs, using these classes could lead to memory leaks, since both references keep eachother from being released. Making part of the variables a weakref can prevent this cyclic dependency, and allows the references to be released.

9.355.4 Member Function Description

- *Object* **get_ref** ( ) const

Returns the *Object* this weakref is referring to.

9.356 WindowDialog

Inherits: *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *AcceptDialog*

Category: Core

9.356.1 Brief Description

Base class for window dialogs.

9.356.2 Member Functions

void	*set_title* (*String* title)
String	*get_title* ( ) const
TextureButton	*get_close_button* ( )

9.356.3 Description

Windowdialog is the base class for all window-based dialogs. It's a by-default toplevel *Control* that draws a window decoration and allows motion and resizing.

9.356.4 Member Function Description

- void **set_title** (*String* title)

Set the title of the window.

- *String* **get_title** ( ) const

Return the title of the window.

- *TextureButton* **get_close_button** ( )

Return the close *TextureButton*.

9.357 World

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.357.1 Brief Description

Class that has everything pertaining to a world.

9.357.2 Member Functions

RID	*get_space* ( ) const
RID	*get_scenario* ( ) const
RID	*get_sound_space* ( ) const
void	*set_environment* (*Environment* env)
Environment	*get_environment* ( ) const
PhysicsDirectSpaceState	*get_direct_space_state* ( )

9.357.3 Description

Class that has everything pertaining to a world. A physics space, a visual scenario and a sound space. Spatial nodes register their resources into the current world.

9.357.4 Member Function Description

- *RID* **get_space** ( ) const
- *RID* **get_scenario** ( ) const
- *RID* **get_sound_space** ( ) const
- void **set_environment** (*Environment* env)
- *Environment* **get_environment** ( ) const
- *PhysicsDirectSpaceState* **get_direct_space_state** ( )

9.358 World2D

Inherits: *Resource* < *Reference* < *Object*

Category: Core

9.358.1 Brief Description

Class that has everything pertaining to a 2D world.

9.358.2 Member Functions

RID	*get_canvas* ( )
RID	*get_space* ( )
RID	*get_sound_space* ( )
Physics2DDirectSpaceState	*get_direct_space_state* ( )

9.358.3 Description

Class that has everything pertaining to a 2D world. A physics space, a visual scenario and a sound space. 2D nodes register their resources into the current 2D world.

9.358.4 Member Function Description

- *RID* **get_canvas** ( )
- *RID* **get_space** ( )
- *RID* **get_sound_space** ( )
- *Physics2DDirectSpaceState* **get_direct_space_state** ( )

9.359 WorldEnvironment

Inherits: *Spatial* < *Node* < *Object*

Category: Core

9.359.1 Brief Description

9.359.2 Member Functions

void	*set_environment* (*Environment* env)
Environment	*get_environment* ( ) const

9.359.3 Member Function Description

- void **set_environment** (*Environment* env)
- *Environment* **get_environment** ( ) const

9.360 XMLParser

Inherits: *Reference* < *Object*

Category: Core

9.360.1 Brief Description

9.360.2 Member Functions

int	*read* ( )
int	*get_node_type* ( )
String	*get_node_name* ( ) const
String	*get_node_data* ( ) const
int	*get_node_offset* ( ) const
int	*get_attribute_count* ( ) const
String	*get_attribute_name* (*int* idx) const
String	*get_attribute_value* (*int* idx) const
bool	*has_attribute* (*String* name) const
String	*get_named_attribute_value* (*String* name) const
String	*get_named_attribute_value_safe* (*String* name) const
bool	*is_empty* ( ) const
int	*get_current_line* ( ) const
void	*skip_section* ( )
int	*seek* (*int* pos)
int	*open* (*String* file)
int	*open_buffer* (*RawArray* buffer)

9.360.3 Numeric Constants

- **NODE_NONE = 0**
- **NODE_ELEMENT = 1**
- **NODE_ELEMENT_END = 2**
- **NODE_TEXT = 3**

- **NODE_COMMENT = 4**
- **NODE_CDATA = 5**
- **NODE_UNKNOWN = 6**

9.360.4 Member Function Description

- *int* **read** ( )
- *int* **get_node_type** ( )
- *String* **get_node_name** ( ) const
- *String* **get_node_data** ( ) const
- *int* **get_node_offset** ( ) const
- *int* **get_attribute_count** ( ) const
- *String* **get_attribute_name** (*int* idx) const
- *String* **get_attribute_value** (*int* idx) const
- *bool* **has_attribute** (*String* name) const
- *String* **get_named_attribute_value** (*String* name) const
- *String* **get_named_attribute_value_safe** (*String* name) const
- *bool* **is_empty** ( ) const
- *int* **get_current_line** ( ) const
- void **skip_section** ( )
- *int* **seek** (*int* pos)
- *int* **open** (*String* file)
- *int* **open_buffer** (*RawArray* buffer)

9.361 YSort

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

9.361.1 Brief Description

Sort all child nodes based on their Y positions.

9.361.2 Member Functions

void	*set_sort_enabled* (*bool* enabled)
bool	*is_sort_enabled* ( ) const

9.361.3 Description

Sort all child nodes based on their Y positions. The child node must inherit from *CanvasItem* for it to be sorted. Nodes that have a higher Y position will be drawn later, so they will appear on top of nodes that have a lower Y position.

9.361.4 Member Function Description

- void **set_sort_enabled** (*bool* enabled)

Set whether the children nodes are sorted or not. (default true)

- *bool* **is_sort_enabled** ( ) const

Returns true if the children nodes are being sorted.

CAPÍTULO 10

Languages

10.1 GDScript

10.1.1 Introduction

GDScript is a high level, dynamically typed programming language used to create content. It uses a syntax similar to Python (blocks are indent-based and many keywords are similar). Its goal is to be optimized for and tightly integrated with Godot Engine, allowing great flexibility for content creation and integration.

History

Initially, Godot was designed to support multiple scripting languages (this ability still exists today). However, only GDScript is in use right now. There is a little history behind this.

In the early days, the engine used the Lua scripting language. Lua is fast, but creating bindings to an object oriented system (by using fallbacks) was complex and slow and took an enormous amount of code. After some experiments with Python, it also proved difficult to embed.

The last third party scripting language that was used for shipped games was Squirrel, but it was dropped as well. At that point, it became evident that a custom scripting language could more optimally make use of Godot's particular architecture:

- Godot embeds scripts in nodes. Most languages are not designed with this in mind.
- Godot uses several built-in data types for 2D and 3D math. Script languages do not provide this, and binding them is inefficient.
- Godot uses threads heavily for lifting and initializing data from the net or disk. Script interpreters for common languages are not friendly to this.
- Godot already has a memory management model for resources, most script languages provide their own, which results in duplicate effort and bugs.
- Binding code is always messy and results in several failure points, unexpected bugs and generally low maintainability.

The result of these considerations is *GDScript*. The language and interpreter for GDScript ended up being smaller than the binding code itself for Lua and Squirrel, while having equal functionality. With time, having a built-in language has proven to be a huge advantage.

Example of GDScript

Some people can learn better by just taking a look at the syntax, so here's a simple example of how GDScript looks.

```
# a file is a class!

# inheritance

extends BaseClass

# member variables

var a = 5
var s = "Hello"
var arr = [1, 2, 3]
var dict = {"key":"value", 2:3}

# constants

const answer = 42
const thename = "Charly"

# built-in vector types

var v2 = Vector2(1, 2)
var v3 = Vector3(1, 2, 3)

# function

func some_function(param1, param2):
    var local_var = 5

    if param1 < local_var:
        print(param1)
    elif param2 > 5:
        print(param2)
    else:
        print("fail!")

    for i in range(20):
        print(i)

    while(param2 != 0):
        param2 -= 1

    var local_var2 = param1+3
    return local_var2


# inner class

class Something:
    var a = 10
```

```
# constructor

func _init():
    print("constructed!")
    var lv = Something.new()
    print(lv.a)
```

If you have previous experience with statically typed languages such as C, C++, or C# but never used a dynamically typed one before, it is advised you read this tutorial: *GDScript more efficiently*.

10.1.2 Language

In the following, an overview is given to GDScript. Details, such as which methods are available to arrays or other objects, should be looked up in the linked class descriptions.

Identifiers

Any string that restricts itself to alphabetic characters (`a` to `z` and `A` to `Z`), digits (`0` to `9`) and _ qualifies as an identifier. Additionally, identifiers must not begin with a digit. Identifiers are case-sensitive (`foo` is different from `FOO`).

Keywords

The following is the list of keywords supported by the language. Since keywords are reserved words (tokens), they can't be used as identifiers.

Keyword	Description
if	See *if/else/elif*.
elif	See *if/else/elif*.
else	See *if/else/elif*.
for	See *for*.
do	Reserved for future implementation of do...while loops.
while	See *while*.
switch	Reserved for future implementation.
case	Reserved for future implementation.
break	Exits the execution of the current `for` or `while` loop.
continue	Immediately skips to the next iteration of the `for` or `while` loop.
pass	Used where a statement is required syntactically but execution of code is undesired, e.g. in empty functions.
return	Returns a value from a function.
class	Defines a class.
extends	Defines what class to extend with the current class. Also tests whether a variable extends a given class.
tool	Executes the script in the editor.
signal	Defines a signal.
func	Defines a function.
static	Defines a static function. Static member variables are not allowed.
const	Defines a constant.
var	Defines a variable.
onready	Initializes a variable once the Node the script is attached to and its children are part of the scene tree.
export	Saves a variable along with the resource it's attached to and makes it visible and modifiable in the editor.
setget	Defines setter and getter functions for a variable.
breakpoint	Editor helper for debugger breakpoints.

Operators

The following is the list of supported operators and their precedence (TODO, change since this was made to reflect python operators)

Operator	Description
`x[index]`	Subscription, Highest Priority
`x.attribute`	Attribute Reference
`extends`	Instance Type Checker
`~`	Bitwise NOT
`-x`	Negative
`* / %`	Multiplication / Division / Remainder
`+ -`	Addition / Subtraction
`<< >>`	Bit Shifting
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`< > == != >= <=`	Comparisons
`in`	Content Test
`! not`	Boolean NOT
`and &&`	Boolean AND
`or ||`	Boolean OR
`= += -= *= /= %= &= |=`	Assignment, Lowest Priority

Literals

Literal	Type
`45`	Base 10 integer
`0x8F51`	Base 16 (hex) integer
`3.14, 58.1e-10`	Floating point number (real)
`"Hello", "Hi"`	Strings
`"""Hello, Dude"""`	Multiline string
`@"Node/Label"`	NodePath or StringName

Comments

Anything from a # to the end of the line is ignored and is considered a comment.

```
# This is a comment
```

Multi-line comments can be created using """ (three quotes in a row) at the beginning and end of a block of text.

```
""" Everything on these
lines is considered
a comment """
```

10.1.3 Built-in types

Basic built-in types

A variable in GDScript can be assigned to several built-in types.

null

`null` is an empty data type that contains no information and can not be assigned any other value.

bool

The Boolean data type can only contain `true` or `false`.

int

The integer data type can only contain integer numbers, (both negative and positive).

float

Used to contain a floating point value (real numbers).

String

A sequence of characters in Unicode format. Strings can contain the standard C escape sequences.

Vector built-in types

Vector2

2D vector type containing `x` and `y` fields. Can alternatively access fields as `width` and `height` for readability. Can also be accessed as array.

Rect2

2D Rectangle type containing two vectors fields: `pos` and `size`. Alternatively contains an `end` field which is `pos+size`.

Vector3

3D vector type containing `x`, `y` and `z` fields. This can also be accessed as an array.

Matrix32

3x2 matrix used for 2D transforms.

Plane

3D Plane type in normalized form that contains a `normal` vector field and a `d` scalar distance.

Quat

Quaternion is a datatype used for representing a 3D rotation. It's useful for interpolating rotations.

AABB

Axis Aligned bounding box (or 3D box) contains 2 vectors fields: `pos` and `size`. Alternatively contains an `end` field which is `pos+size`. As an alias of this type, `Rect3` can be used interchangeably.

Matrix3

3x3 matrix used for 3D rotation and scale. It contains 3 vector fields (`x`, `y` and `z`) and can also be accessed as an array of 3D vectors.

Transform

3D Transform contains a Matrix3 field `basis` and a Vector3 field `origin`.

Engine built-in types

Color

Color data type contains `r`, `g`, `b`, and `a` fields. It can also be accessed as `h`, `s`, and `v` for hue/saturation/value.

Image

Contains a custom format 2D image and allows direct access to the pixels.

NodePath

Compiled path to a node used mainly in the scene system. It can be easily assigned to, and from, a String.

RID

Resource ID (RID). Servers use generic RIDs to reference opaque data.

Object

Base class for anything that is not a built-in type.

InputEvent

Events from input devices are contained in very compact form in InputEvent objects. Due to the fact that they can be received in high amounts from frame to frame they are optimized as their own data type.

Container built-in types

Array

Generic sequence of arbitrary object types, including other arrays or dictionaries (see below). The array can resize dynamically. Arrays are indexed starting from index `0`.

```
var arr=[]
arr=[1, 2, 3]
var b = arr[1] # this is 2
arr[0] = "Hi!" # replacing value 1 with "Hi"
arr.append(4)  # array is now ["Hi", 2, 3, 4]
```

GDScript arrays are allocated linearly in memory for speed. Very large arrays (more than tens of thousands of elements) may however cause memory fragmentation. If this is a concern special types of arrays are available. These only accept a single data type. They avoid memory fragmentation and also use less memory but are atomic and tend to run slower than generic arrays. They are therefore only recommended to use for very large data sets:

- ByteArray: An array of bytes (integers from 0 to 255).
- *IntArray*: An array of integers.
- FloatArray: An array of floats.
- *StringArray*: An array strings.
- *Vector2Array*: An array of *Vector2* objects.
- *Vector3Array*: An array of *Vector3* objects.
- *ColorArray*: An array of *Color* objects.

Dictionary

Associative container which contains values referenced by unique keys.

```
var d={4:5, "a key":"a value", 28:[1,2,3]}
d["Hi!"] = 0
var d = {
    22          : "Value",
    "somekey"   : 2,
    "otherkey" : [2,3,4],
    "morekey"   : "Hello"
}
```

Lua-style table syntax is also supported. Lua-style uses = instead of `:` and doesn't use quotes to mark string keys (making for slightly less to write). Note however that like any GDScript identifier, keys written in this form cannot start with a digit.

```
var d = {
    test22 = "Value",
    somekey = 2,
    otherkey = [2,3,4],
    morekey = "Hello"
}
```

To add a key to an existing dictionary, access it like an existing key and assign to it:

```
var d = {} # create an empty Dictionary
d.Waiting = 14 # add String "Waiting" as a key and assign the value 14 to it
d[4] = "hello" # add integer `4` as a key and assign the String "hello" as its value
d["Godot"] = 3.01 # add String "Godot" as a key and assign the value 3.01 to it
```

10.1.4 Data

Variables

Variables can exist as class members or local to functions. They are created with the `var` keyword and may, optionally, be assigned a value upon initialization.

```
var a  # data type is null by default
var b = 5
var c = 3.8
var d = b + c  # variables are always initialized in order
```

Constants

Constants are similar to variables, but must be constants or constant expressions and must be assigned on initialization.

```
const a = 5
const b = Vector2(20, 20)
const c = 10 + 20 # constant expression
const d = Vector2(20, 30).x  # constant expression: 20
const e = [1, 2, 3, 4][0]  # constant expression: 1
const f = sin(20)  # sin() can be used in constant expressions
const g = x + 20  # invalid; this is not a constant expression!
```

Functions

Functions always belong to a *class*. The scope priority for variable look-up is: local → class member → global. The `self` variable is always available and is provided as an option for accessing class members, but is not always required (and should *not* be sent as the function's first argument, unlike Python).

```
func myfunction(a, b):
    print(a)
    print(b)
    return a + b  # return is optional; without it null is returned
```

A function can `return` at any point. The default return value is `null`.

Referencing Functions

To call a function in a *base class* (i.e. one `extend`-ed in your current class), prepend `.` to the function name:

```
.basefunc(args)
```

Contrary to Python, functions are *not* first class objects in GDScript. This means they cannot be stored in variables, passed as an argument to another function or be returned from other functions. This is for performance reasons.

To reference a function by name at runtime, (e.g. to store it in a variable, or pass it to another function as an argument) one must use the `call` or `funcref` helpers:

```
# Call a function by name in one step
mynode.call("myfunction", args)

# Store a function reference
var myfunc = funcref(mynode, "myfunction")
# Call stored function reference
myfunc.call_func(args)
```

Remember that default functions like `_init`, and most notifications such as `_enter_tree`, `_exit_tree`, `_process`, `_fixed_process`, etc. are called in all base classes automatically. So there is only a need to call the function explicitly when overloading them in some way.

Static functions

A function can be declared static. When a function is static it has no access to the instance member variables or `self`. This is mainly useful to make libraries of helper functions:

```
static func sum2(a, b):
    return a + b
```

Statements and control flow

Statements are standard and can be assignments, function calls, control flow structures, etc (see below). `;` as a statement separator is entirely optional.

if/else/elif

Simple conditions are created by using the `if`/`else`/`elif` syntax. Parenthesis around conditions are allowed, but not required. Given the nature of the tab-based indentation, `elif` can be used instead of `else`/`if` to maintain a level of indentation.

```
if [expression]:
    statement(s)
elif [expression]:
    statement(s)
else:
    statement(s)
```

Short statements can be written on the same line as the condition:

```
if (1 + 1 == 2): return 2 + 2
else:
    var x = 3 + 3
    return x
```

while

Simple loops are created by using `while` syntax. Loops can be broken using `break` or continued using `continue`:

```
while [expression]:
    statement(s)
```

for

To iterate through a range, such as an array or table, a *for* loop is used. When iterating over an array, the current array element is stored in the loop variable. When iterating over a dictionary, the *index* is stored in the loop variable.

```
for x in [5, 7, 11]:
    statement  # loop iterates 3 times with x as 5, then 7 and finally 11

var dict = {"a":0, "b":1, "c":2}
for i in dict:
    print(dict[i])  # loop provides the keys in an arbitrary order; may print 0, 1, 2,
↪ or 2, 0, 1, etc...

for i in range(3):
    statement  # similar to [0, 1, 2] but does not allocate an array

for i in range(1,3):
    statement  # similar to [1, 2] but does not allocate an array

for i in range(2,8,2):
    statement  # similar to [2, 4, 6] but does not allocate an array
```

Classes

By default, the body of a script file is an unnamed class and it can only be referenced externally as a resource or file. Class syntax is meant to be very compact and can only contain member variables or functions. Static functions are allowed, but not static members (this is in the spirit of thread safety, since scripts can be initialized in separate threads without the user knowing). In the same way, member variables (including arrays and dictionaries) are initialized every time an instance is created.

Below is an example of a class file.

```
# saved as a file named myclass.gd

var a = 5

func print_value_of_a():
    print(a)
```

Inheritance

A class (stored as a file) can inherit from

- A global class
- Another class file
- An inner class inside another class file.

Multiple inheritance is not allowed.

Inheritance uses the `extends` keyword:

```
# Inherit/extend a globally available class
extends SomeClass

# Inherit/extend a named class file
extends "somefile.gd"

# Inherit/extend an inner class in another file
extends "somefile.gd".SomeInnerClass
```

To check if a given instance inherits from a given class the `extends` keyword can be used as an operator instead:

```
# Cache the enemy class
const enemy_class = preload("enemy.gd")

# [...]

# use 'extends' to check inheritance
if (entity extends enemy_class):
    entity.apply_damage()
```

Class Constructor

The class constructor, called on class instantiation, is named `_init`. As mentioned earlier, the constructors of parent classes are called automatically when inheriting a class. So there is usually no need to call `._init()` explicitly.

If a parent constructor takes arguments, they are passed like this:

```
func _init(args).(parent_args):
   pass
```

Inner classes

A class file can contain inner classes. Inner classes are defined using the `class` keyword. They are instanced using the `ClassName.new()` function.

```
# inside a class file

# An inner class in this class file
class SomeInnerClass:
    var a = 5
    func print_value_of_a():
        print(a)

# This is the constructor of the class file's main class
func _init():
    var c = SomeInnerClass.new()
    c.print_value_of_a()
```

Classes as resources

Classes stored as files are treated as *resources*. They must be loaded from disk to access them in other classes. This is done using either the `load` or `preload` functions (see below). Instancing of a loaded class resource is done by calling the `new` function on the class object:

```
# Load the class resource when calling load()
var MyClass = load("myclass.gd")

# Preload the class only once at compile time
var MyClass2 = preload("myclass.gd")

func _init():
    var a = MyClass.new()
    a.somefunction()
```

Exports

Class members can be exported. This means their value gets saved along with the resource (e.g. the *scene*) they're attached to. They will also be available for editing in the property editor. Exporting is done by using the `export` keyword:

```
extends Button

export var number = 5  # value will be saved and visible in the property editor
```

An exported variable must be initialized to a constant expression or have an export hint in the form of an argument to the export keyword (see below).

One of the fundamental benefits of exporting member variables is to have them visible and editable in the editor. This way artists and game designers can modify values that later influence how the program runs. For this, a special export syntax is provided.

```
# If the exported value assigns a constant or constant expression,
# the type will be inferred and used in the editor

export var number = 5

# Export can take a basic data type as an argument which will be
# used in the editor

export(int) var number

# Export can also take a resource type to use as a hint

export(Texture) var character_face

# Integers and strings hint enumerated values

# Editor will enumerate as 0, 1 and 2
export(int, "Warrior", "Magician", "Thief") var character_class
# Editor will enumerate with string names
export(String, "Rebecca", "Mary", "Leah") var character_name

# Strings as paths

# String is a path to a file
export(String, FILE) var f
# String is a path to a directory
export(String, DIR) var f
# String is a path to a file, custom filter provided as hint
export(String, FILE, "*.txt") var f
```

```
# Using paths in the global filesystem is also possible,
# but only in tool scripts (see further below)

# String is a path to a PNG file in the global filesystem
export(String, FILE, GLOBAL, "*.png") var tool_image
# String is a path to a directory in the global filesystem
export(String, DIR, GLOBAL) var tool_dir

# The MULTILINE setting tells the editor to show a large input
# field for editing over multiple lines
export(String, MULTILINE) var text

# Limiting editor input ranges

# Allow integer values from 0 to 20
export(int, 20) var i
# Allow integer values from -10 to 20
export(int, -10, 20) var j
# Allow floats from -10 to 20, with a step of 0.2
export(float, -10, 20, 0.2) var k
# Allow values y = exp(x) where y varies betwee 100 and 1000
# while snapping to steps of 20. The editor will present a
# slider for easily editing the value.
export(float, EXP, 100, 1000, 20) var l

# Floats with easing hint

# Display a visual representation of the ease() function
# when editing
export(float, EASE) var transition_speed

# Colors

# Color given as Red-Green-Blue value
export(Color, RGB) var col  # Color is RGB
# Color given as Red-Green-Blue-Alpha value
export(Color, RGBA) var col  # Color is RGBA

# another node in the scene can be exported too

export(NodePath) var node
```

It must be noted that even if the script is not being run while at the editor, the exported properties are still editable (see below for "tool").

Exporting bit flags

Integers used as bit flags can store multiple `true`/`false` (boolean) values in one property. By using the export hint `int, FLAGS`, they can be set from the editor:

```
# Individually edit the bits of an integer
export(int, FLAGS) var spell_elements = ELEMENT_WIND | ELEMENT_WATER
```

Restricting the flags to a certain number of named flags is also possible. The syntax is very similar to the enumeration syntax:

```
# Set any of the given flags from the editor
export(int, FLAGS, "Fire", "Water", "Earth", "Wind") var spell_elements = 0
```

In this example, `Fire` has value 1, `Water` has value 2, `Earth` has value 4 and `Wind` corresponds to value 8. Usually, constants should be defined accordingly (e.g. `const ELEMENT_WIND = 8` and so on).

Using bit flags requires some understanding of bitwise operations. If in doubt, boolean variables should be exported instead.

Exporting arrays

Exporting arrays works but with an important caveat: While regular arrays are created local to every class instance, exported arrays are *shared* between all instances. This means that editing them in one instance will cause them to change in all other instances. Exported arrays can have initializers, but they must be constant expressions.

```
# Exported array, shared between all instances.
# Default value must be a constant expression.

export var a=[1,2,3]

# Typed arrays also work, only initialized empty:

export var vector3s = Vector3Array()
export var strings = StringArray()

# Regular array, created local for every instance.
# Default value can include run-time values, but can't
# be exported.

var b = [a,2,3]
```

Setters/getters

It is often useful to know when a class' member variable changes for whatever reason. It may also be desired to encapsulate its access in some way.

For this, GDScript provides a *setter/getter* syntax using the `setget` keyword. It is used directly after a variable definition:

```
var variable = value setget setterfunc, getterfunc
```

Whenever the value of `variable` is modified by an *external* source (i.e. not from local usage in the class), the *setter* function (`setterfunc` above) will be called. This happens *before* the value is changed. The *setter* must decide what to do with the new value. Vice-versa, when `variable` is accessed, the *getter* function (`getterfunc` above) must `return` the desired value. Below is an example:

```
var myvar setget myvar_set,myvar_get

func myvar_set(newvalue):
    myvar=newvalue

func myvar_get():
    return myvar # getter must return a value
```

Either of the *setter* or *getter* functions can be omitted:

```
# Only a setter
var myvar = 5 setget myvar_set
# Only a getter (note the comma)
var myvar = 5 setget ,myvar_get
```

Get/Setters are especially useful when exporting variables to editor in tool scripts or plugins, for validating input.

As said *local* access will *not* trigger the setter and getter. Here is an illustration of this:

```
func _init():
    # Does not trigger setter/getter
    myinteger=5
    print(myinteger)

    # Does trigger setter/getter
    self.myinteger=5
    print(self.myinteger)
```

Tool mode

Scripts, by default, don't run inside the editor and only the exported properties can be changed. In some cases it is desired that they do run inside the editor (as long as they don't execute game code or manually avoid doing so). For this, the `tool` keyword exists and must be placed at the top of the file:

```
tool
extends Button

func _ready():
    print("Hello")
```

Memory management

If a class inherits from *Reference*, then instances will be freed when no longer in use. No garbage collector exists, just simple reference counting. By default, all classes that don't define inheritance extend **Reference**. If this is not desired, then a class must inherit *Object* manually and must call instance.free(). To avoid reference cycles that can't be freed, a `weakref` function is provided for creating weak references.

Signals

It is often desired to send a notification that something happened in an instance. GDScript supports creation of built-in Godot signals. Declaring a signal in GDScript is easy using the *signal* keyword.

```
# No arguments
signal your_signal_name
# With arguments
signal your_signal_name_with_args(a,b)
```

These signals, just like regular signals, can be connected in the editor or from code. Just take the instance of a class where the signal was declared and connect it to the method of another instance:

```
func _callback_no_args():
    print("Got callback!")
```

```
func _callback_args(a,b):
    print("Got callback with args! a: ",a," and b: ",b)

func _at_some_func():
    instance.connect("your_signal_name",self,"_callback_no_args")
    instance.connect("your_signal_name_with_args",self,"_callback_args")
```

It is also possible to bind arguments to a signal that lacks them with your custom values:

```
func _at_some_func():
    instance.connect("your_signal_name",self,"_callback_args",[22,"hello"])
```

This is very useful when a signal from many objects is connected to a single callback and the sender must be identified:

```
func _button_pressed(which):
    print("Button was pressed: ",which.get_name())

func _ready():
    for b in get_node("buttons").get_children():
        b.connect("pressed",self,"_button_pressed",[b])
```

Finally, emitting a custom signal is done by using the Object.emit_signal method:

```
func _at_some_func():
    emit_signal("your_signal_name")
    emit_signal("your_signal_name_with_args",55,128)
    someinstance.emit_signal("somesignal")
```

Coroutines

GDScript offers support for coroutines via the `yield` built-in function. Calling `yield()` will immediately return from the current function, with the current frozen state of the same function as the return value. Calling `resume` on this resulting object will continue execution and return whatever the function returns. Once resumed the state object becomes invalid. Here is an example:

```
func myfunc():

   print("hello")
   yield()
   print("world")

func _ready():

    var y = myfunc()
    # Function state saved in 'y'
    print("my dear")
    y.resume()
    # 'y' resumed and is now an invalid state
```

Will print:

```
hello
my dear
world
```

It is also possible to pass values between yield() and resume(), for example:

```
func myfunc():

   print("hello")
   print( yield() )
   return "cheers!"

func _ready():

    var y = myfunc()
    # Function state saved in 'y'
    print( y.resume("world") )
    # 'y' resumed and is now an invalid state
```

Will print:

```
hello
world
cheers!
```

Coroutines & signals

The real strength of using `yield` is when combined with signals. `yield` can accept two parameters, an object and a signal. When the signal is received, execution will recommence. Here are some examples:

```
# Resume execution the next frame
yield( get_tree(), "idle_frame" )

# Resume execution when animation is done playing:
yield( get_node("AnimationPlayer"), "finished" )
```

Onready keyword

When using nodes, it's very common to desire to keep references to parts of the scene in a variable. As scenes are only warranted to be configured when entering the active scene tree, the sub-nodes can only be obtained when a call to Node._ready() is made.

```
var mylabel

func _ready():
    mylabel = get_node("MyLabel")
```

This can get a little cumbersome, specially when nodes and external references pile up. For this, GDScript has the `onready` keyword, that defers initialization of a member variable until _ready is called. It can replace the above code with a single line:

```
onready var mylabel = get_node("MyLabel")
```

10.2 GDScript more efficiently

10.2.1 About

This tutorial aims to be a quick reference for how to use GDScript more efficiently. It focuses in common cases specific to the language, but also covers a lot related to using dynamically typed languages.

It's meant to be specially useful for programmers without previous or little experience of dynamically typed languages.

10.2.2 Dynamic nature

Pros & cons of dynamic typing

GDScript is a Dynamically Typed language. As such, it's main advantages are that:

- Language is very simple to learn.
- Most code can be written and changed quickly and without hassle.
- Less code written means less errors & mistakes to fix.
- Easier to read the code (less clutter).
- No compilation is required to test.
- Runtime is tiny.
- Duck-typing and polymorphism by nature.

While the main cons are:

- Less performance than statically typed languages.
- More difficult to refactor (symbols can't be traced)
- Some errors that would typically be detected at compile time in statically typed languages only appear while running the code (because expression parsing is more strict).
- Less flexibility for code-completion (some variable types are only known at run-time).

This, translated to reality, means that Godot+GDScript are a combination designed to games very quickly and efficiently. For games that are very computationally intensive and can't benefit from the engine built-in tools (such as the Vector types, Physics Engine, Math library, etc), the possibility of using C++ is present too. This allows to still create the entire game in GDScript and add small bits of C++ in the areas that need a boost.

Variables & assignment

All variables in a dynamically typed language are "variant"-like. This means that their type is not fixed, and is only modified through assignment. Example:

Static:

```
int a; // value uninitialized
a = 5; // this is valid
a = "Hi!"; // this is invalid
```

Dynamic:

```
var a # null by default
a = 5 # valid, 'a' becomes an integer
a = "Hi!" # valid, 'a' changed to a string
```

As function arguments:

Functions are of dynamic nature too, which means they can be called with different arguments, for example:

Static:

```
void print_value(int value)
{
    printf("value is %i\n",value);
}

[..]

print_value(55); // valid
print_value("Hello"); // invalid
```

Dynamic:

```
func print_value(value):
    print(value)
[..]

print_value(55) # valid
print_value("Hello") # valid
```

Pointers & referencing:

In static languages such as C or C++ (and to some extent Java and C#), there is a distinction between a variable and a pointer/reference to a variable. The later allows the object to be modified by other functions by passing a reference to the original one.

In C# or Java, everything not a built-in type (int, float, sometimes String) is always a pointer or a reference. References are also garbage-collected automatically, which means they are erased when no longer used. Dynamically typed languages tend to use this memory model too. Some Examples:

- C++:

```
void use_class(SomeClass *instance) {

    instance->use();
}

void do_something() {

    SomeClass *instance = new SomeClass; // created as pointer
    use_class(instance); // passed as pointer
    delete instance; // otherwise it will leak memory
}
```

- Java:

```
@Override
public final void use_class(SomeClass instance) {

    instance.use();
}

public final void do_something() {

    SomeClass instance = new SomeClass(); // created as reference
    use_class(instance); // passed as reference
    // garbage collector will get rid of it when not in
    // use and freeze your game randomly for a second
}
```

- GDScript:

```
func use_class(instance); # does not care about class type
    instance.use() # will work with any class that has a ".use()" method.

func do_something():
    var instance = SomeClass.new() # created as reference
    use_class(instance) # passed as reference
    # will be unreferenced and deleted
```

In GDScript, only base types (int, float, string and the vector types) are passed by value to functions (value is copied). Everything else (instances, arrays, dictionaries, etc) is passed as reference. Classes that inherit *Reference* (the default if nothing is specified) will be freed when not used, but manual memory management is allowed too if inheriting manually from *Object*.

10.2.3 Arrays

Arrays in dynamically typed languages can contain many different mixed datatypes inside and are always dynamic (can be resized at any time). Compare for example arrays in statically typed languages:

```
int *array = new int[4]; // create array
array[0] = 10; // initialize manually
array[1] = 20; // can't mix types
array[2] = 40;
array[3] = 60;
// can't resize
use_array(array); // passed as pointer
delete[] array; // must be freed

//or

std::vector<int> array;
array.resize(4);
array[0] = 10; // initialize manually
array[1] = 20; // can't mix types
array[2] = 40;
array[3] = 60;
array.resize(3); // can be resized
use_array(array); // passed reference or value
// freed when stack ends
```

And in GDScript:

```
var array = [10, "hello", 40, 60] # simple, and can mix types
array.resize(3) # can be resized
use_array(array) # passed as reference
# freed when no longer in use
```

In dynamically typed languages, arrays can also double as other datatypes, such as lists:

```
var array = []
array.append(4)
array.append(5)
array.pop_front()
```

Or unordered sets:

```
var a = 20
if a in [10, 20, 30]:
    print("We have a winner!")
```

10.2.4 Dictionaries

Dictionaries are always a very powerful in dynamically typed languages. Most programmers that come from statically typed languages (such as C++ or C#) ignore their existence and make their life unnecessarily more difficult. This datatype is generally not present in such languages (or only on limited form).

Dictionaries can map any value to any other value with complete disregard for the datatype used as either key or value. Contrary to popular belief, they are very efficient because they can be implemented with hash tables. They are, in fact, so efficient that languages such as Lua will go as far as implementing arrays as dictionaries.

Example of Dictionary:

```
var d = { "name": "john", "age": 22 } # simple syntax
print("Name: ", d["name"], " Age: ", d["age"])
```

Dictionaries are also dynamic, keys can be added or removed at any point at little cost:

```
d["mother"] = "Rebecca" # addition
d["age"] = 11 # modification
d.erase("name") # removal
```

In most cases, two-dimensional arrays can often be implemented more easily with dictionaries. Here's a simple battleship game example:

```
# battleship game

const SHIP = 0
const SHIP_HIT = 1
const WATER_HIT = 2

var board = {}

func initialize():
    board[Vector(1,1)] = SHIP
    board[Vector(1,2)] = SHIP
    board[Vector(1,3)] = SHIP

func missile(pos):
```

```
    if pos in board: # something at that pos
        if board[pos] == SHIP: # there was a ship! hit it
            board[pos] = SHIP_HIT
        else:
            print("already hit here!") # hey dude you already hit here
    else: # nothing, mark as water
        board[pos] = WATER_HIT

func game():
    initialize()
    missile(Vector2(1,1))
    missile(Vector2(5,8))
    missile(Vector2(2,3))
```

Dictionaries can also be used as data markup or quick structures. While GDScript dictionaries resemble python dictionaries, it also supports Lua style syntax an indexing, which makes it very useful for writing initial states and quick structs:

```
# same example, lua-style support
# this syntax is a lot more readable and usable

var d = {
    name = "john",
    age = 22
}

print("Name: ", d.name, " Age: ", d.age) # used "." based indexing

# indexing

d.mother = "rebecca" # this doesn't work (use syntax below to add a key:value pair)
d["mother"] = "rebecca" # this works
d.name = "caroline" # if key exists, assignment does work, this is why it's like a␣
↪quick struct.
```

10.2.5 For & while

Iterating in some statically typed languages can be quite complex:

```
const char* strings = new const char*[50];

[..]

for(int i=0; i<50; i++)
{

    printf("value: %s\n", i, strings[i]);
}

// even in STL:

for(std::list<std::string>::const_iterator it = strings.begin(); it != strings.end();␣
↪it++) {

    std::cout << *it << std::endl;
```

```
}
```

This is usually greatly simplified in dynamically typed languages:

```
for s in strings:
    print(s)
```

Container datatypes (arrays and dictionaries) are iterable. Dictionaries allow iterating the keys:

```
for key in dict:
    print(key, " -> ", dict[key])
```

Iterating with indices is also possible:

```
for i in range(strings.size()):
    print(strings[i])
```

The range() function can take 3 arguments:

```
range(n) (will go from 0 to n-1)
range(b, n) (will go from b to n-1)
range(b, n, s) (will go from b to n-1, in steps of s)
```

Some examples:

```
for(int i=0; i<10; i++) {}

for(int i=5; i<10; i++) {}

for(int i=5; i<10; i+=2) {}
```

Translate to:

```
for i in range(10):

for i in range(5, 10):

for i in range(5, 10, 2):
```

And backwards looping is done through a negative counter:

```
for(int i=10; i>0; i--) {}
```

becomes

```
for i in range(10, 0, -1):
```

10.2.6 While

while() loops are the same everywhere:

```
var i = 0

while(i < strings.size()):
    print(strings[i])
    i += 1
```

10.2.7 Duck typing

One of the most difficult concepts to grasp when moving from a statically typed language to a dynamic one is duck typing. Duck typing makes overall code design much simpler and straightforward to write, but it's not obvious how it works.

As an example, imagine a situation where a big rock is falling down a tunnel, smashing everything on its way. The code for the rock, in a statically typed language would be something like:

```
void BigRollingRock::on_object_hit(Smashable *entity)
{
    entity->smash();
}
```

This, way, everything that can be smashed by a rock would have to inherit Smashable. If a character, enemy, piece of furniture, small rock were all smashable, they would need to inherit from the class Smashable, possibly requiring multiple inheritance. If multiple inheritance was undesired, then they would have to inherit a common class like Entity. Yet, it would not be very elegant to add a virtual method `smash()` to Entity only if a few of them can be smashed.

With dynamically typed languages, this is not a problem. Duck typing makes sure you only have to define a `smash()` function where required and that's it. No need to consider inheritance, base classes, etc.

```
func _on_object_hit(object):
    object.smash()
```

And that's it. If the object that hit the big rock has a smash() method, it will be called. No need for inheritance or polymorphism. Dynamically typed languages only care about the instance having the desired method or member, not what it inherits or the class type. The definition of Duck Typing should make this clearer:

"When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck"

In this case, it translates to:

"If the object can be smashed, don't care what it is, just smash it."

Yes, we should call it Hulk typing instead. Anyway though, there exists the possibility of the object being hit not having a smash() function. Some dynamically typed languages simply ignore a method call when it doesn't exist (like Objective C), but GDScript is more strict, so checking if the function exists is desirable:

```
func _on_object_hit(object):
    if (object.has_method("smash")):
        object.smash()
```

Then, simply define that method and anything the rock touches can be smashed.

10.3 Shading language

10.3.1 Introduction

Godot uses a simplified shader language (almost a subset of GLSL). Shaders can be used for:

- Materials
- Post-Processing
- 2D

and are divided in *Vertex*, *Fragment* and *Light* sections.

10.3.2 Language

Typing

The language is statically type and supports only a few operations. Arrays, classes, structures, etc are not supported. Several built-in datatypes are provided:

Data types

DataType	Description
void	Void
bool	boolean (true or false)
float	floating point
vec2	2-component vector, float subindices (x,y or r,g)
vec3	3-component vector, float subindices (x,y,z or r,g,b)
vec4, *color*	4-component vector, float subindices (x,y,z,w or r,g,b,a)
mat2	2x2 matrix, vec3 subindices (x,y)
mat3	3x3 matrix, vec3 subindices (x,y,z)
mat4	4x4 matrix, vec4 subindices (x,y,z,w)
texture	texture sampler, can only be used as uniform
cubemap	cubemap sampler, can only be used as uniform

Syntax

The syntax is similar to C, with statements ending with `;` and comments as `//` and `/* */`. Example:

```
float a = 3;
vec3 b;
b.x = a;
```

Swizzling

It is possible to use swizzling to reassigning subindices or groups of subindices, in order:

```
vec3 a = vec3(1,2,3);
vec3 b = a.zyx; // b will contain vec3(3,2,1)
vec2 c = a.xy; // c will contain vec2(1,2)
vec4 d = a.xyzz; // d will contain vec4(1,2,3,3)
```

Constructors

Constructors take the regular amount of elements, but can also accept less if the element has more subindices, for example:

```
vec3 a = vec3(1,vec2(2,3));
vec3 b = vec3(a);
vec3 c = vec3(vec2(2,3),1);
vec4 d = vec4(a,5);
mat3 m = mat3(a,b,c);
```

Conditionals

For now, only the `if` conditional is supported. Example:

```
if (a < b) {
    c = b;
}
```

Uniforms

A variable can be declared as uniform. In this case, its value will come from outside the shader (it will be the responsibility of the material or whatever using the shader to provide it).

```
uniform vec3 direction;
uniform color tint;

vec3 result = tint.rgb * direction;
```

Functions

Simple support for functions is provided. Functions can't access uniforms or other shader variables.

```
vec3 addtwo(vec3 a, vec3 b) {
    return a+b;
}

vec3 c = addtwo(vec3(1,1,1), vec3(2,2,2));
```

10.3.3 Built-in functions

Several built-in functions are provided for convenience, listed as follows:

Function	Description
float *sin* (float)	Sine
float *cos* (float)	Cosine
float *tan* (float)	Tangent
float *asin* (float)	arc-Sine
float *acos* (float)	arc-Cosine
float *atan* (float)	arc-Tangent
vec_type *pow* (vec_type, float)	Power
vec_type *pow* (vec_type, vec_type)	Power (Vec. Exponent)
vec_type *exp* (vec_type)	Base-e Exponential
vec_type *log* (vec_type)	Natural Logarithm
vec_type *sqrt* (vec_type)	Square Root
vec_type *abs* (vec_type)	Absolute
vec_type *sign* (vec_type)	Sign
vec_type *floor* (vec_type)	Floor
vec_type *trunc* (vec_type)	Trunc
vec_type *ceil* (vec_type)	Ceiling
vec_type *fract* (vec_type)	Fractional

Continúa en la página siguiente

Tabla 10.1 – proviene de la página anterior

Function	Description
vec_type *mod* (vec_type,vec_type)	Remainder
vec_type *min* (vec_type,vec_type)	Minimum
vec_type *min* (vec_type,vec_type)	Maximum
vec_type *clamp* (vec_type value,vec_type min, vec_type max)	Clamp to Min-Max
vec_type *mix* (vec_type a,vec_type b, float c)	Linear Interpolate
vec_type *mix* (vec_type a,vec_type b, vec_type c)	Linear Interpolate (Vector Coef.)
vec_type *step* (vec_type a,vec_type b)	‘ a[i] < b[i] ? 0.0 : 1.0‘
vec_type *smoothstep* (vec_type a,vec_type b,vec_type c)	
float *length* (vec_type)	Vector Length
float *distance* (vec_type, vec_type)	Distance between vector.
float *dot* (vec_type, vec_type)	Dot Product
vec3 *dot* (vec3, vec3)	Cross Product
vec_type *normalize* (vec_type)	Normalize to unit length
vec3 *reflect* (vec3, vec3)	Reflect
color *tex* (texture, vec2)	Read from a texture in normalized coords
color *texcube* (texture, vec3)	Read from a cubemap
color *texscreen* (vec2)	Read from screen (generates a copy)

10.3.4 Built-in variables

Depending on the shader type, several built-in variables are available, listed as follows:

Material - VertexShader

Variable	Description
const vec3 *SRC_VERTEX*	Model-Space Vertex
const vec3 *SRC_NORMAL*	Model-Space Normal
const vec3 *SRC_TANGENT*	Model-Space Tangent
const float *SRC_BINORMALF*	Direction to Compute Binormal
vec3 *VERTEX*	View-Space Vertex
vec3 *NORMAL*	View-Space Normal
vec3 *TANGENT*	View-Space Tangent
vec3 *BINORMAL*	View-Space Binormal
vec2 *UV*	UV
vec2 *UV2*	UV2
color *COLOR*	Vertex Color
out vec4 *VAR1*	Varying 1 Output
out vec4 *VAR2*	Varying 2 Output
out float *SPEC_EXP*	Specular Exponent (for Vertex Lighting)
out float *POINT_SIZE*	Point Size (for points)
const mat4 *WORLD_MATRIX*	Object World Matrix
const mat4 *INV_CAMERA_MATRIX*	Inverse Camera Matrix
const mat4 *PROJECTION_MATRIX*	Projection Matrix
const mat4 *MODELVIEW_MATRIX*	(InvCamera * Projection)
const float *INSTANCE_ID*	Instance ID (for multimesh)
const float *TIME*	Time (in seconds)

Material - FragmentShader

Variable	Description
const vec3 *VERTEX*	View-Space vertex
const vec4 *POSITION*	View-Space Position
const vec3 *NORMAL*	View-Space Normal
const vec3 *TANGENT*	View-Space Tangent
const vec3 *BINORMAL*	View-Space Binormal
const vec3 *NORMALMAP*	Alternative to NORMAL, use for normal texture output.
const vec3 *NORMALMAP_DEPTH*	Complementary to the above, allows changing depth of normalmap.
const vec2 *UV*	UV
const vec2 *UV2*	UV2
const color *COLOR*	Vertex Color
const vec4 *VAR1*	Varying 1
const vec4 *VAR2*	Varying 2
const vec2 *SCREEN_UV*	Screen Texture Coordinate (for using with texscreen)
const float *TIME*	Time (in seconds)
const vec2 *POINT_COORD*	UV for point, when drawing point sprites.
out vec3 *DIFFUSE*	Diffuse Color
out vec4 *DIFFUSE_ALPHA*	Diffuse Color with Alpha (using this sends geometry to alpha pipeline)
out vec3 *SPECULAR*	Specular Color
out vec3 *EMISSION*	Emission Color
out float *SPEC_EXP*	Specular Exponent (Fragment Version)
out float *GLOW*	Glow
out mat4 *INV_CAMERA_MATRIX*	Inverse camera matrix, can be used to obtain world coords (see example below).

Material - LightShader

Variable	Description
const vec3 *NORMAL*	View-Space normal
const vec3 *LIGHT_DIR*	View-Space Light Direction
const vec3 *EYE_VEC*	View-Space Eye-Point Vector
const vec3 *DIFFUSE*	Material Diffuse Color
const vec3 *LIGHT_DIFFUSE*	Light Diffuse Color
const vec3 *SPECULAR*	Material Specular Color
const vec3 *LIGHT_SPECULAR*	Light Specular Color
const float *SPECULAR_EXP*	Specular Exponent
const vec1 *SHADE_PARAM*	Generic Shade Parameter
const vec2 *POINT_COORD*	Current UV for Point Sprite
out vec2 *LIGHT*	Resulting Light
const float *TIME*	Time (in seconds)

CanvasItem (2D) - VertexShader

Variable	Description
const vec2 *SRC_VERTEX*	CanvasItem space vertex.
vec2 *UV*	UV
out vec2 *VERTEX*	Output LocalSpace vertex.
out vec2 *WORLD_VERTEX*	Output WorldSpace vertex. (use this or the one above)
color *COLOR*	Vertex Color
out vec4 *VAR1*	Varying 1 Output
out vec4 *VAR2*	Varying 2 Output
out float *POINT_SIZE*	Point Size (for points)
const mat4 *WORLD_MATRIX*	Object World Matrix
const mat4 *EXTRA_MATRIX*	Extra (user supplied) matrix via CanvasItem.draw_set_transform(). Identity by default.
const mat4 *PROJECTION_MATRIX*	Projection Matrix (model coords to screen).
const float *TIME*	Time (in seconds)

CanvasItem (2D) - FragmentShader

Variable	Description
const vec4 *SRC_COLOR*	Vertex color
const vec4 *POSITION*	Screen Position
vec2 *UV*	UV
out color *COLOR*	Output Color
out vec3 *NORMAL*	Optional Normal (used for 2D Lighting)
out vec3 *NORMALMAP*	Optional Normal in standard normalmap format (flipped y and Z from 0 to 1)
out float *NORMALMAP_DEPTH*	Depth option for above normalmap output, default value is 1.0
const texture *TEXTURE*	Current texture in use for CanvasItem
const vec2 *TEXTURE_PIXEL_SIZE*	Pixel size for current 2D texture
in vec4 *VAR1*	Varying 1 Output
in vec4 *VAR2*	Varying 2 Output
const vec2 *SCREEN_UV*	Screen Texture Coordinate (for using with texscreen)
const vec2 *POINT_COORD*	Current UV for Point Sprite
const float *TIME*	Time (in seconds)

CanvasItem (2D) - LightShader

10.3.5 Examples

Material that reads a texture, a color and multiples them, fragment program:

```
uniform color modulate;
uniform texture source;

DIFFUSE = modulate.rgb * tex(source, UV).rgb;
```

Material that glows from red to white:

```
DIFFUSE = vec3(1,0,0) + vec(1,1,1) * mod(TIME, 1.0);
```

Standard Blinn Lighting Shader

```
float NdotL = max(0.0, dot(NORMAL, LIGHT_DIR));
vec3 half_vec = normalize(LIGHT_DIR + EYE_VEC);
float eye_light = max(dot(NORMAL, half_vec), 0.0);
LIGHT = LIGHT_DIFFUSE + DIFFUSE + NdotL;
if (NdotL > 0.0) {
    LIGHT += LIGHT_SPECULAR + SPECULAR + pow(eye_light, SPECULAR_EXP);
};
```

Obtaining world-space normal and position in material fragment program:

```
// Use reverse multiply because INV_CAMERA_MATRIX is world2cam

vec3 world_normal = NORMAL * mat3(INV_CAMERA_MATRIX);
vec3 world_pos = (VERTEX - INV_CAMERA_MATRIX.w.xyz) * mat3(INV_CAMERA_MATRIX);
```

10.3.6 Notes

- **Do not** use DIFFUSE_ALPHA unless you really intend to use transparency. Transparent materials must be sorted by depth and slow down the rendering pipeline. For opaque materials, just use DIFFUSE.
- **Do not** use DISCARD unless you really need it. Discard makes rendering slower, specially on mobile devices.
- TIME may reset after a while (may last an hour or so), it's meant for effects that vary over time.
- In general, every built-in variable not used results in less shader code generated, so writing a single giant shader with a lot of code and optional scenarios is often not a good idea.

10.4 Locales

This is the list of supported locales and variants in the engine. It's based on the Unix standard locale strings:

Locale	Language and Variant
ar	Arabic
ar_AE	Arabic (United Arab Emirates)
ar_BH	Arabic (Bahrain)
ar_DZ	Arabic (Algeria)
ar_EG	Arabic (Egypt)
ar_IQ	Arabic (Iraq)
ar_JO	Arabic (Jordan)
ar_KW	Arabic (Kuwait)
ar_LB	Arabic (Lebanon)
ar_LY	Arabic (Libya)
ar_MA	Arabic (Morocco)
ar_OM	Arabic (Oman)
ar_QA	Arabic (Qatar)
ar_SA	Arabic (Saudi Arabia)
ar_SD	Arabic (Sudan)

Continúa en la página siguiente

Tabla 10.2 – proviene de la página anterior

Locale	Language and Variant
ar_SY	Arabic (Syria)
ar_TN	Arabic (Tunisia)
ar_YE	Arabic (Yemen)
be	Belarusian
be_BY	Belarusian (Belarus)
bg	Bulgarian
bg_BG	Bulgarian (Bulgaria)
ca	Catalan
ca_ES	Catalan (Spain)
cs	Czech
cs_CZ	Czech (Czech Republic)
da	Danish
da_DK	Danish (Denmark)
de	German
de_AT	German (Austria)
de_CH	German (Switzerland)
de_DE	German (Germany)
de_LU	German (Luxembourg)
el	Greek
el_CY	Greek (Cyprus)
el_GR	Greek (Greece)
en	English
en_AU	English (Australia)
en_CA	English (Canada)
en_GB	English (United Kingdom)
en_IE	English (Ireland)
en_IN	English (India)
en_MT	English (Malta)
en_NZ	English (New Zealand)
en_PH	English (Philippines)
en_SG	English (Singapore)
en_US	English (United States)
en_ZA	English (South Africa)
es	Spanish
es_AR	Spanish (Argentina)
es_BO	Spanish (Bolivia)
es_CL	Spanish (Chile)
es_CO	Spanish (Colombia)
es_CR	Spanish (Costa Rica)
es_DO	Spanish (Dominican Republic)
es_EC	Spanish (Ecuador)
es_ES	Spanish (Spain)
es_GT	Spanish (Guatemala)
es_HN	Spanish (Honduras)
es_MX	Spanish (Mexico)
es_NI	Spanish (Nicaragua)
es_PA	Spanish (Panama)
es_PE	Spanish (Peru)
es_PR	Spanish (Puerto Rico)

Continúa en la página siguiente

Tabla 10.2 – proviene de la página anterior

Locale	Language and Variant
es_PY	Spanish (Paraguay)
es_SV	Spanish (El Salvador)
es_US	Spanish (United States)
es_UY	Spanish (Uruguay)
es_VE	Spanish (Venezuela)
et	Estonian
et_EE	Estonian (Estonia)
fi	Finnish
fi_FI	Finnish (Finland)
fr	French
fr_BE	French (Belgium)
fr_CA	French (Canada)
fr_CH	French (Switzerland)
fr_FR	French (France)
fr_LU	French (Luxembourg)
ga	Irish
ga_IE	Irish (Ireland)
hi	Hindi (India)
hi_IN	Hindi (India)
hr	Croatian
hr_HR	Croatian (Croatia)
hu	Hungarian
hu_HU	Hungarian (Hungary)
in	Indonesian
in_ID	Indonesian (Indonesia)
is	Icelandic
is_IS	Icelandic (Iceland)
it	Italian
it_CH	Italian (Switzerland)
it_IT	Italian (Italy)
iw	Hebrew
iw_IL	Hebrew (Israel)
ja	Japanese
ja_JP	Japanese (Japan)
ja_JP_JP	Japanese (Japan,JP)
ko	Korean
ko_KR	Korean (South Korea)
lt	Lithuanian
lt_LT	Lithuanian (Lithuania)
lv	Latvian
lv_LV	Latvian (Latvia)
mk	Macedonian
mk_MK	Macedonian (Macedonia)
ms	Malay
ms_MY	Malay (Malaysia)
mt	Maltese
mt_MT	Maltese (Malta)
nl	Dutch
nl_BE	Dutch (Belgium)

Continúa en la página siguiente

Tabla 10.2 – proviene de la página anterior

Locale	Language and Variant
nl_NL	Dutch (Netherlands)
no	Norwegian
no_NO	Norwegian (Norway)
no_NO_NY	Norwegian (Norway,Nynorsk)
pl	Polish
pl_PL	Polish (Poland)
pt	Portuguese
pt_BR	Portuguese (Brazil)
pt_PT	Portuguese (Portugal)
ro	Romanian
ro_RO	Romanian (Romania)
ru	Russian
ru_RU	Russian (Russia)
sk	Slovak
sk_SK	Slovak (Slovakia)
sl	Slovenian
sl_SI	Slovenian (Slovenia)
sq	Albanian
sq_AL	Albanian (Albania)
sr	Serbian
sr_BA	Serbian (Bosnia and Herzegovina)
sr_CS	Serbian (Serbia and Montenegro)
sr_ME	Serbian (Montenegro)
sr_RS	Serbian (Serbia)
sv	Swedish
sv_SE	Swedish (Sweden)
th	Thai
th_TH	Thai (Thailand)
th_TH_TH	Thai (Thailand,TH)
tr	Turkish
tr_TR	Turkish (Turkey)
uk	Ukrainian
uk_UA	Ukrainian (Ukraine)
vi	Vietnamese
vi_VN	Vietnamese (Vietnam)
zh	Chinese
zh_CN	Chinese (China)
zh_HK	Chinese (Hong Kong)
zh_SG	Chinese (Singapore)
zh_TW	Chinese (Taiwan)

10.5 BBCode in RichTextLabel

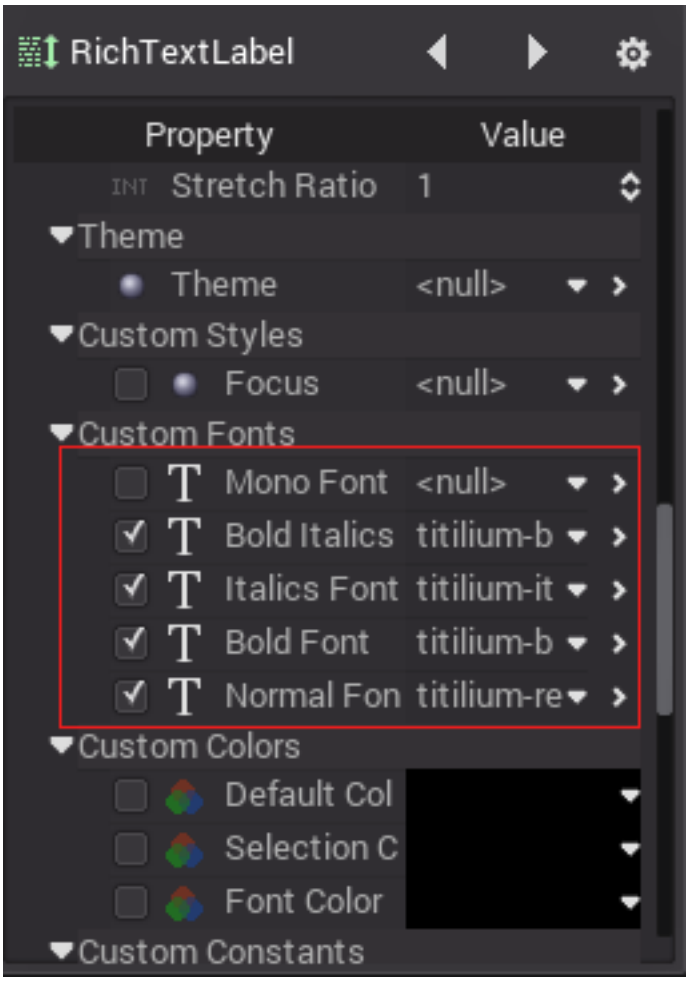
10.5.1 Introduction

RichTextLabel allows to display complex text markup in a control. It has a built-in API for generating the markup, but can also parse a BBCode.

Note that the BBCode tags can also be used to some extent in the *XML source of the class reference*.

10.5.2 Setting up

For RichTextLabel to work properly, it must be set up. This means loading the intended fonts in the relevant properties:



10.5.3 Reference

Built-in color names

List of valid color names for the [color=<name>] tag:

- aqua
- black
- blue
- fuchsia
- gray
- green
- lime
- maroon

- navy
- purple
- red
- silver
- teal
- white
- yellow

Hexadecimal color codes

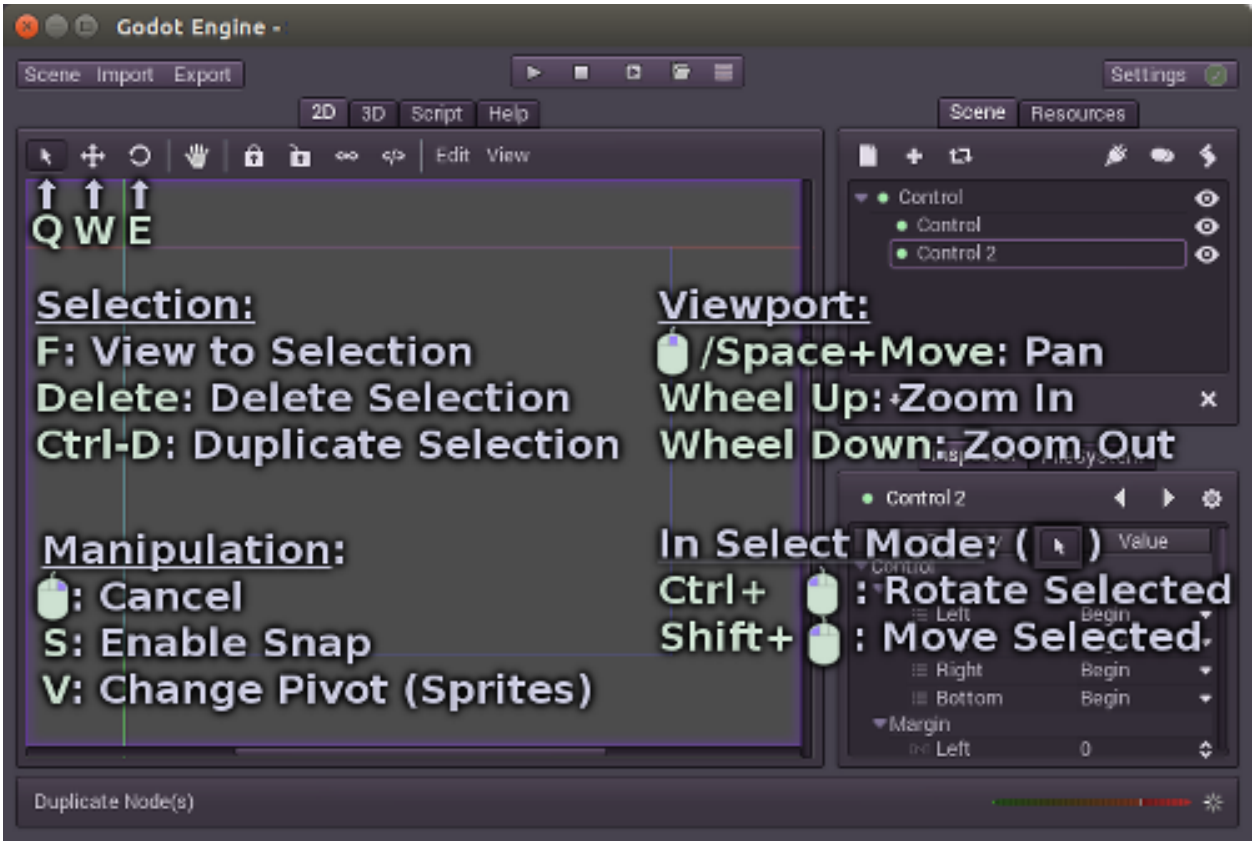
Any valid 6 digit hexadecimal code is supported. e.g: [color=#ffffff]white[/color]

CAPÍTULO 11

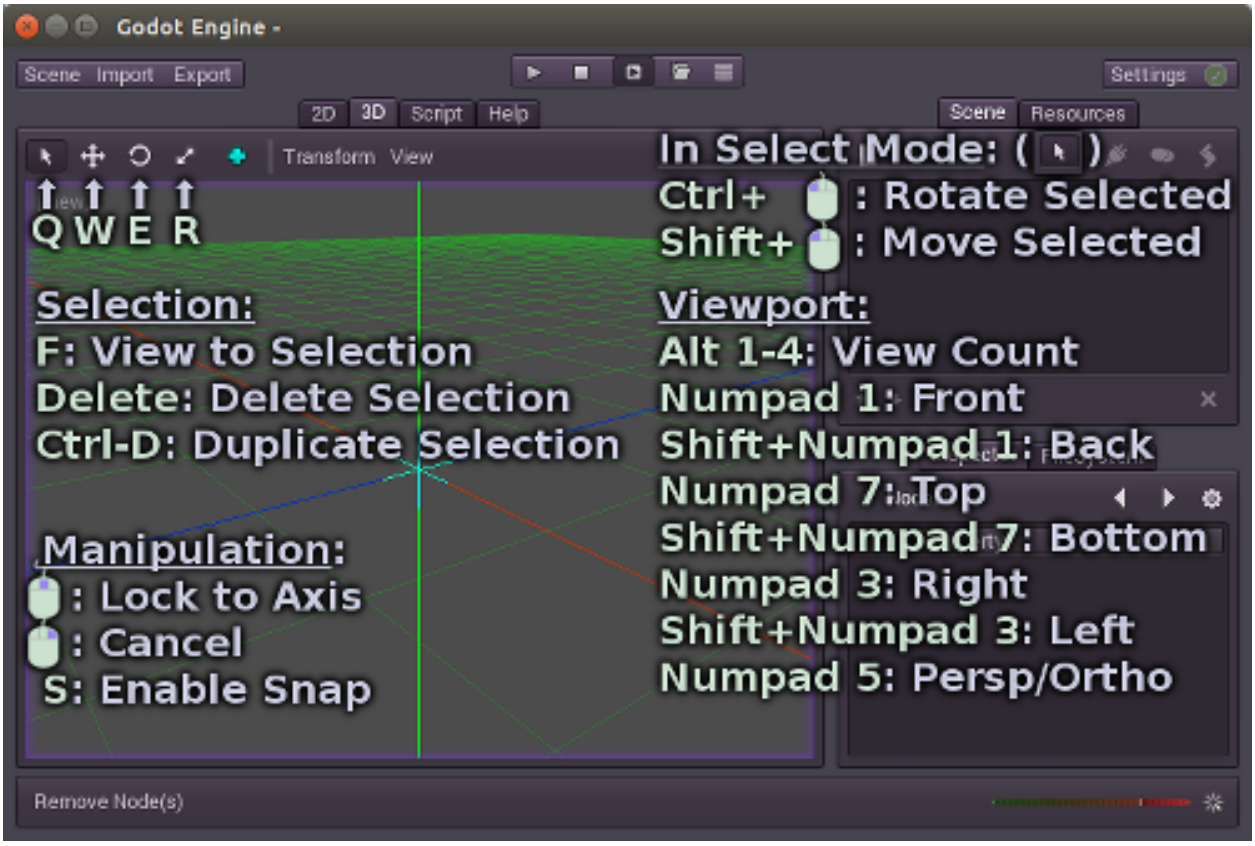
Cheat sheets

11.1 2D and 3D keybindings

11.1.1 2D viewport

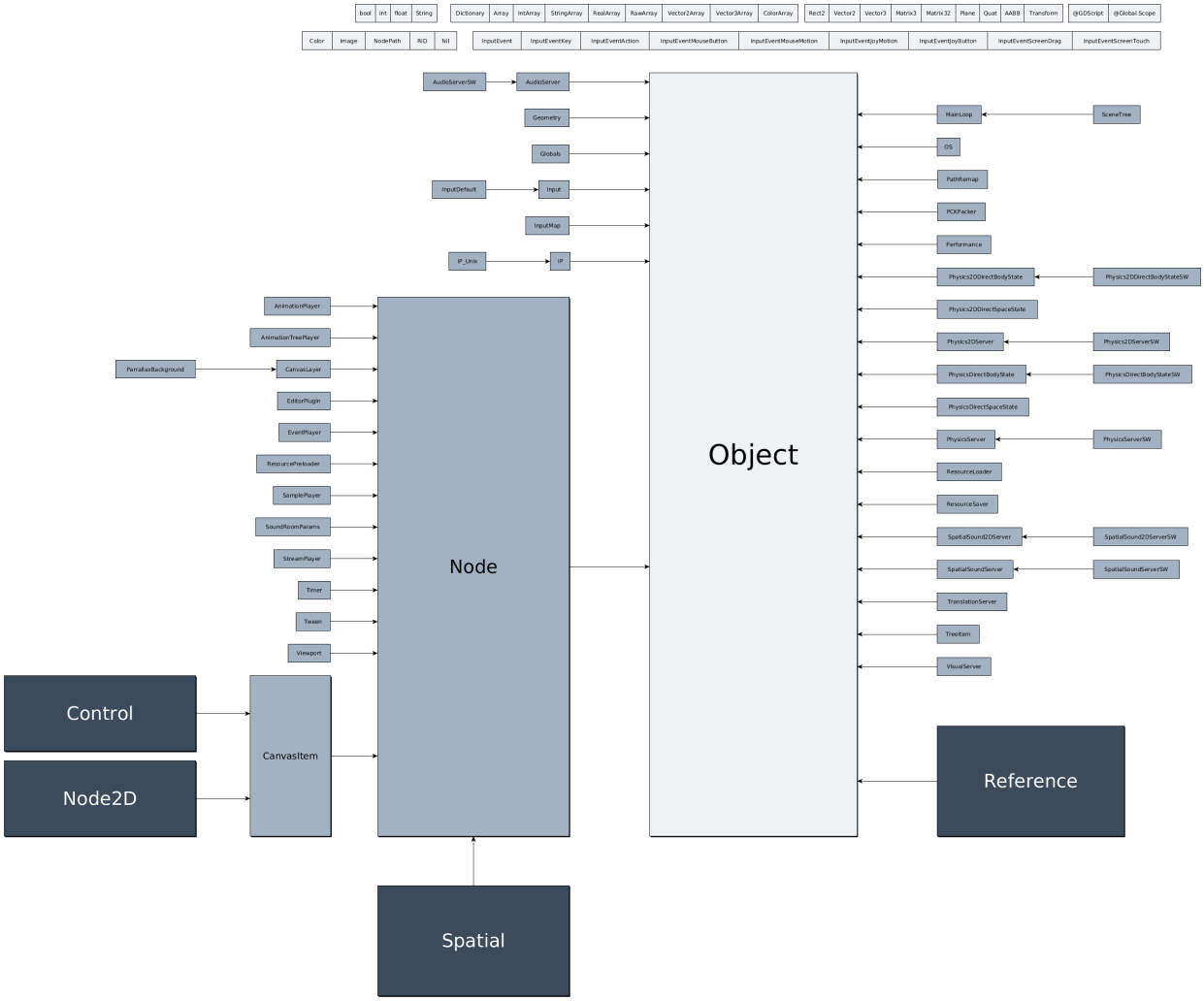


11.1.2 3D viewport

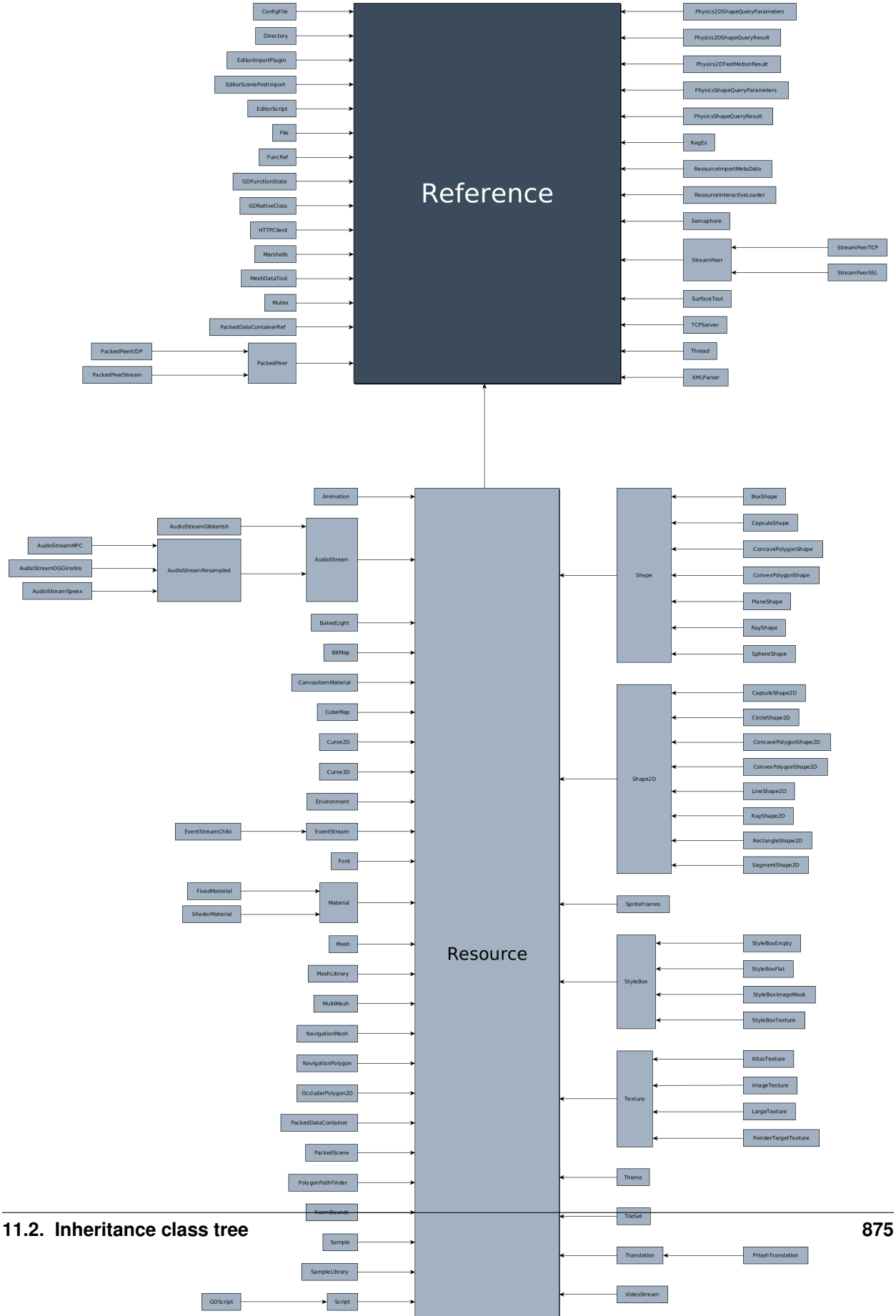


11.2 Inheritance class tree

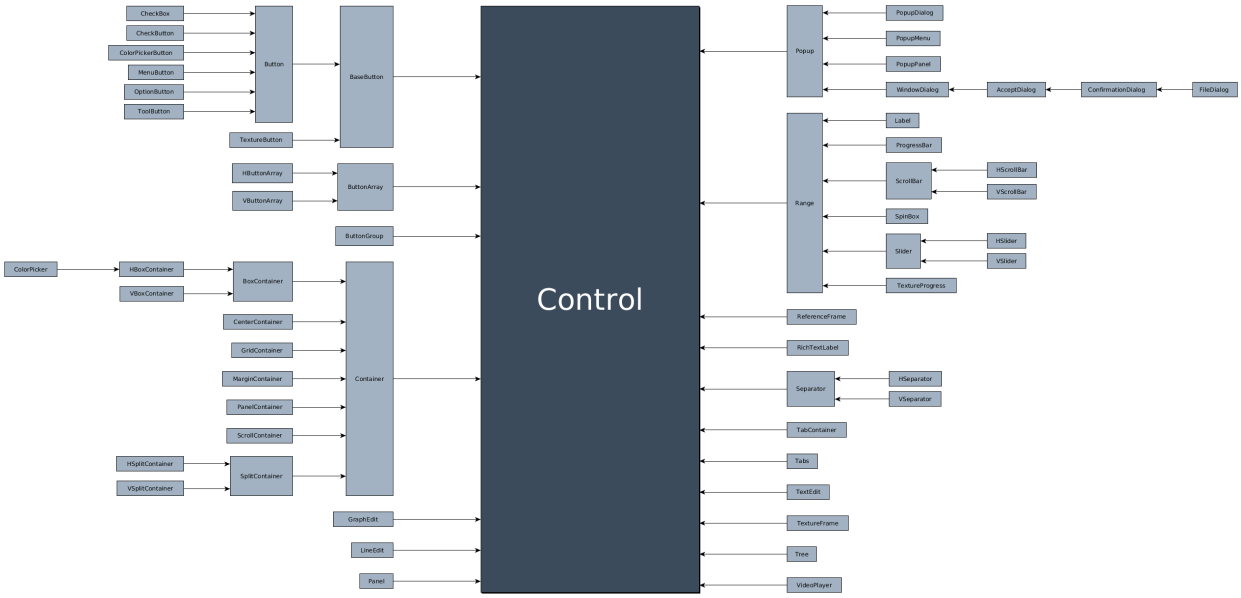
11.2.1 Object



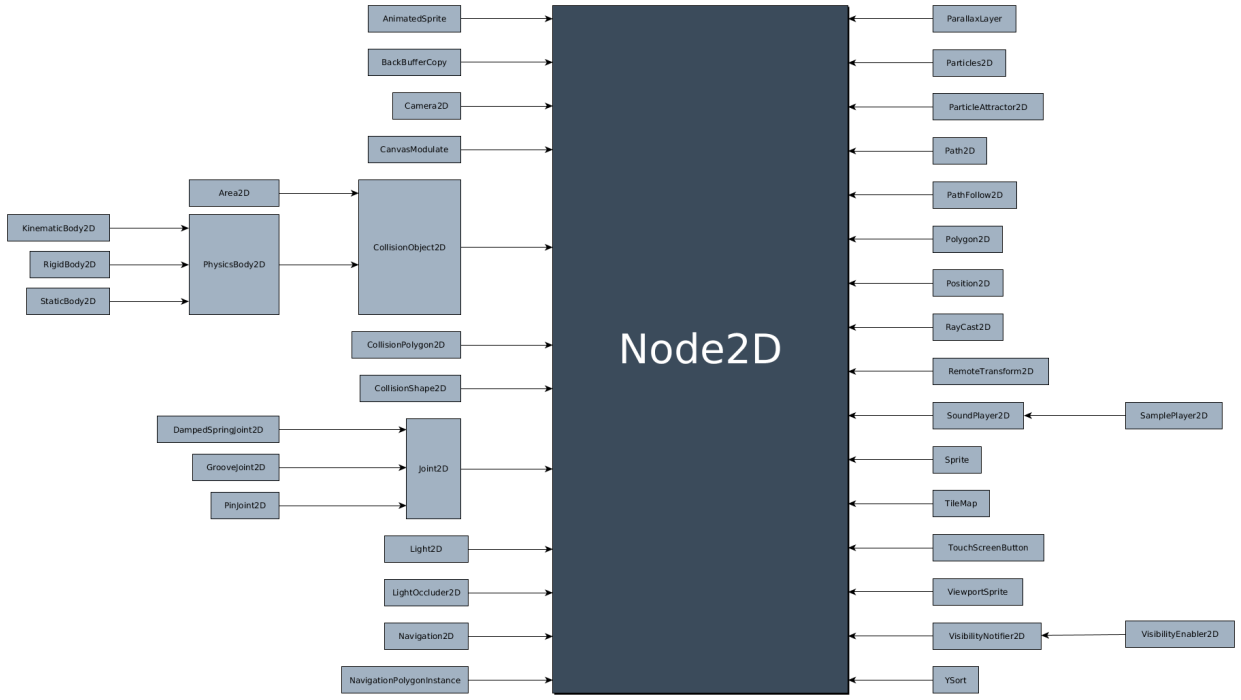
11.2.2 Reference



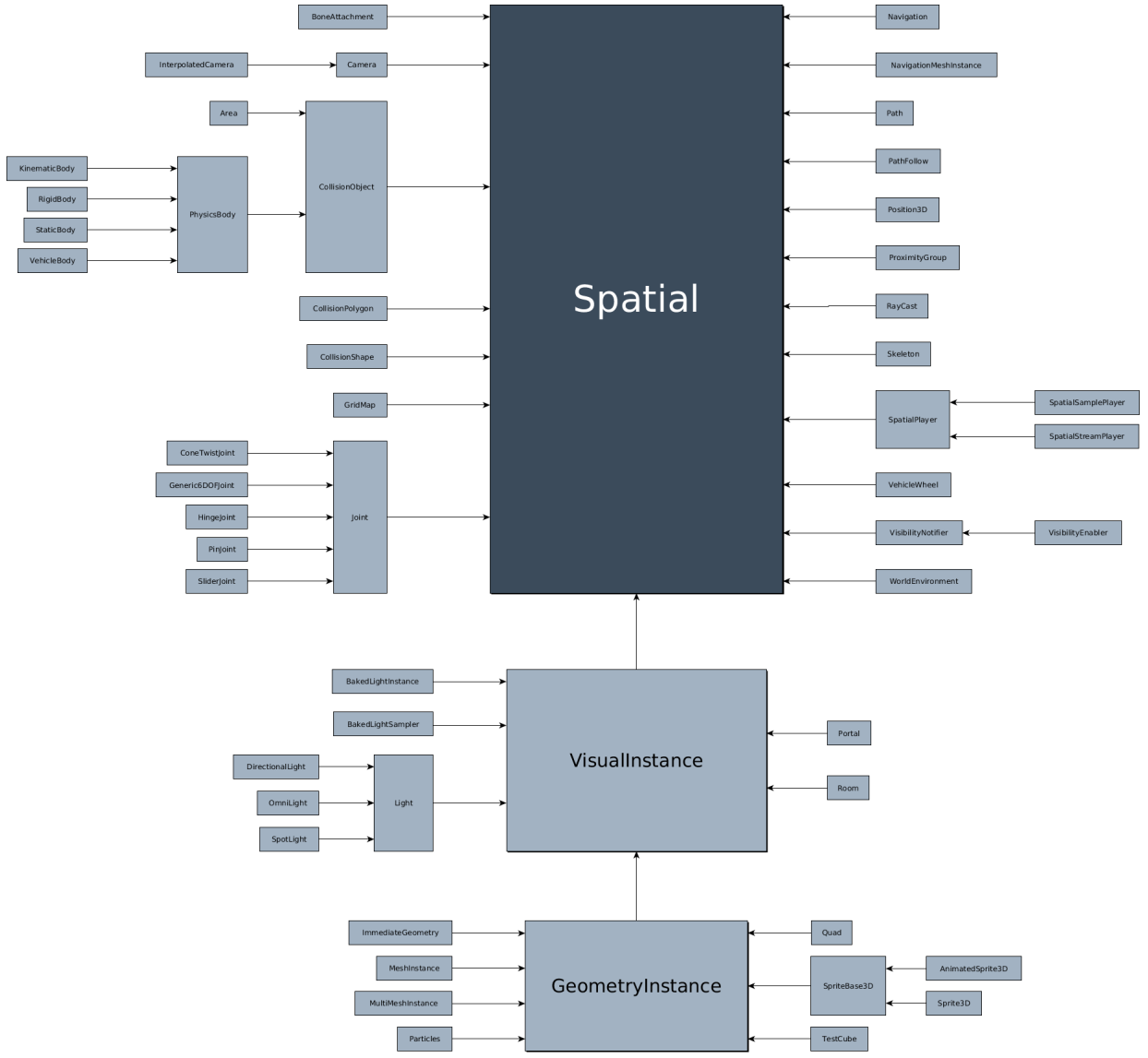
11.2.3 Control



11.2.4 Node2D



11.2.5 Spatial



Source files: `class_tree.zip`.

CAPÍTULO 12

Compiling

12.1 Introduction to the buildsystem

12.1.1 Scons

Godot uses Scons to build. We love it, we are not changing it for anything else. We are not even sure other build systems are up to the task of building Godot. We constantly get requests to move the build system to CMake, or Visual Studio, but this is not going to happen. There are many reasons why we have chosen SCons over other alternatives and are listed as follows:

- Godot can be compiled for a dozen different platforms. All PC platforms, all mobile platforms, many consoles, and many web-based platforms (such as HTML5 and Chrome PNACL).
- Developers often need to compile for several of the platforms **at the same time**, or even different targets of the same platform. They can't afford reconfiguring and rebuilding the project each time. SCons can do this with no sweat, without breaking the builds.
- SCons will *never* break a build no matter how many changes, configurations, additions, removals etc. You have more chances to die struck by lightning than needing to clean and rebuild in SCons.
- Godot build process is not simple. Several files are generated by code (binders), others are parsed (shaders), and others need to offer customization (plugins). This requires complex logic which is easier to write in an actual programming language (like Python) rather than using a mostly macro-based language only meant for building.
- Godot build process makes heavy use of cross compiling tools. Each platform has a specific detection process, and all these must be handled as specific cases with special code written for each.

So, please get at least a little familiar with it if you are planning to build Godot yourself.

12.1.2 Platform selection

Godot's build system will begin by detecting the platforms it can build for. If not detected, the platform will simply not appear on the list of available platforms. The build requirements for each platform are described in the rest of this tutorial section.

Scons is invoked by just calling `scons`.

However, this will do nothing except list the available platforms, for example:

```
user@host:~/godot$ scons
scons: Reading SConscript files ...
No valid target platform selected.
The following were detected:
    android
    server
    javascript
    windows
    x11

Please run scons again with argument: platform=<string>
scons: done reading SConscript files.
scons: Building targets ...
scons: `.' is up to date.
scons: done building targets.
```

To build for a platform (for example, x11), run with the `platform=` (or just `p=` to make it short) argument:

```
user@host:~/godot$ scons platform=x11
```

This will start the build process, which will take a while. If you want scons to build faster, use the `-j <cores>` parameter to specify how many cores will be used for the build. Or just leave it using one core, so you can use your computer for something else :)

Example for using 4 cores:

```
user@host:~/godot$ scons platform=x11 -j 4
```

12.1.3 Resulting binary

The resulting binaries will be placed in the bin/ subdirectory, generally with this naming convention:

```
godot.<platform>.[opt].[tools/debug].<architecture>[extension]
```

For the previous build attempt the result would look like this:

```
user@host:~/godot$ ls bin
bin/godot.x11.tools.64
```

This means that the binary is for X11, is not optimized, has tools (the whole editor) compiled in, and is meant for 64 bits.

A Windows binary with the same configuration will look like this.

```
C:\GODOT> DIR BIN/
godot.windows.tools.64.exe
```

Just copy that binary to wherever you like, as it self-contains the project manager, editor and all means to execute the game. However, it lacks the data to export it to the different platforms. For that the export templates are needed (which can be either downloaded from *godotengine.org* <*http://godotengine.org*>, or you can build them yourself).

Aside from that, there are a few standard options that can be set in all build targets, and will be explained as follows.

12.1.4 Tools

Tools are enabled by default in al PC targets (Linux, Windows, OSX), disabled for everything else. Disabling tools produces a binary that can run projects but that does not include the editor or the project manager.

```
scons platform=<platform> tools=yes/no
```

12.1.5 Target

Target controls optimization and debug flags. Each mode means:

- **debug**: Build with C++ debugging symbols, runtime checks (performs checks and reports error) and none to little optimization.
- **release_debug**: Build without C++ debugging symbols and optimization, but keep the runtime checks (performs checks and reports errors). Official binaries use this configuration.
- **release**: Build without symbols, with optimization and with little to no runtime checks. This target can't be used together with tools=yes, as the tools require some debug functionality and run-time checks to run.

```
scons platform=<platform> target=debug/release_debug/release
```

This flag appends ".debug" suffix (for debug), or ".tools" (for debug with tools enabled). When optimization is enabled (release) it appends the ".opt" suffix.

12.1.6 Bits

Bits is meant to control the CPU or OS version intended to run the binaries. It works mostly on desktop platforms and ignored everywhere else.

- **32**: Build binaries for 32 bits platform.
- **64**: Build binaries for 64 bits platform.
- **default**: Built whatever the build system feels is best. On Linux this depends on the host platform (if not cross compiling), while on Windows and Mac it defaults to produce 32 bits binaries unless 64 bits is specified.

```
scons platform=<platform> bits=default/32/64
```

This flag appends ".32" or ".64" suffixes to resulting binaries when relevant.

12.1.7 Export templates

Official export templates are downloaded from the Godot Engine site: *godotengine.org <http://godotengine.org>*. However, you might want to build them yourself (in case you want newer ones, you are using custom modules, or simply don't trust your own shadow).

If you download the official export templates package and unzip it, you will notice that most are just optimized binaries or packages for each platform:

```
android_debug.apk
android_release.apk
javascript_debug.zip
javascript_release.zip
linux_server_32
linux_server_64
```

```
linux_x11_32_debug
linux_x11_32_release
linux_x11_64_debug
linux_x11_64_release
osx.zip
version.txt
windows_32_debug.exe
windows_32_release.exe
windows_64_debug.exe
windows_64_release.exe
```

To create those yourself, just follow the instructions detailed for each platform in this same tutorial section. Each platform explains how to create it's own template.

If you are working for multiple platforms, OSX is definitely the best host platform for cross compilation, since you can cross-compile for almost every target (except for winrt). Linux and Windows come in second place, but Linux has the advantage of being the easier platform to set this up.

12.2 Compiling for Windows

12.2.1 Requirements

For compiling under Windows, the following is required:

- Visual C++, Visual Studio Community (recommended), at least the 2013 version (12.0) up to 2015 (14.0). **If you're using Express, make sure you get/have a version that can compile for C++, Desktop**.
- Python 2.7+ (3.0 is untested as of now). Using the 32-bits installer is recommended.
- Pywin32 Python Extension for parallel builds (which increase the build speed by a great factor).
- SCons build system.

12.2.2 Setting up SCons

Python adds the interpreter (python.exe) to the path. It usually installs in `C:\Python` (or `C:\Python[Version]`). SCons installs inside the Python install and provides a batch file called "scons.bat". The location of this file can be added to the path or it can simply be copied to `C:\Python` together with the interpreter executable.

To check whether you have installed Python and SCons correctly, you can type `python --version` and `scons --version` into the standard Windows Command Prompt (cmd.exe).

12.2.3 Downloading Godot's source

Godot's source is hosted on GitHub. Downloading it (cloning) via Git is recommended.

The tutorial will presume from now on that you placed the source into `C:\godot`.

12.2.4 Compiling

SCons will not be able out of the box to compile from the standard Windows "Command Prompt" (cmd.exe) because SCons and Visual C++ compiler will not be able to locate environment variables and executables they need for compilation.

Therefore, you need to start a Visual Studio command prompt. It sets up environment variables needed by SCons to locate the compiler. It should be called similar to one of the bellow names (for your respective version of Visual Studio):

- "Developer Command Prompt for VS2013"
- "VS2013 x64 Native Tools Command Prompt"
- "VS2013 x86 Native Tools Command Prompt"
- "VS2013 x64 Cross Tools Command Prompt"
- "VS2013 x86 Cross Tools Command Prompt"

You should be able to find at least the Developer Command Prompt for your version of Visual Studio in your start menu.

However Visual Studio sometimes seems to not install some of the above shortcuts, except the Developer Console at these locations that are automatically searched by the start menu search option:

```
Win 7:
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio␣
↪Tools
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2013\Visual Studio␣
↪Tools
```

If you found the Developer Console, it will do for now to create a 32 bit version of Godot, but if you want the 64 bit version, you might need to setup the prompts manually for easy access.

If you don't see some of the shortcuts, "How the prompts actually work" section bellow will explain how to setup these prompts if you need them.

About the Developer/Tools Command Prompts and the Visual C++ compiler

There is a few things you need to know about these consoles and the Visual C++ compiler.

Your Visual Studio installation will ship with several Visual C++ compilers, them being more or less identical, however each cl.exe (Visual C++ compiler) will compile Godot for a different architecture (32 or 64 bit, ARM compiler is not supported).

The **Developer Command Prompt** will build a 32 bit version of Godot by using the 32 bit Visual C++ compiler.

Native Tools Prompts (mentioned above) are used when you want the 32bit cl.exe to compile a 32 bit executable (x86 Native Tools Command Prompt). For the 64 bit cl.exe, it will compile a 64 bit executable (x64 Native Tools Command Prompt).

The **Cross Tools** are used when your Windows is using one architecture (32 bit, for example) and you need to compile to a different architecture (64 bit). As you might be familiar, 32 bit Windows can not run 64 bit executables, but you still might need to compile for them.

For example:

- "VS2013 x64 Cross Tools Command Prompt" will use a 32 bit cl.exe that will compile a 64 bit application.
- "VS2013 x86 Cross Tools Command Prompt" will use a 64 bit cl.exe that will compile a 32 bit application. This one is useful if you are running a 32 bit Windows.

On a 64 bit Windows, you can run any of above prompts and compilers (cl.exe executables) because 64 bit windows can run any 32 bit application. 32 bit Windows can not run 64 bit executables, so the Visual Studio installer will not even install shortcuts for some of these prompts.

Note that you need to choose the **Developer Console** or the correct **Tools Prompt** to build Godot for the correct architecture. Use only Native Prompts if you are not sure yet what exactly Cross Compile Prompts do.

Running SCons

Once inside the **Developer Console/Tools Console Prompt**, go to the root directory of the engine source code and type:

```
C:\godot> scons platform=windows
```

Tip: if you installed "Pywin32 Python Extension" you can append the -j command to instruct SCons to run parallel builds like this:

```
C:\godot> scons -j6 platform=windows
```

In general, it is OK to have at least as many threads compiling Godot as you have cores in your CPU, if not one or two more, I use -j6 (six threads) for my 4 core CPU, your mileage may vary. Feel free to add -j option to any SCons command you see bellow if you setup the "Pywin32 Python Extension".

If all goes well, the resulting binary executable will be placed in `C:\godot\bin\` with the name of `godot.windows.tools.32.exe` or `godot.windows.tools.64.exe`. SCons will automatically detect what compiler architecture the environment (the prompt) is setup for and will build a corresponding executable.

This executable file contains the whole engine and runs without any dependencies. Executing it will bring up the project manager.

How the prompts actually work

The Visual Studio command prompts are just shortcuts that call the standard Command Prompt and have it run a batch file before giving you control. The batch file itself is called **vcvarsall.bat** and it sets up environment variables, including the PATH variable, so that the correct version of the compiler can be run.The Developer Command Prompt calls a different file called **VsDevCmd.bat** but none of the other tools that this batch file enables are needed by Godot/SCons.

Since you are probably using VS2013 or VS2015, if you need to recreate them manually, use the bellow folders, or place them on the desktop/taskbar:

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio␣
↪Tools
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2013\Visual Studio␣
↪Tools
```

Start the creation of the shortcut by pressing the `right mouse button/New/Shortcut` in an empty place in your desired location.

Then copy one of these commands bellow for the corresponding tool you need into the "Path" and "Name" sections of the shortcut creation wizard, and fix the path to the batch file if needed.

- VS2013 is in the "Microsoft Visual Studio 12.0" folder.
- VS2015 is in the "Microsoft Visual Studio 14.0" folder.
- etc.

```
Name: Developer Command Prompt for VS2013
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.
↪0\Common7\Tools\VsDevCmd.bat""
```

```
Name: VS2013 x64 Cross Tools Command Prompt
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.
↪bat"" x86_amd64

Name: VS2013 x64 Native Tools Command Prompt
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.
↪bat"" amd64

Name: VS2013 x86 Native Tools Command Prompt
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.
↪bat"" x86

Name: VS2013 x86 Cross Tools Command Prompt
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.
↪bat"" amd64_x86
```

After you create the shortcut, in the shortcut's properties, that you can access by right clicking with your mouse on the shortcut itself, you can choose the starting directory of the command prompt ("Start in" field).

Some of these shortcuts (namely the 64 bit compilers) seem to not be available in the Express edition of Visual Studio or Visual C++.

In case you are wondering what these prompt shortcuts do, they call the standard cmd.exe with \k option and have it run a batch file...

```
%comspec% - path to cmd.exe
\k - keep alive option of the command prompt
remainder - command to run via cmd.exe

cmd.exe \k(eep cmd.exe alive after commands behind this option run) ""runme.bat""␣
↪with_this_option
```

How to run an automated build of Godot

If you need to just run the compilation process via a batch file or directly in the vanilla Windows Command Prompt you need to do the following command:

```
"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" x86
```

with one of the following parameters:

- x86 (32 bit cl.exe to compile for the 32 bit architecture)
- amd64 (64 bit cl.exe to compile for the 64 bit architecture)
- x86_amd64 (32 bit cl.exe to compile for the 64 bit architecture)
- amd64_x86 (64 bit cl.exe to compile for the 32 bit architecture)

and after that one, you can run SCons:

```
scons platform=windows
```

or you can do them together:

```
32 bit Godot
"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" x86 && scons␣
↪platform=windows
```

```
64 bit Godot
"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" amd64 && scons␣
↪platform=windows
```

12.2.5 Development in Visual Studio or other IDEs

For most projects, using only scripting is enough but when development in C++ is needed, for creating modules or extending the engine, working with an IDE is usually desirable.

You can create a Visual Studio solution via SCons by running SCons with the vsproj=yes parameter, like this:

```
scons p=windows vsproj=yes
```

You will be able to open Godot's source in a Visual Studio solution now but, currently, you can not build Godot via Visual Studio, as it does not work. It can be made to work manually if you are inclined to do so (.bat file called from NMake settings) but it is beyond the scope of this article.

12.2.6 Cross-compiling for Windows from other operating systems

If you are a Linux or Mac user, you need to install mingw32 and mingw-w64. Under Ubuntu or Debian, just run the following commands:

```
apt-get install mingw32 mingw-w64
```

If you are using another distro, SCons will check for the following binaries:

```
i586-mingw32msvc-gcc
i686-w64-mingw32-gcc
```

If the binaries are named or located somewhere else, export the following env variables:

```
export MINGW32_PREFIX="/path/to/i586-mingw32msvc-"
export MINGW64_PREFIX="/path/to/i686-w64-mingw32-"
```

To make sure you are doing things correctly, executing the following in the shell should result in a working compiler:

```
user@host:~$ ${MINGW32_PREFIX}gcc
gcc: fatal error: no input files
```

12.2.7 Creating Windows export templates

Windows export templates are created by compiling Godot as release, with the following flags:

- (using Mingw32 command prompt, using the bits parameter)

```
C:\godot> scons platform=windows tools=no target=release bits=32
C:\godot> scons platform=windows tools=no target=release_debug bits=32
```

- (using Mingw-w64 command prompt, using the bits parameter)

```
C:\godot> scons platform=windows tools=no target=release bits=64
C:\godot> scons platform=windows tools=no target=release_debug bits=64
```

- (using the Visual Studio command prompts for the correct architecture, notice the lack of bits parameter)

```
C:\godot> scons platform=windows tools=no target=release
C:\godot> scons platform=windows tools=no target=release_debug
```

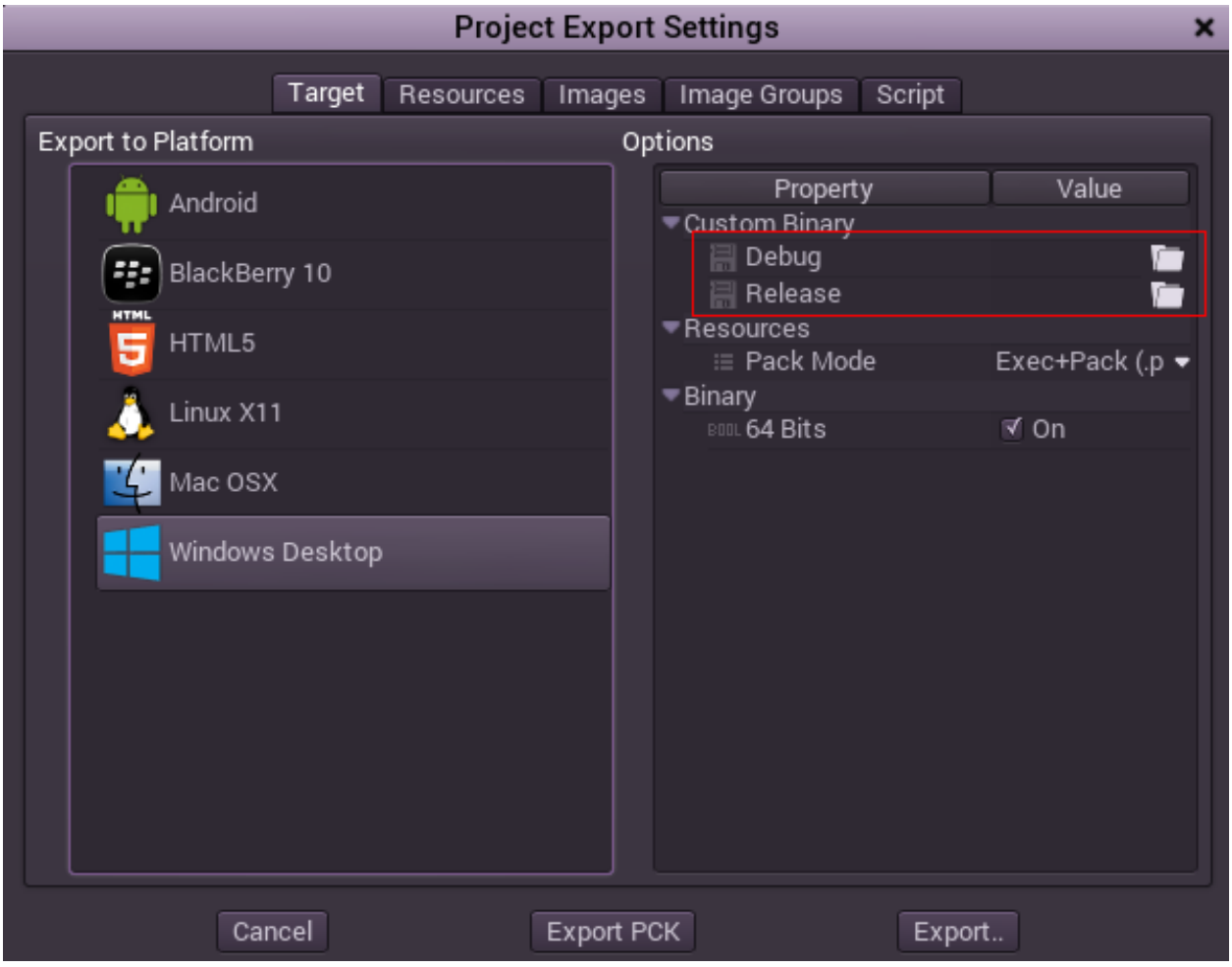
If you plan on replacing the standard templates, copy these to:

```
C:\USERS\YOURUSER\AppData\Roaming\Godot\Templates
```

With the following names:

```
windows_32_debug.exe
windows_32_release.exe
windows_64_debug.exe
windows_64_release.exe
```

However, if you are writing your custom modules or custom C++ code, you might instead want to configure your binaries as custom export templates here:



You don't even need to copy them, you can just reference the resulting files in the `bin\` directory of your Godot source folder, so the next time you build you automatically have the custom templates referenced.

12.3 Compiling for X11 (Linux, *BSD)

12.3.1 Requirements

For compiling under Linux or other Unix variants, the following is required:

- GCC (G++) or Clang
- Python 2.7+ (3.0 is untested as of now)
- SCons build system
- pkg-config (used to detect the dependencies below)
- X11, Xcursor and Xinerama development libraries
- MesaGL development libraries
- ALSA development libraries
- PulseAudio development libraries (for sound support)
- Freetype (for the editor)
- OpenSSL (for HTTPS and TLS)
- libudev-dev (optional, for gamepad support)

Distro-specific oneliners

Fedora	`sudo dnf install scons pkgconfig libX11-devel libXcursor-devel libXinerama-devel \ mesa-libGL-devel alsa-lib-devel pulseaudio-libs-devel freetype-devel openssl-devel libudev-devel`
FreeBSD	`sudo pkg install scons pkg-config xorg-libraries libXcursor xineramaproto libglapi libGLU \ freetype2 openssl`
Mageia	`urpmi scons pkgconfig "pkgconfig(alsa) pkgconfig(freetype2) pkgconfig(glu) pkgconfig(libpulse)" \ "pkgconfig(openssl) pkgconfig(udev) pkgconfig(x11) pkgconfig(xcursor) pkgconfig(xinerama)"`
Ubuntu	`sudo apt-get install build-essential scons pkg-config libx11-dev libxcursor-dev libxinerama-dev \ libgl1-mesa-dev libglu-dev libasound2-dev libpulse-dev libfreetype6-dev libssl-dev libudev-dev`
Arch	`pacman -S scons libxcursor libxinerama mesa glu alsa-lib pulseaudio freetype2`

12.3.2 Compiling

Start a terminal, go to the root dir of the engine source code and type:

```
user@host:~/godot$ scons platform=x11
```

If all goes well, the resulting binary executable will be placed in the "bin" subdirectory. This executable file contains the whole engine and runs without any dependencies. Executing it will bring up the project manager.

If you wish to compile using Clang rather than GCC, use this command:

```
user@host:~/godot$ scons platform=x11 use_llvm=yes
```

12.3.3 Building export templates

To build X11 (Linux, *BSD) export templates, run the build system with the following parameters:

- (32 bits)

```
user@host:~/godot$ scons platform=x11 tools=no target=release bits=32
user@host:~/godot$ scons platform=x11 tools=no target=release_debug bits=32
```

- (64 bits)

```
user@host:~/godot$ scons platform=x11 tools=no target=release bits=64
user@host:~/godot$ scons platform=x11 tools=no target=release_debug bits=64
```

Note that cross compiling for the opposite bits (64/32) as your host platform is not always straight-forward and might need a chroot environment.

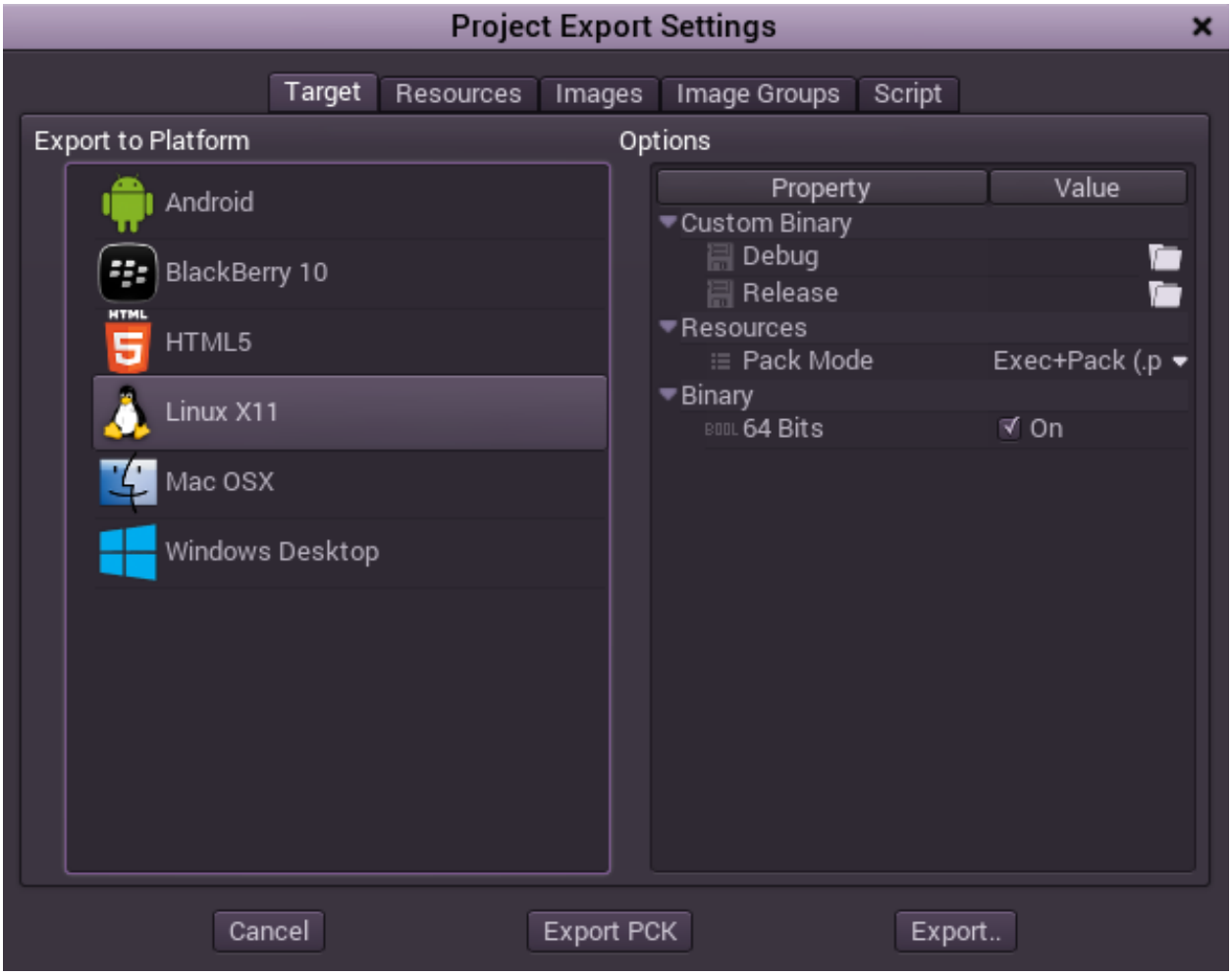
To create standard export templates, the resulting files must be copied to:

```
/home/youruser/.godot/templates
```

and named like this (even for *BSD which is seen as "Linux X11" by Godot):

```
linux_x11_32_debug
linux_x11_32_release
linux_x11_64_debug
linux_x11_64_release
```

However, if you are writing your custom modules or custom C++ code, you might instead want to configure your binaries as custom export templates here:



You don't even need to copy them, you can just reference the resulting files in the bin/ directory of your Godot source folder, so the next time you build you automatically have the custom templates referenced.

12.4 Compiling for OSX

12.4.1 Requirements

For compiling under Linux or other Unix variants, the following is required:

- Python 2.7+ (3.0 is untested as of now)
- SCons build system
- XCode

12.4.2 Compiling

Start a terminal, go to the root dir of the engine source code and type:

```
user@host:~/godot$ scons platform=osx
```

If all goes well, the resulting binary executable will be placed in the "bin" subdirectory. This executable file contains the whole engine and runs without any dependencies. Executing it will bring up the project manager.

To create an .app like in the official builds, you need to use the template located in `tools/Godot.app`. Typically:

```
user@host:~/godot$ cp -r tools/Godot.app .
user@host:~/godot$ mkdir -p Godot.app/Contents/MacOS
user@host:~/godot$ cp bin/godot.osx.opt.tools.64 Godot.app/Contents/MacOS/Godot
user@host:~/godot$ chmod +x Godot.app/Contents/MacOS/Godot
```

12.4.3 Cross-compiling

It is possible to compile for OSX in a Linux environment (and maybe also in Windows with Cygwin). For that you will need OSXCross to be able to use OSX as target. First, follow the instructions to install it:

Clone the *OSXCross repository <https://github.com/tpoechtrager/osxcross>* somewhere on your machine (or download a zip file and extract it somewhere), e.g.:

```
user@host:~$ git clone https://github.com/tpoechtrager/osxcross.git /home/myuser/
↪sources/osxcross
```

1. Follow the instructions to package the SDK: https://github.com/tpoechtrager/osxcross#packaging-the-sdk
2. Follow the instructions to install OSXCross: https://github.com/tpoechtrager/osxcross#installation

After that, you will need to define the `OSXCROSS_ROOT` as the path to the OSXCross installation (the same place where you cloned the repository/extracted the zip), e.g.:

```
user@host:~$ export OSXCROSS_ROOT=/home/myuser/sources/osxcross
```

Now you can compile with SCons like you normally would:

```
user@host:~/godot$ scons platform=osx
```

If you have an OSXCross SDK version different from the one expected by the SCons buildsystem, you can specify a custom one with the `osxcross_sdk` argument:

```
user@host:~/godot$ scons platform=osx osxcross_sdk=darwin15
```

12.5 Compiling for Android

12.5.1 Note

For most cases, using the built-in deployer and export templates is good enough. Compiling the Android APK manually is mostly useful for custom builds or custom packages for the deployer.

Also, you still need to do all the steps mentioned in the *Exporting for Android* tutorial before attempting your custom export template.

12.5.2 Requirements

For compiling under Windows, Linux or OSX, the following is required:

- Python 2.7+ (3.0 is untested as of now).
- SCons build system.
- Android SDK version 19 [Note: Please install all Tools and Extras of sdk manager]
- Android build tools version 19.1
- Android NDK
- Gradle
- OpenJDK 6 or later (or Oracle JDK 6 or later)

12.5.3 Setting up SCons

Set the environment variable ANDROID_HOME to point to the Android SDK.

Set the environment variable ANDROID_NDK_ROOT to point to the Android NDK.

To set those environment variables on Windows, press Windows+R, type "control system", then click on **Advanced system settings** in the left pane, then click on **Environment variables** on the window that appears.

To set those environment variables on Linux, use `export ANDROID_HOME=/path/to/android-sdk` and `export ANDROID_NDK_ROOT=/path/to/android-ndk`.

12.5.4 Compiling

Go to the root dir of the engine source code and type:

```
C:\godot> scons platform=android
```

This should result in a regular .so in `\bin` folder as if it was compiled with flags: `tools=no target=debug`. The resulting file will be huge because it will contain all debug symbols, so for next builds, using `target=release_debug` or `target=release` is recommended.

Copy the .so to the `libs/armeabi` Android folder (or symlink if you are in Linux or OSX). Note: Git does not support empty directories so you will have to create it if it does not exist:

```
C:\godot> mkdir platform/android/java/libs
C:\godot> mkdir platform/android/java/libs/armeabi
```

Then copy:

```
C:\godot> copy bin/libgodot.android.<version>.so platform/android/java/libs/armeabi/
↪libgodot_android.so
```

Or alternatively, if you are under a Unix system you can symlink:

```
user@host:~/godot$ ln -s bin/libgodot.android.<version>.so platform/android/java/libs/
↪armeabi/libgodot_android.so
```

Remember that only *one* of libgodot_android.so must exist for each platform, for each build type (release, debug, etc), it must be replaced.

Note: The file inside `libs/armeabi` must be renamed to **"libgodot_android.so"**, or else unsatisfied link error will happen at runtime.

If you also want to include support for x86 Android, add the following compile flag: `android_arch=x86`, then copy/symlink the resulting binary to the `x86` folder:

```
C:\godot> copy bin/libgodot.android.<version>.x86.so platform/android/java/libs/x86/
↪libgodot_android.so
```

This will create a fat binary that works in both platforms, but will add about 6 megabytes to the APK.

12.5.5 Toolchain

We usually try to keep the Godot Android build code up to date, but Google changes their toolchain versions very often, so if compilation fails due to wrong toolchain version, go to your NDK directory and check the current number, then set the following environment variable:

```
NDK_TARGET (by default set to "arm-linux-androideabi-4.9")
```

12.5.6 Building the APK

To compile the APK, go to the Java folder and run `gradlew.bat build` (or `./gradlew build` on Unix):

```
C:\godot\platform\android\java> gradlew.bat build
```

In the `java/bin` subfolder, the resulting apk can be used as export template.

Note: If you reaaaally feel oldschool, you can copy your entire game (or symlink) to the assets/ folder of the Java project (make sure engine.cfg is in assets/) and it will work, but you lose all the benefits of the export system (scripts are not byte-compiled, textures not converted to Android compression, etc. so it's not a good idea).

12.5.7 Compiling export templates

Godot needs the freshly compiled APK as export templates. It opens the APK, changes a few things inside, adds your file and spits it back. It's really handy! (and required some reverse engineering of the format).

Compiling the standard export templates is done by calling scons with the following arguments:

- (debug)

```
C:\godot> scons platform=android target=release_debug
C:\godot> cp bin/libgodot_android.opt.debug.so platform/android/java/libs/armeabi/
↪libgodot_android.so
C:\godot> cd platform/android/java
C:\godot\platform\android\java> gradlew.bat build
```

Resulting APK is in:

```
platform/android/java/bin/Godot-release-unsigned.apk
```

- (release)

```
C:\godot> scons platform=android target=release
C:\godot> cp bin/libgodot_android.opt.so platform/android/java/libs/armeabi/libgodot_
↪android.so
C:\godot> cd platform/android/java
C:\godot\platform\android\java> gradlew.bat build
```

Resulting APK is in:

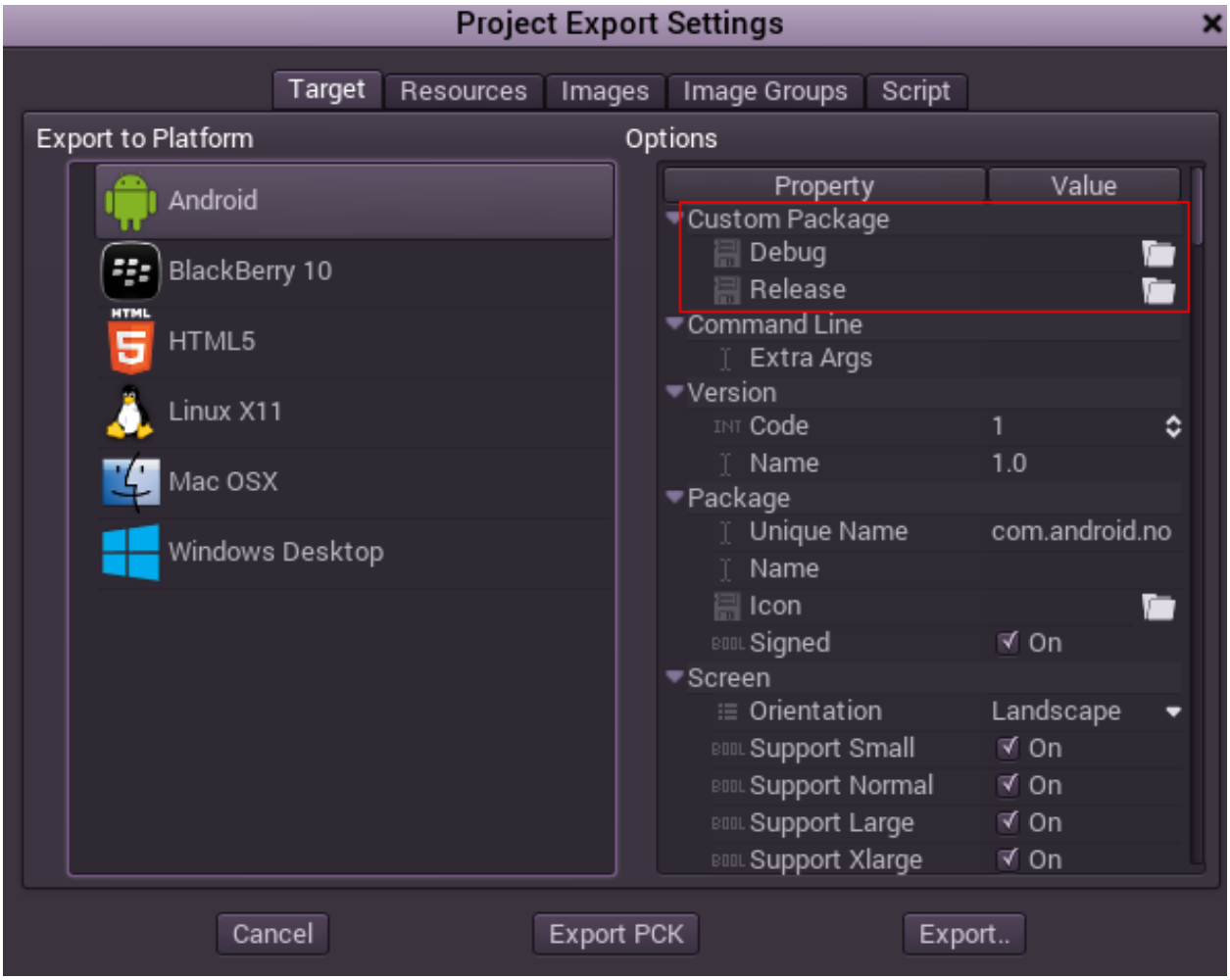
```
platform/android/java/bin/Godot-release-unsigned.apk
```

(same as before)

They must be copied to your templates folder with the following names:

```
android_debug.apk
android_release.apk
```

However, if you are writing your custom modules or custom C++ code, you might instead want to configure your APKs as custom export templates here:



You don't even need to copy them, you can just reference the resulting file in the `bin\` directory of your Godot source folder, so the next time you build you automatically have the custom templates referenced.

12.5.8 Troubleshooting

Application not installed

Android might complain the application is not correctly installed. If so, check the following:

- Check that the debug keystore is properly generated.
- Check that jarsigner is from JDK6.

If it still fails, open a command line and run logcat:

```
C:\android-sdk\platform-tools> adb logcat
```

And check the output while the application is installed. Reason for failure should be presented there.

Seek assistance if you can't figure it out.

Application exits immediately

If the application runs but exits immediately, there might be one of the following reasons:

- libgodot_android.so is not in `libs/armeabi`
- Device does not support armv7 (try compiling yourself for armv6)
- Device is Intel, and apk is compiled for ARM.

In any case, `adb logcat` should also show the cause of the error.

12.6 Compiling for iOS

12.6.1 Requirements

- SCons (you can get it from macports, you should be able to run `scons` in a terminal when installed)
- Xcode with the iOS SDK and the command line tools.

12.6.2 Compiling

Open a Terminal, go to the root dir of the engine source code and type:

```
$ scons p=iphone bin/godot.iphone.debug
```

for a debug build, or:

```
$ scons p=iphone bin/godot.iphone.opt target=release
```

for a release build (check `platform/iphone/detect.py` for the compiler flags used for each configuration).

Alternatively, you can run

```
$ scons p=isim bin/godot.isim.tools
```

for a Simulator executable.

12.6.3 Run

To run on a device or simulator, follow these instructions: *Exporting for iOS*.

Replace or add your executable to the Xcode project, and change the "executable name" property on Info.plist accordingly if you use an alternative build.

12.7 Cross-compiling for iOS on Linux

The procedure for this is somewhat complex and requires a lot of steps, but once you have the environment properly configured it will be easy to compile Godot for iOS anytime you want.

12.7.1 Disclaimer

While it is possible to compile for iOS on a Linux environment, Apple is very restrictive about the tools to be used (specially hardware-wise), allowing pretty much only their products to be used for development. So this is **not official**. However, a statement from Apple in 2010 says they relaxed some of the App Store review guidelines to allow any tool to be used, as long as the resulting binary does not download any code, which means it should be OK to use the procedure described here and cross-compiling the binary.

12.7.2 Requirements

- XCode with the iOS SDK (a dmg image)
- Clang >= 3.5 for your development machine installed and in the `PATH`. It has to be version >= 3.5 to target `arm64` architecture.
- Fuse for mounting and umounting the dmg image.
- darling-dmg, which needs to be built from source. The procedure for that is explained below.
  - For building darling-dmg, you'll need the development packages of the following libraries: fuse, icu, openssl, zlib, bzip2.
- cctools-port for the needed build tools. The procedure for building is quite peculiar and is described below.
  - This also has some extra dependencies: automake, autogen, libtool.

12.7.3 Configuring the environment

darling-dmg

Clone the repository on your machine:

```
$ git clone https://github.com/darlinghq/darling-dmg.git
```

Build it:

```
$ cd darling-dmg
$ mkdir build
$ cd build
$ cmake .. -DCMAKE_BUILD_TYPE=Release
$ make -j 4  # The number is the amount of cores your processor has, for faster build
$ cd ../..
```

Preparing the SDK

Mount the XCode image:

```
$ mkdir xcode
$ ./darling-dmg/build/darling-dmg /path/to/Xcode_7.1.1.dmg xcode
[...]
Everything looks OK, disk mounted
```

Extract the iOS SDK:

```
$ mkdir -p iPhoneSDK/iPhoneOS9.1.sdk
$ cp -r xcode/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/
↪iPhoneOS.sdk/* iPhoneSDK/iPhoneOS9.1.sdk
$ cp -r xcode/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/
↪include/c++/* iPhoneSDK/iPhoneOS9.1.sdk/usr/include/c++
$ fusermount -u xcode  # unmount the image
```

Pack the SDK:

```
$ cd iPhoneSDK
$ tar -cf - * | xz -9 -c - > iPhoneOS9.1.sdk.tar.xz
```

Toolchain

Build cctools:

```
$ git clone https://github.com/tpoechtrager/cctools-port.git
$ cd cctools-port/usage_examples/ios_toolchain
$ ./build.sh /path/iPhoneOS9.1.sdk.tar.xz arm64
```

Copy the tools to a nicer place. Note that the SCons scripts for building will look under `usr/bin` inside the directory you provide for the toolchain binaries, so you must copy to such subdirectory, akin to the following commands:

```
$ mkdir -p /home/user/iostoolchain/usr
$ cp -r target/bin /home/user/iostoolchain/usr/
```

Now you should have the iOS toolchain binaries in `/home/user/iostoolchain/usr/bin`.

12.7.4 Compiling Godot for iPhone

Once you've done the above steps, you should keep two things in your environment: the built toolchain and the iPhoneOS SDK directory. Those can stay anywhere you want since you have to provide their paths to the SCons build command.

For the iPhone platform to be detected, you need the `OSXCROSS_IOS` environment variable defined to anything.

```
$ export OSXCROSS_IOS=anything
```

Now you can compile for iPhone using SCons like the standard Godot way, with some additional arguments to provide the correct paths:

```
$ scons -j 4 platform=iphone bits=32 target=release_debug IPHONESDK="/path/to/
↪iPhoneSDK" IPHONEPATH="/path/to/iostoolchain" ios_triple="arm-apple-darwin11-"
$ scons -j 4 platform=iphone bits=64 target=release_debug IPHONESDK="/path/to/
↪iPhoneSDK" IPHONEPATH="/path/to/iostoolchain" ios_triple="arm-apple-darwin11-"
```

Producing fat binaries

Apple requires a fat binary with both architectures (`armv7` and `arm64`) in a single file. To do this, use the `arm-apple-darwin11-lipo` executable. The following example assumes you are in the root Godot source directory:

```
$ /path/to/iostoolchain/usr/bin/arm-apple-darwin11-lipo -create bin/godot.iphone.opt.
↪debug.32 bin/godot.iphone.opt.debug.64 -output bin/godot.iphone.opt.debug.fat
```

Then you will have an iOS fat binary in `bin/godot.iphone.opt.debug.fat`.

12.8 Compiling for Universal Windows Apps

This page documents the current state of the "winrt" platform, used to support "Windows Store Apps" for Windows 8.1, and Windows Phone 8.1 apps using Microsoft's new "Universal" APIs.

12.8.1 Requirements

- Windows 8
- SCons (see *Compiling for Windows* for more details)
- Visual Studio 2013 for Windows (but *not* "for Windows Desktop"). Tested on "Microsoft Visual Studio Express 2013 for Windows Version 12.0.31101.00 Update 4".

12.8.2 Compiling

The platform can compile binaries for both Windows 8.1 and Windows Phone 8.1. The architecture is decided by the environment variable "PLATFORM".

Windows 8.1

- Open a "VS 2013 x64 Cross Tools Command Prompt"
- The value of environment variable "PLATFORM" should be "x64"
- Run scons with `platform=winrt` from the root of the source tree:

```
C:\godot_source> scons platform=winrt
```

- You should get an executable file inside bin/ named according to your build options, for the architecture "x64", for example "godot.winrt.tools.x64.exe".

Windows Phone 8.1

- Open a "Visual Studio 2012 ARM Phone Tools Command Prompt"
- The value of environment variable "PLATFORM" should be "arm"
- Run scons with `platform=winrt` from the root of the source tree:

```
C:\godot_source> scons platform=winrt
```

- You should get an executable file inside bin/ named according to your build options, for the architecture "arm", for example "godot.winrt.tools.arm.exe".

12.8.3 Running

On Visual studio, create a new project using any of the "Universal App" templates found under Visual C++ -> Store Apps -> Universal Apps. "Blank App" should be fine.

On the "Solution Explorer" box, you should have 3 sections, "App.Windows (Windows 8.1)", "App.WindowsPhone (Windows Phone 8.1)" and "App.Shared". You need to add files to each section:

App.Shared

Add a folder named "game" containing your game content (can be individual files or your data.pck). Remember to set the "Content" property of each file to "True", otherwise your files won't get included in the package.

App.Windows

- Add your windows executable, and all the .dll files found on platform/winrt/x64/bin on the godot source. Remember to also set the "Content" property.
- Find the file "Package.appxmanifest". Right click on it and select "Open with..." then "XML (Text) Editor" from the list.
- Find the "Application" section, and add (or modify) the "Executable" property with the name of your .exe. Example:

```
<Application Id="App" Executable="godot.winrt.tools.x64.exe" EntryPoint="App_Windows.
↪App">
```

App.WindowsPhone

Repeat all the steps from App.Windows, using your arm executable and the dlls found in platform/winrt/arm/bin. Remember to set the "Content" property for all the files.

Use the green "Play" button on the top to run. The drop down menu next to it should let you choose the project (App.Windows or App.WindowsPhone) and the device ("Local Machine", "Device" for an attached phone, etc).

12.8.4 Angle

ANGLE precompiled binaries are provided on platform/winrt/x64 and platform/winrt/arm. They are built from MSOpenTech's "future-dev" branch, found here: https://github.com/MSOpenTech/angle. The visual studio 'solutions' used are found on `projects/winrt/windows/angle.sln` and `projects/winrt/windowsphone/angle.sln`.

12.8.5 What's missing

- Audio
- Semaphores
- Keyboard input
- Proper handling of screen rotation
- Proper handling of other events such as focus lost, back button, etc.

- Packaging and deploying to devices from the editor.
- Adding Angle to our tree and compiling it from there. The same source could also be used to build for Windows (and use Angle instead of native GL, which will be more compatible with graphics hardware)

12.8.6 Packages

This is what we know:

- App packages are documented here: http://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh464929.aspx
- There are 2 command line tools that might be useful, App Packager and SignTool.
- There are a bunch of tools on "powershell" that deal with packages that might be relevant: http://technet.microsoft.com/library/dn448373.aspx
- When running a Windows 8.1 app on "Local Machine" from Visual studio, the app seems to run from an uncompressed directory on the filesystem in an arbitrary location (ie. outside of the proper directory where Apps are installed), but there is some special registry entry made for it, so we know it's possible to skip the packaging step to run locally (in the case of very big games this can be useful).

12.9 Compiling for the Web

12.9.1 Requirements

To compile export templates for the Web, the following is required:

- Emscripten SDK
- Python 2.7+ (3.0 is untested as of now)
- SCons build system

12.9.2 Compiling

Start a terminal and set the environment variable `EMSCRIPTEN_ROOT` to the installation directory of Emscripten:

```
export EMSCRIPTEN_ROOT=~/emsdk/emscripten/master
```

Now go to the root directory of the engine source code and instruct SCons to compile for JavaScript. Specify `target` as either `release` for a release build or `release_debug` for a debug build:

```
scons platform=javascript tools=no target=release
scons platform=javascript tools=no target=release_debug
```

The engine will now be compiled to JavaScript by Emscripten. If all goes well, the resulting file will be placed in the `bin` subdirectory. Its name is `godot.javascript.opt.js` for release or `godot.javascript.opt.debug.js` for debug. Additionally, a file of the same name but with the extension `.html.mem` will be generated.

12.9.3 Building export templates

After compiling, further steps are required to build the template. The actual web export template has the form of a zip file containing at least these 4 files:

1. `godot.js` — This is the file that was just compiled, but under a different name.

   For the release template:

   ```
   cp bin/godot.javascript.opt.js godot.js
   ```

   For the debug template:

   ```
   cp bin/godot.javascript.opt.debug.js godot.js
   ```

2. `godot.mem` — Another file created during compilation. This file initially has the same name as the JavaScript file, except `.js` is replaced by `.html.mem`.

   For the release template:

   ```
   cp bin/godot.javascript.opt.html.mem godot.mem
   ```

   For the debug template:

   ```
   cp bin/godot.javascript.opt.debug.html.mem godot.mem
   ```

3. `godot.html` and
4. `godotfs.js` — Both of these files are located within the Godot Engine repository, under `tools/html_fs/`.

```
cp tools/html_fs/godot.html .
cp tools/html_fs/godotfs.js .
```

Once these 4 files are assembled, zip them up and your export template is ready to go. The correct name for the template file is `javascript_release.zip` for the release template:

```
zip javascript_release godot.js godot.mem godotfs.js godot.html
```

And `javascript_debug.zip` for the debug template:

```
zip javascript_debug godot.js godot.mem godotfs.js godot.html
```

The resulting files must be placed in the `templates` directory in your Godot user directory:

```
mv javascript_release.zip ~/.godot/templates
mv javascript_debug.zip ~/.godot/templates
```

If you are writing custom modules or using custom C++ code, you may want to configure your zip files as custom export templates. This can be done in the export GUI, using the "Custom Package" option. There's no need to copy the templates in this case — you can simply reference the resulting files in your Godot source folder, so the next time you build, the custom templates will already be referenced.

12.9.4 Customizing the HTML page

Rather than the default `godot.html` file from the Godot Engine repository's `tools/html_fs/` directory, it is also possible to use a custom HTML page. This allows drastic customization of the final web presentation.

The JavaScript object `Module` is the page's interface to Emscripten. Check the official documentation for information on how to use it: https://kripken.github.io/emscripten-site/docs/api_reference/module.html

The default HTML page offers a good example to start off with, separating the Emscripten interface logic in the JavaScript `Module` object from the page logic in the `Presentation` object.

When exporting a game, several placeholders in the `godot.html` file are substituted by values dependent on the export:

Placeholder	substituted by
`$GODOT_JS`	Name of the compiled Godot Engine JavaScript file
`$GODOT_FS`	Name of the filesystem access JavaScript file
`$GODOT_MEM`	Name of the memory initialization file
`$GODOT_CANVAS_WIDTH`	Integer specifying the initial display width of the game
`$GODOT_CANVAS_HEIGHT`	Integer specifying the initial display height of the game
`$GODOT_DEBUG_ENABLED`	String `true` if debugging, `false` otherwise
`$GODOT_CONTROLS_ENABLED`	String `true` if `html/controls_enabled` is enabled, `false` otherwise
`$GODOT_HEAD_TITLE`	Title of the page, normally used as content of the HTML `<title>` element
`$GODOT_HEAD_INCLUDE`	Custom string to include just before the end of the HTML `<head>` element
`$GODOT_STYLE_FONT_FAMILY`	CSS format `font-family` to use, without terminating semicolon
`$GODOT_STYLE_INCLUDE`	Custom string to include just before the end of the page's CSS style sheet

The first five of the placeholders listed should always be implemented in the HTML page, since they are important for the correct presentation of the game. The other placeholders are optional.

Finally, the custom HTML page is installed by replacing the existing `godot.html` file in the export template with the new one, retaining the name of the original.

12.10 Batch building templates

The following is almost the same script that we use to build all the export templates that go to the website. If you want to build or roll them yourself, this might be of use.

(note: Apple stuff is missing)

```
#This script is intended to run on Linux or OSX. Cygwin might work.

# if this flag is set, build is tagged as release in the version
# echo $IS_RELEASE_BUILD

#Need to set path to EMScripten
export EMSCRIPTEN_ROOT=/home/to/emscripten

#Build templates

#remove this stuff, will be created anew
rm -rf templates
mkdir -p templates


# Windows 32 Release and Debug

scons -j 4 p=windows target=release tools=no bits=32
cp bin/godot.windows.opt.32.exe templates/windows_32_release.exe
upx templates/windows_32_release.exe
scons -j 4 p=windows target=release_debug tools=no bits=32
cp bin/godot.windows.opt.debug.32.exe templates/windows_32_debug.exe
upx templates/windows_32_debug.exe

# Windows 64 Release and Debug (UPX does not support it yet)
```

```
scons -j 4 p=windows target=release tools=no bits=64
cp bin/godot.windows.opt.64.exe templates/windows_64_release.exe
x86_64-w64-mingw32-strip templates/windows_64_release.exe
scons -j 4 p=windows target=release_debug tools=no bits=64
cp bin/godot.windows.opt.debug.64.exe templates/windows_64_debug.exe
x86_64-w64-mingw32-strip templates/windows_64_debug.exe

# Linux 64 Release and Debug

scons -j 4 p=x11 target=release tools=no bits=64
cp bin/godot.x11.opt.64 templates/linux_x11_64_release
upx templates/linux_x11_64_release
scons -j 4 p=x11 target=release_debug tools=no bits=64
cp bin/godot.x11.opt.debug.64 templates/linux_x11_64_debug
upx templates/linux_x11_64_debug

# Linux 32 Release and Debug

scons -j 4 p=x11 target=release tools=no bits=32
cp bin/godot.x11.opt.32 templates/linux_x11_32_release
upx templates/linux_x11_32_release
scons -j 4 p=x11 target=release_debug tools=no bits=32
cp bin/godot.x11.opt.debug.32 templates/linux_x11_32_debug
upx templates/linux_x11_32_debug

# Server for 32 and 64 bits (always in debug)
scons -j 4 p=server target=release_debug tools=no bits=64
cp bin/godot_server.server.opt.debug.64 templates/linux_server_64
upx templates/linux_server_64
scons -j 4 p=server target=release_debug tools=no bits=32
cp bin/godot_server.server.opt.debug.32 templates/linux_server_32
upx templates/linux_server_32


# Android
**IMPORTANT REPLACE THIS BY ACTUAL VALUES**

export ANDROID_HOME=/home/to/android-sdk
export ANDROID_NDK_ROOT=/home/to/android-ndk

# git does not allow empty dirs, so create those
mkdir -p platform/android/java/libs/armeabi
mkdir -p platform/android/java/libs/x86

#Android Release

scons -j 4 p=android target=release
cp bin/libgodot.android.opt.so platform/android/java/libs/armeabi/libgodot_android.so
./gradlew build
cp platform/android/java/build/outputs/apk/java-release-unsigned.apk templates/
↪android_release.apk

#Android Debug

scons -j 4 p=android target=release_debug
cp bin/libgodot.android.opt.debug.so platform/android/java/libs/armeabi/libgodot_
↪android.so
```

```
./gradlew build
cp platform/android/java/build/outputs/apk/java-release-unsigned.apk templates/
↪android_debug.apk

# EMScripten

scons -j 4 p=javascript target=release
cp bin/godot.javascript.opt.html godot.html
cp bin/godot.javascript.opt.js godot.js
cp tools/html_fs/filesystem.js .
zip javascript_release.zip godot.html godot.js filesystem.js
mv javascript_release.zip templates/

scons -j 4 p=javascript target=release_debug
cp bin/godot.javascript.opt.debug.html godot.html
cp bin/godot.javascript.opt.debug.js godot.js
cp tools/html_fs/filesystem.js .
zip javascript_debug.zip godot.html godot.js filesystem.js
mv javascript_debug.zip templates/

# BlackBerry 10 (currently disabled)

#./path/to/bbndk/bbndk-env.sh
#scons -j 4 platform/bb10/godot_bb10_opt.qnx.armle target=release
#cp platform/bb10/godot_bb10_opt.qnx.armle platform/bb10/bar

#scons -j 4 platform/bb10/godot_bb10.qnx.armle target=release_debug
#cp platform/bb10/godot_bb10.qnx.armle platform/bb10/bar
#cd platform/bb10/bar
#zip -r bb10.zip *
#mv bb10.zip ../../../templates
#cd ../../..


# BUILD ON MAC

[...]

# Build release executables with editor

mkdir -p release

scons -j 4 p=server target=release_debug bits=64
cp bin/godot_server.server.opt.tools.64 release/linux_server.64
upx release/linux_server.64

scons -j 4 p=x11 target=release_debug tools=yes bits=64
cp bin/godot.x11.opt.tools.64 release/godot_x11.64
# upx release/godot_x11.64 -- fails on some linux distros

scons -j 4 p=x11 target=release_debug tools=yes bits=32
cp bin/godot.x11.opt.tools.32 release/godot_x11.32

scons -j 4 p=windows target=release_debug tools=yes bits=64
cp bin/godot.windows.opt.tools.64.exe release/godot_win64.exe
x86_64-w64-mingw32-strip release/godot_win64.exe
#upx release/godot_win64.exe
```

```
scons -j 4 p=windows target=release_debug tools=yes bits=32
cp bin/godot.windows.opt.tools.32.exe release/godot_win32.exe
x86_64-w64-mingw32-strip release/godot_win32.exe
#upx release/godot_win64.exe

[..] # mac stuff

# Update classes.xml (used to generate doc)

cp doc/base/classes.xml .
release/linux_server.64 -doctool classes.xml


cd demos
rm -f godot_demos.zip
zip -r godot_demos *
cd ..

cd tools/export/blender25
zip -r bettercollada *
mv bettercollada.zip ../../..
cd ../../..
```

12.11 Configuring an IDE

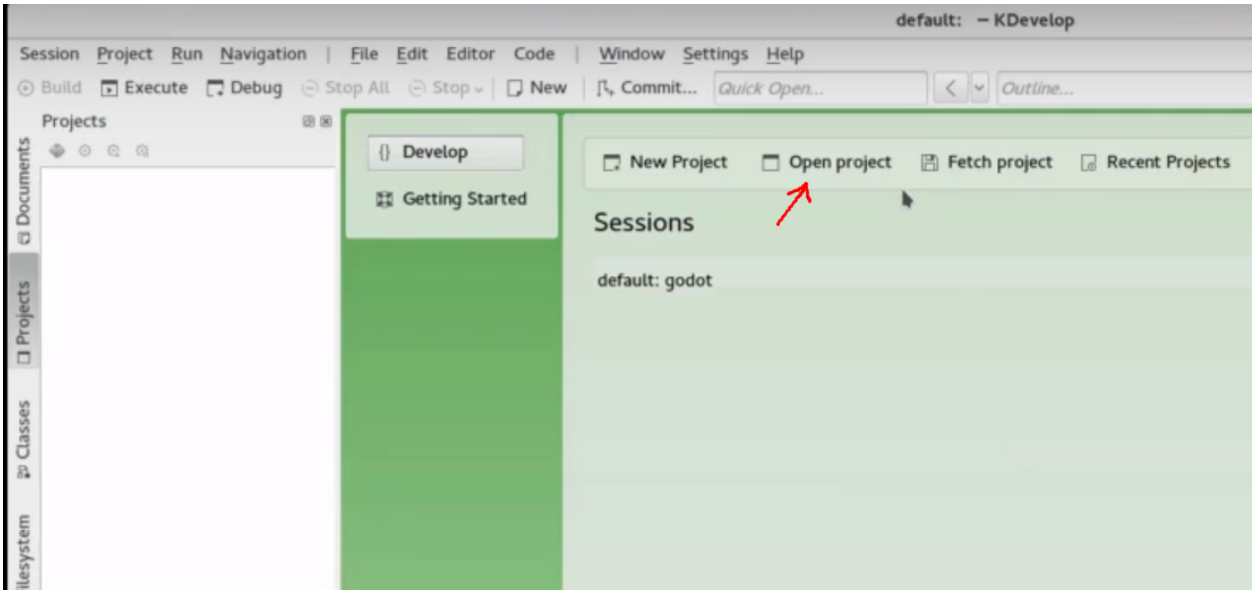
We assume that you already cloned and *compiled* Godot.

12.11.1 Kdevelop

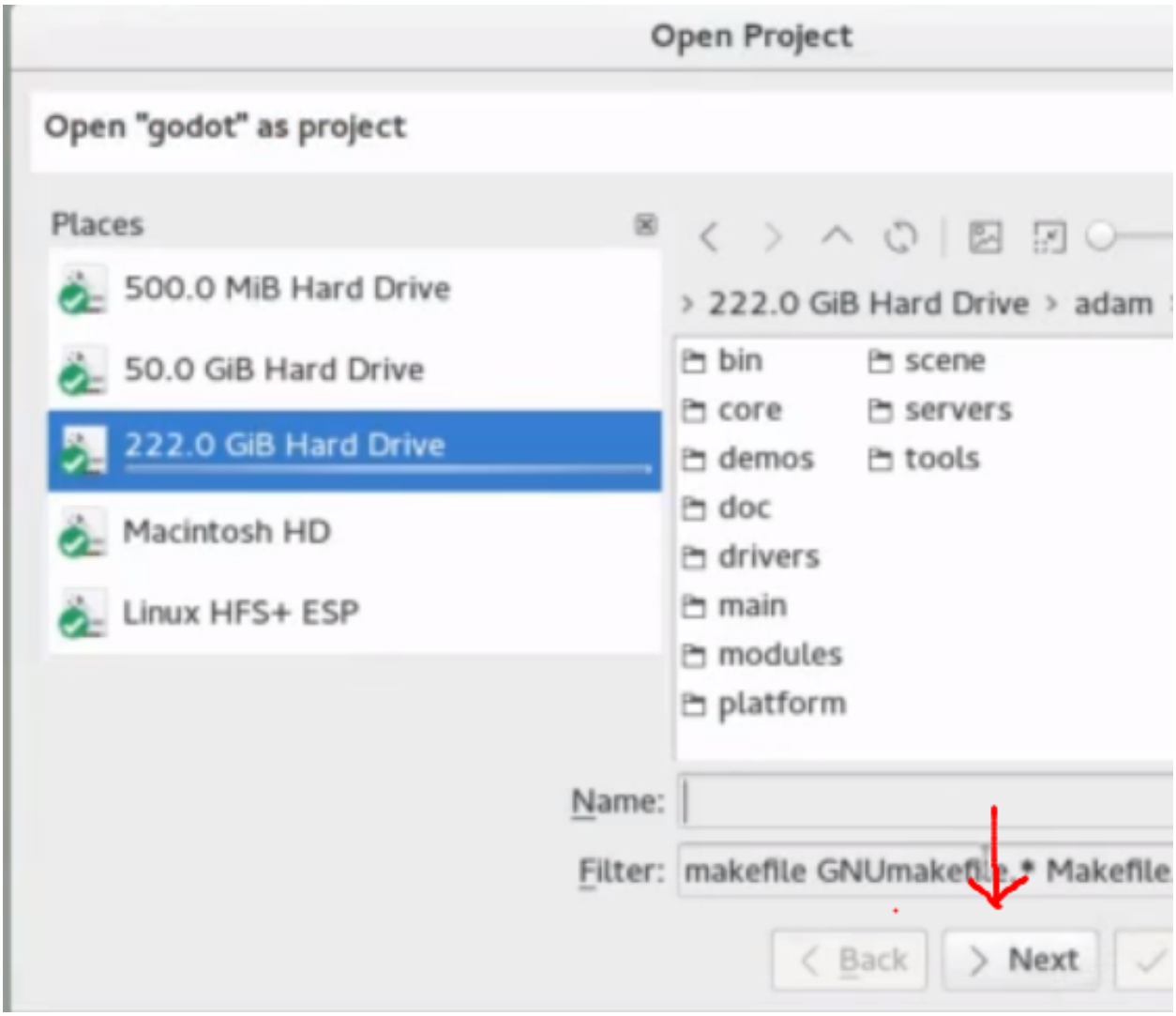
It is a free, open source IDE (Integrated Development Environment) for Linux, Solaris, FreeBSD, Mac OS X and other Unix flavors.

You can find a video tutorial here. Or you may follow this text version tutorial.

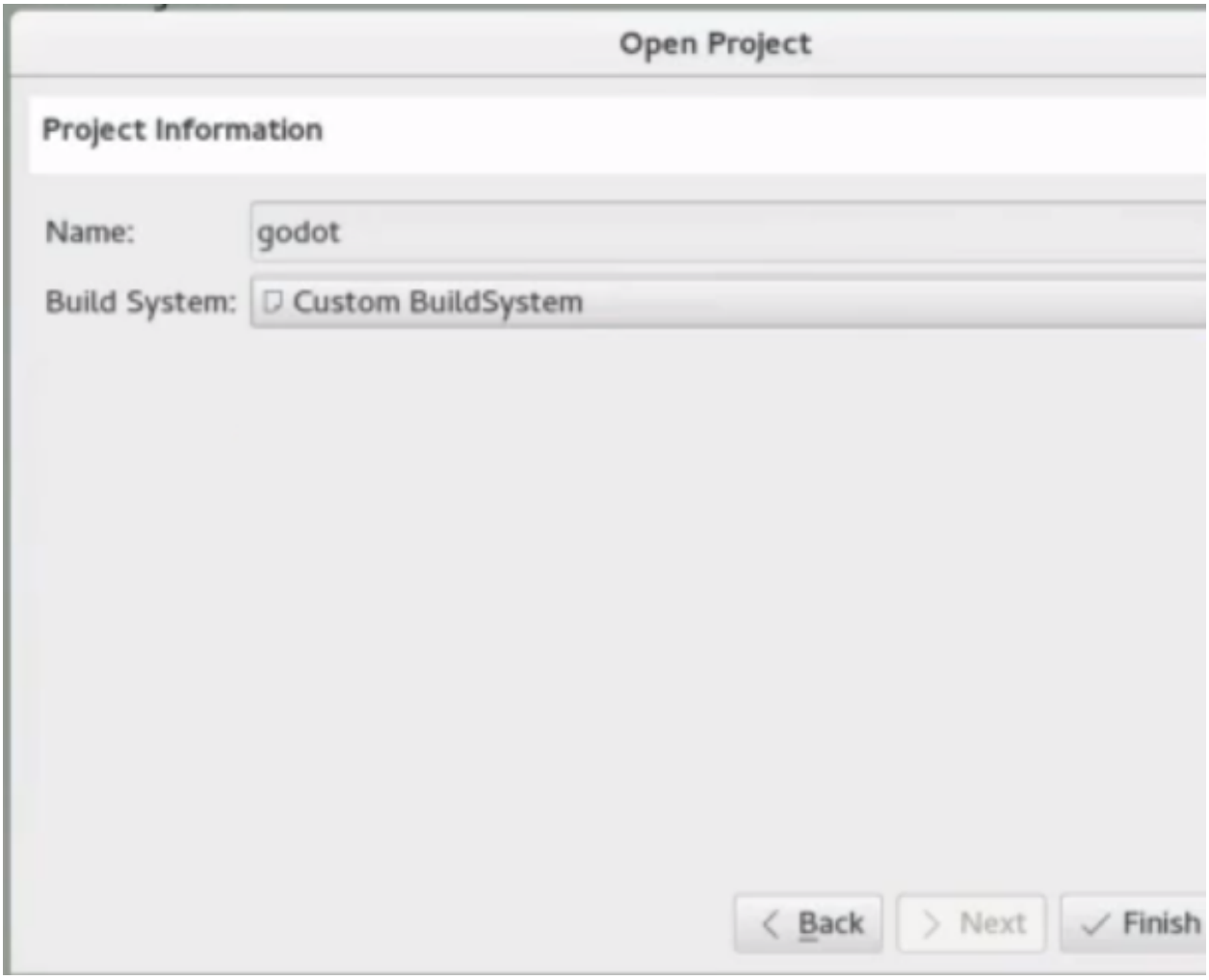
Start by opening Kdevelop and choosing "open project".



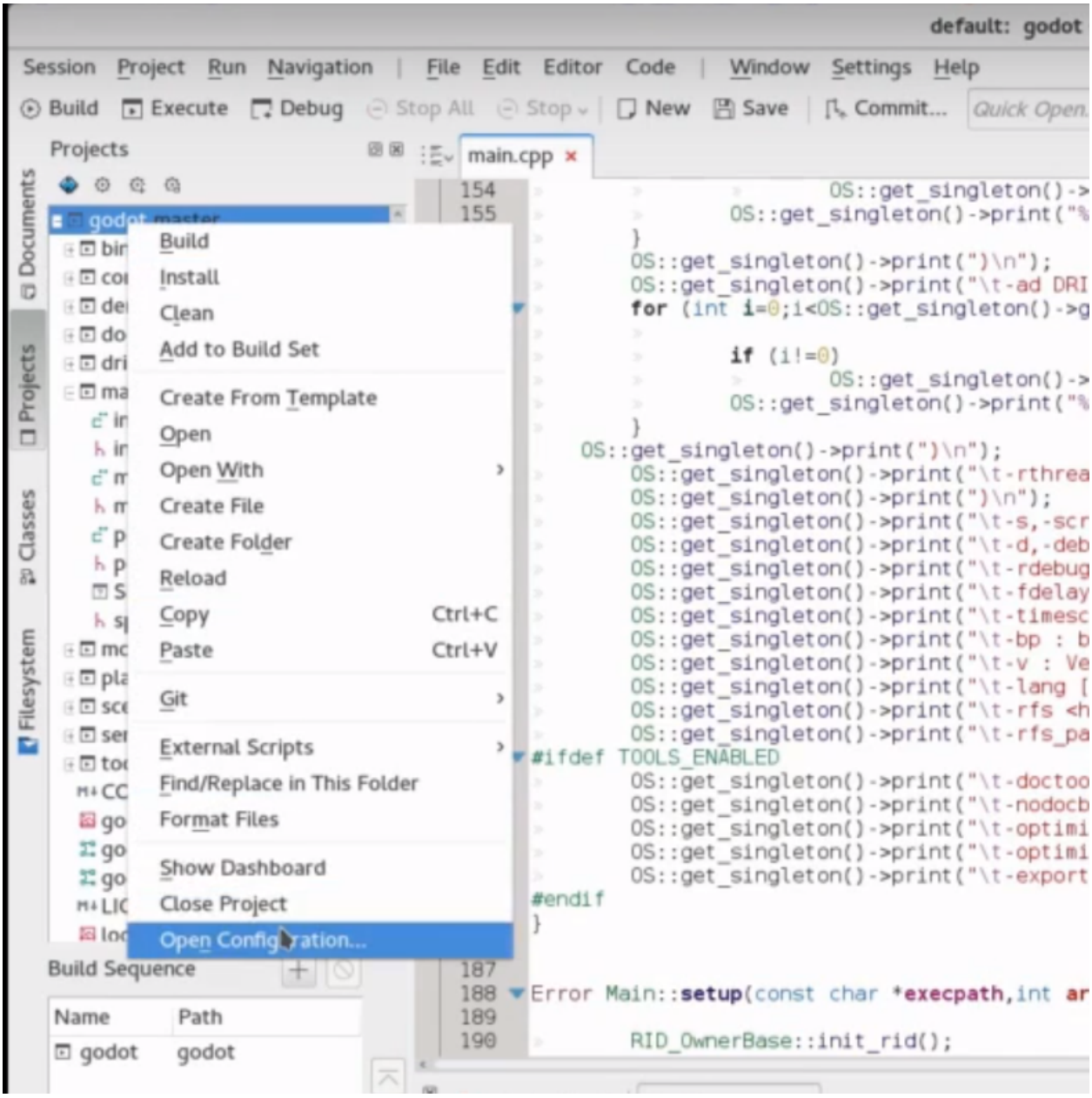
Choose the directory where you cloned Godot.



For the build system, choose “custom build system”.

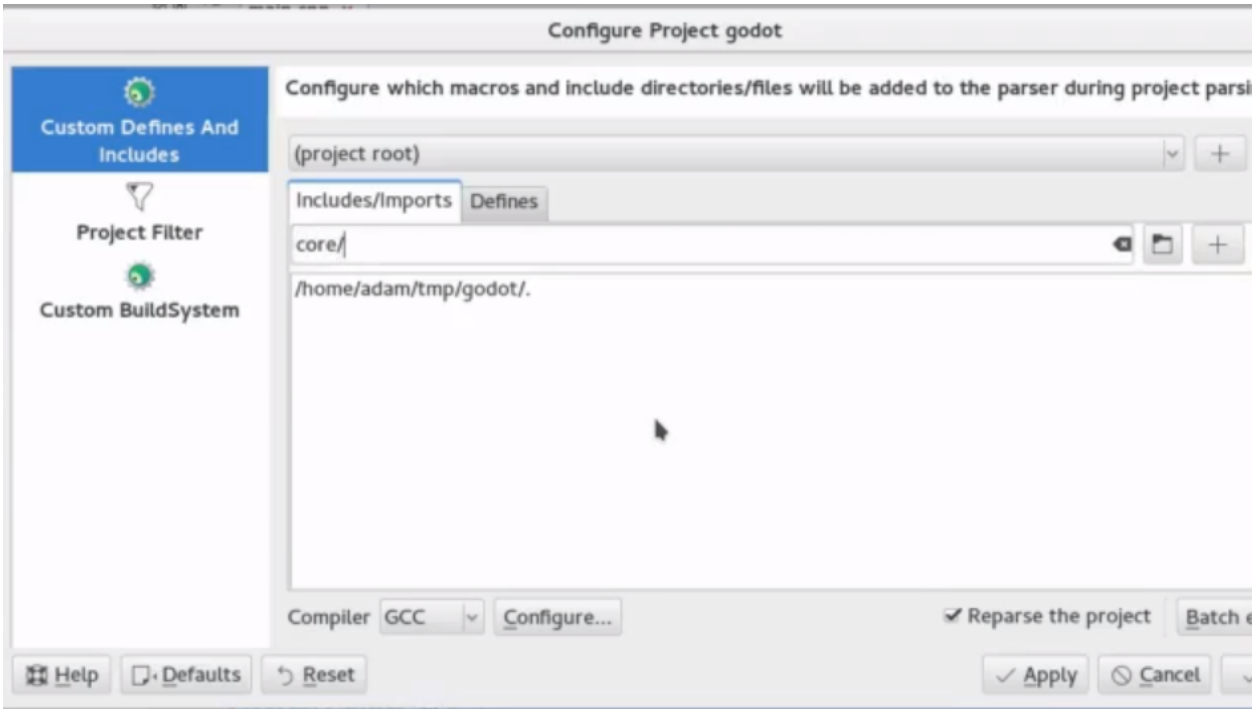


Now that the project has been imported, open the project configuration.

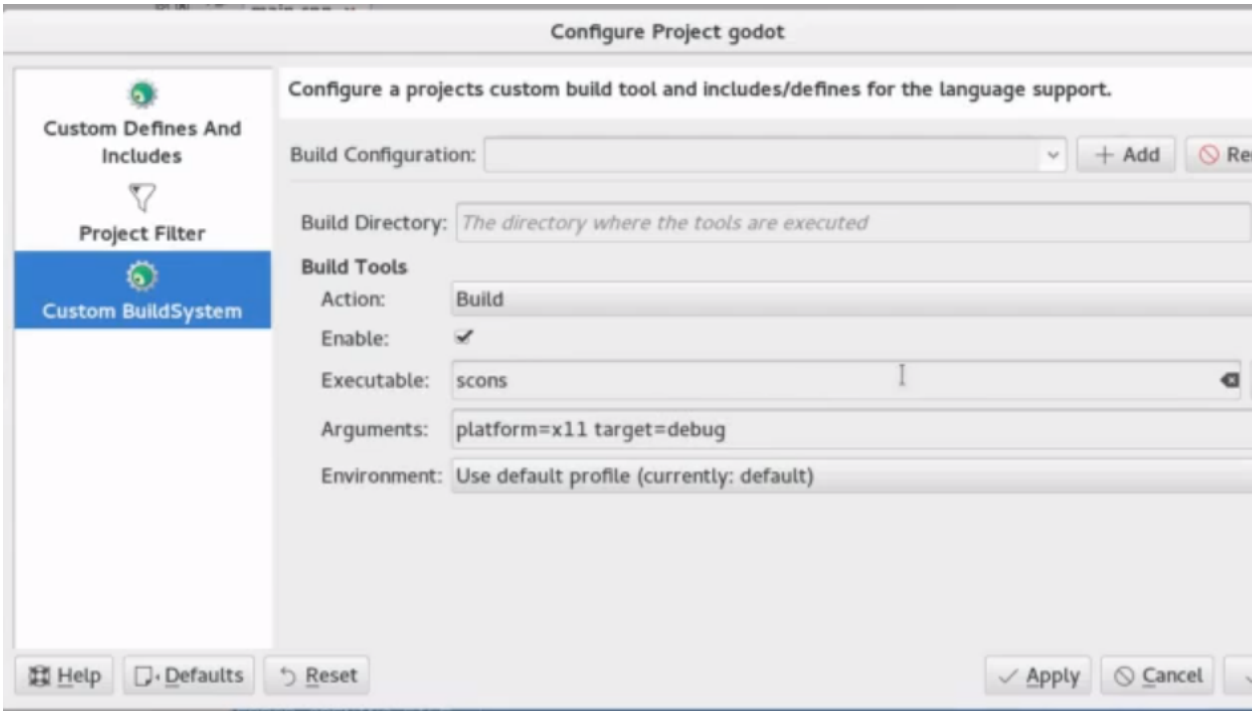


Add the following includes/imports:

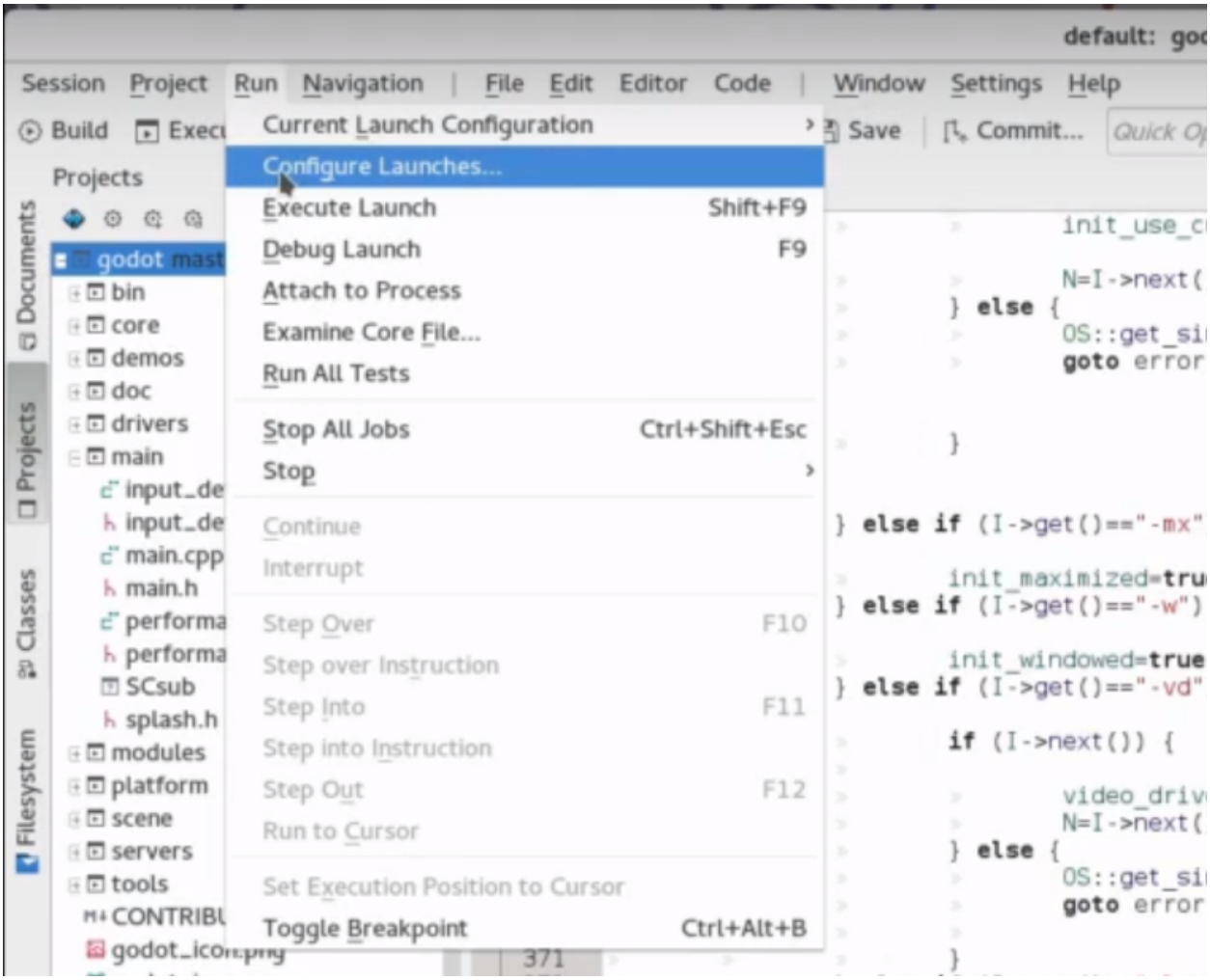
```
.  // a dot to indicate the root of the Godot project
core/
core/os/
core/math/
tools/
drivers/
platform/x11/  // make that platform/osx/ is you're using OS X
```



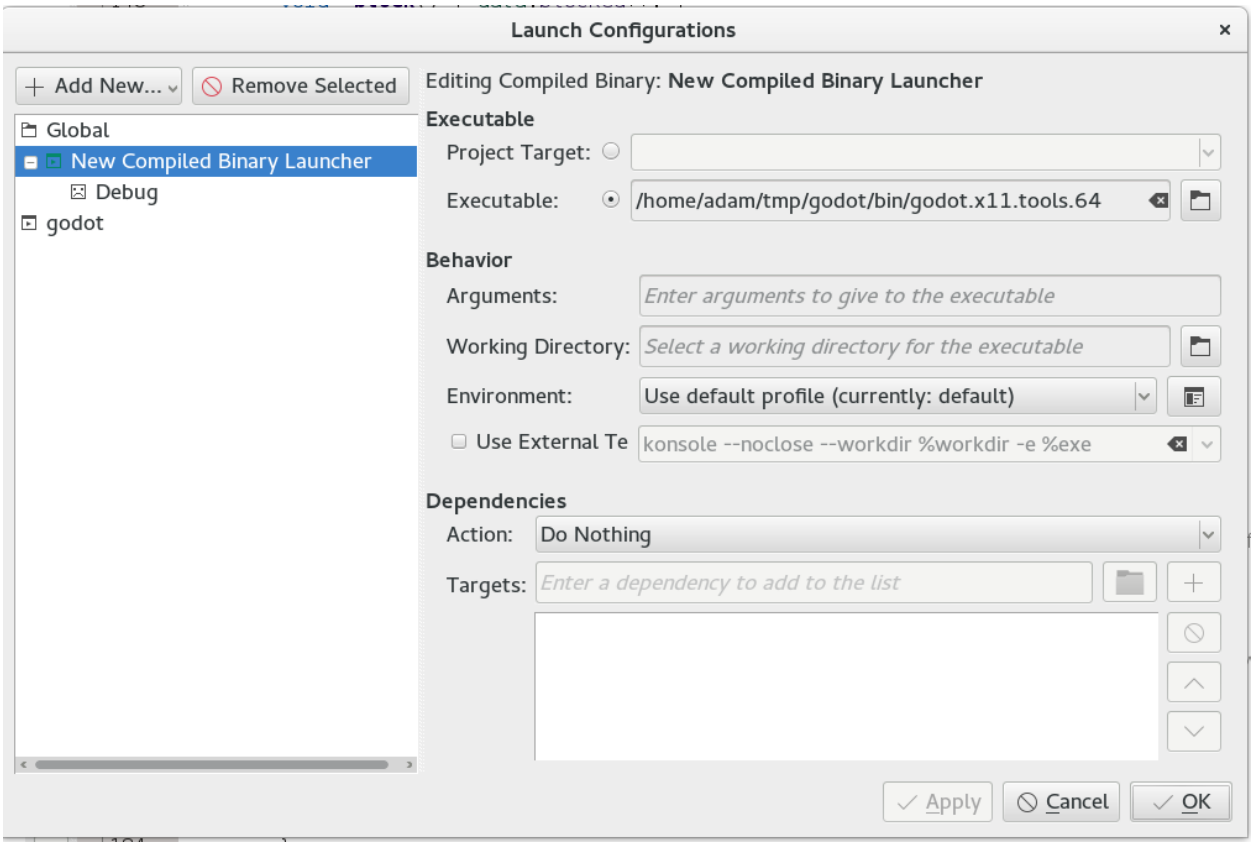
Apply the changes then switch to the "Custom Buildsystem" tab. Leave the build directory blank. Enable build tools and add `scons` as the executable and add `platform=x11 target=debug` (`platform=osx` if you're on OS X).



Next we need to tell KDevelop where to find the binary. From the "run" menu, choose "Configure Launches".



Click “Add new” if no launcher exists. Then add the path to your executable in the executable section. Your executable should be located in the `bin/` sub-directory and should be named something like `godot.x11.tools.64` (the name could be different depending on your platform and depending on your build options).



That's it! Now you should be good to go :)

12.11.2 Eclipse

TODO.

12.11.3 QtCreator

Importing the project

- Choose *New Project -> Import Project -> Import Existing Project*.
- Set the path to your Godot root directory and enter the project name.
- Here you can choose which folders and files will be visible to the project. C/C++ files are added automatically. Potentially useful additions: *.py for buildsystem files, *.java for Android development, *.mm for OSX. Click "Next".
- Click *Finish*.
- Add a line containing `.` to *project_name.files* to get working code completion.

Build and run

Build configuration:

- Click on *Projects* and open the *Build* tab.

- Delete the pre-defined `make` build step.
- Click *Add Build Step -> Custom Process Step*.
- Type `scons` in the *Command* field.
- Fill the *Arguments* field with your compilation options. (e.g.: `p=x11 target=debug -j 4`)

Run configuration:

- Open the *Run* tab.
- Point the *Executable* to your compiled Godot binary.
- If you want to run a specific game or project, point *Working directory* to the game directory.
- If you want to run the editor, add `-e` to the *Command line arguments* field.

12.11.4 Xcode

Project Setup

- Create an `external build` project anywhere
- Set the *Build tool* to the path to scons

Modify Build Target's `Info Tab`:

- Set *Arguments* to something like: platform=osx tools=yes bits=64 target=debug
- Set *Directory* to the path to Godot's source folder. Keep it blank if project is already there.
- You may uncheck *Pass build settings in environment*

Add a Command Line Target:

- Go to `File > New > Target...` and add a new `command line target`
- Name it something so you know not to compile with this target
- e.g. GodotXcodeIndex
- Goto the newly created target's *Build Settings* tab and search for *Header Search Paths*
- Set *Header Search Paths* to an absolute path to Godot's source folder
- Make it recursive by adding two *'s to the end of the path
- e.g. /Users/me/repos/godot-source/**

Add Godot Source to the Project:

- Drag and drop godot source into project file browser.
- `Uncheck` *Create External Build System*
- Click Next
- `Select` *create groups*
- Check off only your command line target in the *Add to targets* section
- Click finish. Xcode will now index the files.
- Grab a cup of coffee... Maybe make something to eat, too
- You should have jump to definition, auto completion, and full syntax highlighting when it is done.

Scheme Setup

Edit Build Scheme of External Build Target:

- Open scheme editor of external build target
- Expand the *Build* menu
- Goto *Post Actions*
- Add a new script `run action`
- Write a script that gives the binary a name that Xcode will recognize
- e.g. ln -f "$SRCROOT"/bin/godot.osx.tools.64 "$SRCROOT"/bin/godot
- Build the external build target

Edit Run Scheme of External Build Target:

- Open the scheme editor again
- `Click Run`
- Set the *Executable* to the file you linked in your post build action script
- Check *Debug executable* if it isn't already
- You can go to *Arguments* tab and add an -e and a -path to a project to debug the editor not the project selection screen

Test It:

- set a breakpoint in platform/osx/godot_main_osx.mm
- it should break at the point!

12.11.5 Other editors (vim, emacs, Atom...)

TODO.

CAPÍTULO 13

Advanced

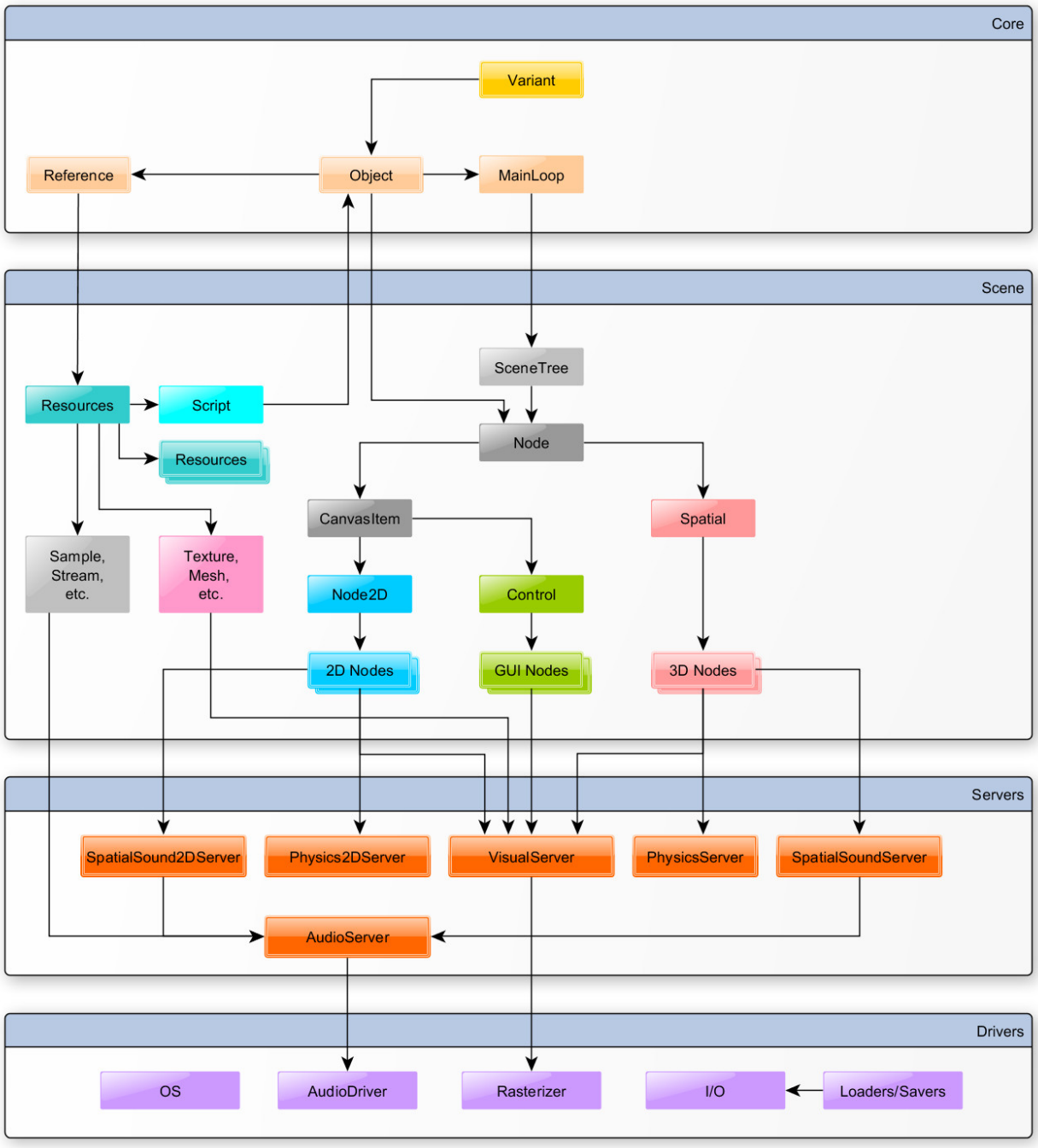
13.1 Developing in C++

13.1.1 Introduction to Godot development

This page is meant to introduce the global organization of Godot Engine's source code, and give useful tips for extending/fixing the engine on the C++ side.

Architecture diagram

The following diagram describes the architecture used by Godot, from the core components down to the abstracted drivers, via the scene structure and the servers.



Debugging the editor with gdb

If you are writing or correcting bugs affecting Godot Engine's editor, remember that the binary will by default run the project manager first, and then only run the editor in another process once you've selected a project. To launch a project directly, you need to run the editor by passing the `-e` argument to Godot Engine's binary from within your project's folder. Typically:

```
$ cd ~/myproject
$ gdb godot
```

```
> run -e
```

Or:

```
$ gdb godot
> run -e -path ~/myproject
```

13.1.2 Core types

Godot has a rich set of classes and templates that compose its core, and everything is built upon them.

This reference will try to list them in order for their better understanding.

Definitions

Godot uses the standard C98 datatypes, such as `uint8_t`, `uint32_t`, `int64_t`, etc. which are nowadays supported by every compiler. Reinventing the wheel for those is not fun, as it makes code more difficult to read.

In general, care is not taken to use the most efficient datatype for a given task unless using large structures or arrays. `int` is used through most of the code unless necessary. This is done because nowadays every device has at least a 32 bits bus and can do such operations in one cycle. It makes code more readable too.

For files or memory sizes, `size_t` is used, which is warranted to be 64 bits.

For Unicode characters, CharType instead of wchar_t is used, because many architectures have 4 bytes long wchar_t, where 2 bytes might be desired. However, by default, this has not been forced and CharType maps directly to wchar_t.

References:

- core/typedefs.h

Memory model

PC is a wonderful architecture. Computers often have gigabytes of RAM, terabytes of storage and gigahertz of CPU, and when an application needs more resources the OS will just swap out the inactive ones. Other architectures (like mobile or consoles) are in general more limited.

The most common memory model is the heap, where an application will request a region of memory, and the underlying OS will try to fit it somewhere and return it. This often works best and is very flexible, but over time and with abuse, this can lead to segmentation.

Segmentation slowly creates holes that are too small for most common allocations, so that memory is wasted. There is a lot of literature about heap and segmentation, so this topic will not be developed further here. Modern operating systems use paged memory, which helps mitigate the problem of segmentation but doesn't solve it.

However, in many studies and tests, it is shown that given enough memory, if the maximum allocation size is below a given threshold in proportion to the maximum heap size and proportion of memory intended to be unused, segmentation will not be a problem over time as it will remain constant. In other words, just leave 10-20 % of your memory free and perform all small allocations and you are fine.

Godot ensures that all objects that can be allocated dynamically are small (less than a few kb at most). But what happens if an allocation is too large (like an image or mesh geometry or large array)? In this case Godot has the option to use a dynamic memory pool. This memory needs to be locked to be accessed, and if an allocation runs out of

memory, the pool will be rearranged and compacted on demand. Depending on the need of the game, the programmer can configure the dynamic memory pool size.

Allocating memory

Godot has many tools for tracking memory usage in a game, specially during debug. Because of this, the regular C and C++ library calls should not be used. Instead, a few other ones are provided.

For C-style allocation, Godot provides a few macros:

```
memalloc()
memrealloc()
memfree()
```

These are equivalent to the usual malloc, realloc, free of the standard library.

For C++-style allocation, special macros are provided:

```
memnew( Class / Class(args) )
memdelete( instance )

memnew_arr( Class , amount )
memdelete_arr( pointer to array )
```

which are equivalent to new, delete, new[] and delete[].

memnew/memdelete also use a little C++ magic and notify Objects right after they are created, and right before they are deleted.

For dynamic memory, the DVector<> template is provided. Just use it like:

```
DVector<int>
```

DVector is just a standard vector class, it can be accessed using the [] operator, but that's probably slow for large amount of accesses (as it has to lock internally). A few helpers exist for this:

```
DVector<int>::Read r = dvector.read()
int someint = r[4]
```

and

```
DVector<int>::Write w = dvector.write()
w[4]=22;
```

respectively. These allow fast read/write from DVectors and keep it locked until they go out of scope.

References:

- core/os/memory.h
- core/dvector.h

Containers

Godot provides also a set of common containers:

- Vector

- List
- Set
- Map

The are very simple and aim to be as minimal as possible, as templates in C++ are often inlined and make the binary size much fatter, both in debug symbols and code. List, Set and Map can be iterated using pointers, like this:

```
for(List<int>::Element *E=somelist.front();E;E=E->next()) {
    print_line(E->get()); //print the element
}
```

The Vector<> class also has a few nice features:

- It does copy on write, so making copies of it is cheap as long as they are not modified.
- It supports multi-threading, by using atomic operations on the reference counter.

References:

- core/vector.h
- core/list.h
- core/set.h
- core/map.h

String

Godot also provides a String class. This class has a huge amount of features, full Unicode support in all the functions (like case operations) and utf8 parsing/extracting, as well as helpers for conversion and visualization.

References:

- core/ustring.h

StringName

StringNames are like a String, but they are unique. Creating a StringName from a string results in a unique internal pointer for all equal strings. StringNames are really useful for using strings as identifier, as comparing them is basically comparing a pointer.

Creation of a StringName (specially a new one) is slow, but comparison is fast.

References:

- core/string_db.h

Math types

There are several linear math types available in the core/math directory, they are basically just that.

References:

- core/math

NodePath

This is a special datatype used for storing paths in a scene tree and referencing them fast.

References:

- core/path_db.h

RID

RIDs are resource IDs. Servers use these to reference data stored in them. RIDs are opaque, meaning that the data they reference can't be accessed directly. RIDs are unique, even for different types of referenced data.

References:

- core/rid.h

13.1.3 Variant class

About

Variant is the most important datatype of Godot, it's the most important class in the engine. A Variant takes up only 20 bytes and can store almost any engine datatype inside of it. Variants are rarely used to hold information for long periods of time, instead they are used mainly for communication, editing, serialization and generally moving data around.

A Variant can:

- Store almost any datatype
- Perform operations between many variants (GDScript uses Variant as it's atomic/native datatype).
- Be hashed, so it can be compared quickly to over variants
- Be used to convert safely between datatypes
- Be used to abstract calling methods and their arguments (Godot exports all it's functions through variants)
- Be used to defer calls or move data between threads.
- Be serialized as binary and stored to disk, or transferred via network.
- Be serialized to text and use it for printing values and editable settings.
- Work as an exported property, so the editor can edit it universally.
- Be used for dictionaries, arrays, parsers, etc.

Basically, thanks to the Variant class, writing Godot itself was a much, much easier task, as it allows for highly dynamic constructs not common of C++ with little effort. Become a friend of Variant today.

References:

- core/variant.h

Dictionary and Array

Both are implemented using variants. A Dictionary can match any datatype used as key to any other datatype. An Array just holds an array of Variants. Of course, a Variant can also hold a Dictionary and an Array inside, making it even more flexible.

Both have a shared mode and a COW mode. Scripts often use them in shared mode (meaning modifications to a container will modify all references to it), or COW mode (modifications will always alter the local copy, making a copy of the internal data if necessary, but will not affect the other copies). In COW mode, Both Dictionary and Array are thread-safe, otherwise a Mutex should be created to lock if multi thread access is desired.

References:

- core/dictionary.h
- core/array.h

13.1.4 Object class

General definition

Object is the base class for almost everything. Most classes in Godot inherit directly or indirectly from it. Objects provide reflection and editable properties, and declaring them is a matter of using a single macro like this.

```
class CustomObject : public Object {

    OBJ_TYPE(CustomObject,Object); // this is required to inherit
};
```

This makes Objects gain a lot of functionality, like for example

```
obj = memnew(CustomObject);
print_line("Object Type: ",obj->get_type()); //print object type

obj2 = obj->cast_to<OtherType>(); // converting between types, this also works␣
↪without RTTI enabled.
```

References:

- core/object.h

Registering an Object

ObjectTypeDB is a static class that holds the entire list of registered classes that inherit from Object, as well as dynamic bindings to all their methods properties and integer constants.

Classes are registered by calling:

```
ObjectTypeDB::register_type<MyCustomType>()
```

Registering it will allow the type to be instanced by scripts, code, or creating them again when deserializing.

Registering as virtual is the same but it can't be instanced.

```
ObjectTypeDB::register_virtual_type<MyCustomType>()
```

Object-derived classes can override the static function `static void _bind_methods()`. When one class is registered, this static function is called to register all the object methods, properties, constants, etc. It's only called once. If an Object derived class is instanced but has not been registered, it will be registered as virtual automatically.

Inside `_bind_methods`, there are a couple of things that can be done. Registering functions is one:

```
ObjectTypeDB::register_method(_MD("methodname","arg1name","arg2name"),&
↪MyCustomMethod);
```

Default values for arguments can be passed in reverse order:

```
ObjectTypeDB::register_method(_MD("methodname","arg1name","arg2name"),&
↪MyCustomType::method,DEFVAL(-1)); //default value for arg2name
```

`_MD` is a macro that converts "methodname" to a StringName for more efficiency. Argument names are used for introspection, but when compiling on release, the macro ignores them, so the strings are unused and optimized away.

Check `_bind_methods` of Control or Object for more examples.

If just adding modules and functionality that is not expected to be documented as thoroughly, the `_MD()` macro can safely be ignored and a string passing the name can be passed for brevity.

References:

- core/object_type_db.h

Constants

Classes often have enums such as:

```
enum SomeMode {
   MODE_FIRST,
   MODE_SECOND
};
```

For these to work when binding to methods, the enum must be declared convertible to int, for this a macro is provided:

```
VARIANT_ENUM_CAST( MyClass::SomeMode ); // now functions that take SomeMode can be␣
↪bound.
```

The constants can also be bound inside `_bind_methods`, by using:

```
BIND_CONSTANT( MODE_FIRST );
BIND_CONSTANT( MODE_SECOND );
```

Properties (set/get)

Objects export properties, properties are useful for the following:

- Serializing and deserializing the object.
- Creating a list of editable values for the Object derived class.

Properties are usually defined by the PropertyInfo() class. Usually constructed as:

```
PropertyInfo(type,name,hint,hint_string,usage_flags)
```

For example:

```
PropertyInfo(Variant::INT,"amount",PROPERTY_HINT_RANGE,"0,49,1",PROPERTY_USAGE_EDITOR)
```

This is an integer property, named "amount", hint is a range, range goes from 0 to 49 in steps of 1 (integers). It is only usable for the editor (edit value visually) but won't be serialized.

Another example:

```
PropertyInfo(Variant::STRING,"modes",PROPERTY_HINT_ENUM,"Enabled,Disabled,Turbo")
```

This is a string property, can take any string but the editor will only allow the defined hint ones. Since no usage flags were specified, the default ones are PROPERTY_USAGE_STORAGE and PROPERTY_USAGE_EDITOR.

There are plenty of hints and usage flags available in object.h, give them a check.

Properties can also work like C# properties and be accessed from script using indexing, but this usage is generally discouraged, as using functions is preferred for legibility. Many properties are also bound with categories, such as "animation/frame" which also make indexing impossible unless using operator [].

From `_bind_methods()`, properties can be created and bound as long as set/get functions exist. Example:

```
ADD_PROPERTY( PropertyInfo(Variant::INT,"amount"), _SCS("set_amount"), _SCS("get_
↪amount") )
```

This creates the property using the setter and the getter. `_SCS` is a macro that creates a StringName efficiently.

Binding properties using `_set`/`_get`/`_get_property_list`

An additional method of creating properties exists when more flexibility is desired (i.e. adding or removing properties on context).

The following functions can be overridden in an Object derived class, they are NOT virtual, DO NOT make them virtual, they are called for every override and the previous ones are not invalidated (multilevel call).

```
void _get_property_info(List<PropertyInfo> *r_props); //return list of properties
bool _get(const StringName& p_property, Variany& r_value) const; //return true if␣
↪property was found
bool _set(const StringName& p_property, const Variany& p_value); //return true if␣
↪property was found
```

This is also a little less efficient since `p_property` must be compared against the desired names in serial order.

Dynamic casting

Godot provides dynamic casting between Object-derived classes, for example:

```
void somefunc(Object *some_obj) {

    Button *button = some_obj->cast_to<Button>();
}
```

If cast fails, NULL is returned. This system uses RTTI, but it also works fine (although a bit slower) when RTTI is disabled. This is useful on platforms where a very small binary size is ideal, such as HTML5 or consoles (with low memory footprint).

Signals

Objects can have a set of signals defined (similar to Delegates in other languages). Connecting to them is rather easy:

```
obj->connect(<signal>,target_instance,target_method)
//for example
obj->connect("enter_tree",this,"_node_entered_tree")
```

The method `_node_entered_tree` must be registered to the class using `ObjectTypeDB::register_method` (explained before).

Adding signals to a class is done in `_bind_methods`, using the `ADD_SIGNAL` macro, for example:

```
ADD_SIGNAL( MethodInfo("been_killed") )
```

References

Reference inherits from Object and holds a reference count. It is the base for reference counted object types. Declaring them must be done using Ref<> template. For example:

```
class MyReference: public Reference {
    OBJ_TYPE( MyReference,Reference );
};

Ref<MyReference> myref = memnew( MyReference );
```

`myref` is reference counted. It will be freed when no more Ref<> templates point to it.

References:

- core/reference.h

Resources:

Resource inherits from Reference, so all resources are reference counted. Resources can optionally contain a path, which reference a file on disk. This can be set with `resource.set_path(path)`. This is normally done by the resource loader though. No two different resources can have the same path, attempt to do so will result in an error.

Resources without a path are fine too.

References:

- core/resource.h

Resource loading

Resources can be loaded with the ResourceLoader API, like this:

```
Ref<Resource> res = ResourceLoader::load("res://someresource.res")
```

If a reference to that resource has been loaded previously and is in memory, the resource loader will return that reference. This means that there can be only one resource loaded from a file referenced on disk at the same time.

- resourceinteractiveloader (TODO)

References:

- core/io/resource_loader.h

Resource saving

Saving a resource can be done with the resource saver API:

```
ResourceSaver::save("res://someresource.res",instance)
```

Instance will be saved. Sub resources that have a path to a file will be saved as a reference to that resource. Sub resources without a path will be bundled with the saved resource and assigned sub-IDs, like "res://someresource.res::1". This also helps to cache them when loaded.

References:

- core/io/resource_saver.h

13.1.5 Custom modules in C++

Modules

Godot allows extending the engine in a modular way. New modules can be created and then enabled/disabled. This allows for adding new engine functionality at every level without modifying the core, which can be split for use and reuse in different modules.

Modules are located in the `modules/` subdirectory of the build system. By default, two modules exist, GDScript (which, yes, is not part of the core engine), and the GridMap. As many new modules as desired can be created and combined, and the SCons build system will take care of it transparently.

What for?

While it's recommended that most of a game is written in scripting (as it is an enormous time saver), it's perfectly possible to use C++ instead. Adding C++ modules can be useful in the following scenarios:

- Binding an external library to Godot (like Bullet, Physx, FMOD, etc).
- Optimize critical parts of a game.
- Adding new functionality to the engine and/or editor.
- Porting an existing game.

- Write a whole, new game in C++ because you can't live without C++.

Creating a new module

Before creating a module, make sure to download the source code of Godot and manage to compile it. There are tutorials in the documentation for this.

To create a new module, the first step is creating a directory inside `modules/`. If you want to maintain the module separately, you can checkout a different VCS into modules and use it.

The example module will be called "sumator", and is placed inside the Godot source tree (`C:\godot` refers to wherever the Godot sources are located):

```
C:\godot> cd modules
C:\godot\modules> mkdir sumator
C:\godot\modules> cd sumator
C:\godot\modules\sumator>
```

Inside we will create a simple sumator class:

```
/* sumator.h */
#ifndef SUMATOR_H
#define SUMATOR_H

#include "reference.h"

class Sumator : public Reference {
    OBJ_TYPE(Sumator,Reference);

    int count;

protected:
    static void _bind_methods();

public:
    void add(int value);
    void reset();
    int get_total() const;

    Sumator();
};

#endif
```

And then the cpp file.

```
/* sumator.cpp */

#include "sumator.h"

void Sumator::add(int value) {

    count+=value;
}

void Sumator::reset() {

    count=0;
```

```
}

int Sumator::get_total() const {

    return count;
}

void Sumator::_bind_methods() {

    ObjectTypeDB::bind_method("add",&Sumator::add);
    ObjectTypeDB::bind_method("reset",&Sumator::reset);
    ObjectTypeDB::bind_method("get_total",&Sumator::get_total);
}

Sumator::Sumator() {
    count=0;
}
```

Then, the new class needs to be registered somehow, so two more files need to be created:

```
register_types.h
register_types.cpp
```

With the following contents:

```
/* register_types.h */

void register_sumator_types();
void unregister_sumator_types();
/* yes, the word in the middle must be the same as the module folder name */
```

```
/* register_types.cpp */

#include "register_types.h"
#include "object_type_db.h"
#include "sumator.h"

void register_sumator_types() {

        ObjectTypeDB::register_type<Sumator>();
}

void unregister_sumator_types() {
   //nothing to do here
}
```

Next, we need to create a `SCsub` file so the build system compiles this module:

```
# SCsub
Import('env')

env.add_source_files(env.modules_sources,"*.cpp") # just add all cpp files to the␣
↪build
```

And finally, the configuration file for the module, this is a simple python script that must be named `config.py`:

```
# config.py

def can_build(platform):
    return True

def configure(env):
    pass
```

The module is asked if it's ok to build for the specific platform (in this case, True means it will build for every platform).

The second function allows to customize the build process for the module, like adding special compiler flags, options, etc. (This can be done in `SCsub`, but `configure(env)` is called at a previous stage). If unsure, just ignore this.

And that's it. Hope it was not too complex! Your module should look like this:

```
godot/modules/sumator/config.py
godot/modules/sumator/sumator.h
godot/modules/sumator/sumator.cpp
godot/modules/sumator/register_types.h
godot/modules/sumator/register_types.cpp
godot/modules/sumator/SCsub
```

You can then zip it and share the module with everyone else. When building for every platform (instructions in the previous sections), your module will be included.

Using the module

Using your newly created module is very easy, from any script you can now do:

```
var s = Sumator.new()
s.add(10)
s.add(20)
s.add(30)
print(s.get_total())
s.reset()
```

And the output will be `60`.

Summing up

As you see, it's really easy to develop Godot in C++. Just write your stuff normally and remember to:

- use `OBJ_TYPE` macro for inheritance, so Godot can wrap it
- use `_bind_methods` to bind your functions to scripting, and to allow them to work as callbacks for signals.

But this is not all, depending what you do, you will be greeted with some surprises.

- If you inherit from *Node* (or any derived node type, such as Sprite), your new class will appear in the editor, in the inheritance tree in the "Add Node" dialog.
- If you inherit from *Resource*, it will appear in the resource list, and all the exposed properties can be serialized when saved/loaded.
- By this same logic, you can extend the Editor and almost any area of the engine.

13.1.6 Creating Android modules

Introduction

Making video games portable is all fine and dandy, until mobile gaming monetization shows up.

This area is complex, usually a mobile game that monetizes needs special connections to a server for stuff such as:

- Analytics
- In-app purchases
- Receipt validation
- Install tracking
- Ads
- Video ads
- Cross-promotion
- In-game soft & hard currencies
- Promo codes
- A/B testing
- Login
- Cloud saves
- Leaderboards and scores
- User support & feedback
- Posting to Facebook, Twitter, etc.
- Push notifications

Oh yeah, developing for mobile is a lot of work. On iOS, you can just write a C++ module and take advantage of the C++/ObjC intercommunication, so this is rather easy.

For C++ developers Java is a pain, the build system is severely bloated and interfacing it with C++ through JNI (Java Native Interface) is more pain that you don't want even for your worst enemy.

Maybe REST?

Most of these APIs allow communication via REST+JSON APIs. Godot has great support for HTTP, HTTPS and JSON, so consider this as an option that works in every platform. Only write the code once and you are set to go.

Popular engines that have half the share of apps published on mobile get special plugins written just for them. Godot does not have that luxury yet. So, if you write a REST implementation of a SDK for Godot, please share it with the community.

Android module

Writing an Android module is similar to *Custom modules in C++*, but needs a few more steps.

Make sure you are familiar with building your own *Android export templates*, as well as creating *Custom modules in C++*.

config.py

In the config.py for the module, some extra functions are provided for convenience. First, it's often wise to detect if android is being built and only enable building in this case:

```
def can_build(plat):
    return plat=="android"
```

If more than one platform can be built (typical if implementing the module also for iOS), check manually for Android in the configure functions:

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # android specific code
```

Java singleton

An android module will usually have a singleton class that will load it, this class inherits from `Godot.SingletonBase`. A singleton object template follows:

```
// package com.android.godot; // for 1.1
package org.godotengine.godot; // for 2.0

public class MySingleton extends Godot.SingletonBase {

    public int myFunction(String p_str) {
        // a function to bind
    }

    static public Godot.SingletonBase initialize(Activity p_activity) {
        return new MySingleton(p_activity);
    }

    public MySingleton(Activity p_activity) {
        //register class name and functions to bind
        registerClass("MySingleton", new String[]{"myFunction"});

        // you might want to try initializing your singleton here, but android
        // threads are weird and this runs in another thread, so you usually have to␣
↪do
        activity.runOnUiThread(new Runnable() {
                public void run() {
                    //useful way to get config info from engine.cfg
                    String key = GodotLib.getGlobal("plugin/api_key");
                    SDK.initializeHere();
                }
        });

    }

    // forwarded callbacks you can reimplement, as SDKs often need them

    protected void onMainActivityResult(int requestCode, int resultCode, Intent data)
↪{}
```

```
    protected void onMainPause() {}
    protected void onMainResume() {}
    protected void onMainDestroy() {}

    protected void onGLDrawFrame(GL10 gl) {}
    protected void onGLSurfaceChanged(GL10 gl, int width, int height) {} //␣
↪singletons will always miss first onGLSurfaceChanged call

}
```

Calling back to Godot from Java is a little more difficult. The instance ID of the script must be known first, this is obtained by calling `get_instance_ID()` on the script. This returns an integer that can be passed to Java.

From Java, use the `calldeferred` function to communicate back with Godot. Java will most likely run in a separate thread, so calls are deferred:

```
GodotLib.calldeferred(<instanceid>, "<function>", new Object[]{param1,param2,etc});
```

Add this singleton to the build of the project by adding the following to config.py:

(Before Version 2.0)

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_module_file("MySingleton.java")
        #env.android_module_file("MySingleton2.java") call again for more files
```

(After Version 2.0)

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_add_java_dir("Directory that contain MySingelton.java")
```

AndroidManifest

Some SDKs need custom values in AndroidManifest.xml. Permissions can be edited from the godot exporter so there is no need to add those, but maybe other functionalities are needed.

Create the custom chunk of android manifest and put it inside the module, add it like this:

(Before Version 2.0)

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
```

```
        env.android_module_file("MySingleton.java")
        env.android_module_manifest("AndroidManifestChunk.xml")
```

(After Version 2.0)

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_add_java_dir("Directory that contain MySingelton.java")
        env.android_add_to_manifest("AndroidManifestChunk.xml")
```

SDK library

So, finally it's time to add the SDK library. The library can come in two flavors, a JAR file or an Android project for ant. JAR is the easiest to integrate, just put it in the module directory and add it:

(Before Version 2.0)

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_module_file("MySingleton.java")
        env.android_module_manifest("AndroidManifestChunk.xml")
        env.android_module_library("MyLibrary-3.1.jar")
```

(After Version 2.0)

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_add_java_dir("Directory that contain MySingelton.java")
        env.android_add_to_manifest("AndroidManifestChunk.xml")
        env.android_add_dependency("compile files('something_local.jar')") # if you␣
↪have a jar, the path is relative to platform/android/java/gradlew, so it will start␣
↪with ../../../modules/module_name/
        env.android_add_maven_repository("maven url") #add a maven url
        env.android_add_dependency("compile 'com.google.android.gms:play-services-
↪ads:8'") #get dependency from maven repository
```

SDK project

When this is an Android project, things usually get more complex. Copy the project folder inside the module directory and configure it:

```
c:\godot\modules\mymodule\sdk-1.2> android -p . -t 15
```

As of this writing, Godot uses minsdk 10 and target sdk 15. If this ever changes, it should be reflected in the manifest template: *AndroidManifest.xml.template <https://github.com/godotengine/godot/blob/master/platform/android/AndroidManifest.xml.template>*

Then, add the module folder to the project:

(Before Version 2.0)

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_module_file("MySingleton.java")
        env.android_module_manifest("AndroidManifestChunk.xml")
        env.android_module_source("sdk-1.2","")
```

(After Version 2.0)

Building

As you probably modify the contents of the module, and modify your .java inside the module, you need the module to be built with the rest of Godot, so compile android normally.

```
c:\godot> scons p=android
```

This will cause your module to be included, the .jar will be copied to the java folder, the .java will be copied to the sources folder, etc. Each time you modify the .java, scons must be called.

Afterwards, just continue the steps for compiling android *Compiling for Android*.

Using the module

To use the module from GDScript, first enable the singleton by adding the following line to engine.cfg (Godot Engine 2.0 and greater):

```
[android]

modules="org/godotengine/godot/MySingleton"
```

For Godot Engine 1.1 is

```
[android]

modules="com/android/godot/MySingleton"
```

More than one singleton module can be enabled by separating with commas:

```
[android]

modules="com/android/godot/MySingleton,com/android/godot/MyOtherSingleton"
```

Then just request the singleton Java object from Globals like this:

```
# in any file

var singleton = null

func _init():
    singleton = Globals.get_singleton("MySingleton")
    print(singleton.myFunction("Hello"))
```

Troubleshooting

(This section is a work in progress, report your problems here!)

Godot crashes upon load

Check `adb logcat` for possible problems, then:

- Make sure libgodot_android.so is in the `libs/armeabi` folder
- Check that the methods used in the Java singleton only use simple Java datatypes, more complex ones are not supported.

Future

Godot has an experimental Java API Wrapper that allows to use the entire Java API from GDScript.

It's simple to use and it's used like this:

```
class = JavaClassWrapper.wrap(<javaclass as text>)
```

This is most likely not functional yet, if you want to test it and help us make it work, contact us through the developer mailing list.

13.2 Data and file formats

13.2.1 Binary serialization API

Introduction

Godot has a simple serialization API based on Variant. It's used for converting data types to an array of bytes efficiently. This API is used in the functions `get_var` and `store_var` of *File* as well as the packet APIs for *PacketPeer*. This format is not used for binary scenes and resources.

Packet specification

The packet is designed to be always padded to 4 bytes. All values are little endian encoded. All packets have a 4 byte header representing an integer, specifying the type of data:

Type	Value
0	null
1	bool
2	integer
3	float
4	string
5	vector2
6	rect2
7	vector3
8	matrix32
9	plane
10	quaternion
11	aabb (rect3)
12	matrix3x3
13	transform (matrix 4x3)
14	color
15	image
16	node path
17	rid (unsupported)
18	object (unsupported)
19	input event
20	dictionary
21	array
22	byte array
23	int array
24	float array
25	string array
26	vector2 array
27	vector3 array
28	color array

Following this is the actual packet contents, which varies for each type of packet:

0: null

1: bool

Offset	Len	Type	Description
4	4	Integer	0 for False, 1 for True

2: integer

Offset	Len	Type	Description
4	4	Integer	Signed, 32-Bit Integer

3: float

Offset	Len	Type	Description
4	4	Float	IEE 754 32-Bits Float

4: string

Offset	Len	Type	Description
4	4	Integer	String Length (in Bytes)
8	X	Bytes	UTF-8 Encoded String

This field is padded to 4 bytes.

5: vector2

Offset	Len	Type	Description
4	4	Float	X Coordinate
8	4	Float	Y Coordinate

6: rect2

Offset	Len	Type	Description
4	4	Float	X Coordinate
8	4	Float	Y Coordinate
12	4	Float	X Size
16	4	Float	Y Size

7: vector3

Offset	Len	Type	Description
4	4	Float	X Coordinate
8	4	Float	Y Coordinate
12	4	Float	Z Coordinate

8: matrix32

Offset	Len	Type	Description
4	4	Float	[0][0]
8	4	Float	[0][1]
12	4	Float	[1][0]
16	4	Float	[1][1]
20	4	Float	[2][0]
24	4	Float	[2][1]

9: plane

Offset	Len	Type	Description
4	4	Float	Normal X
8	4	Float	Normal Y
12	4	Float	Normal Z
16	4	Float	Distance

10: quaternion

Offset	Len	Type	Description
4	4	Float	Imaginary X
8	4	Float	Imaginary Y
12	4	Float	Imaginary Z
16	4	Float	Real W

11: aabb (rect3)

Offset	Len	Type	Description
4	4	Float	X Coordinate
8	4	Float	Y Coordinate
12	4	Float	Z Coordinate
16	4	Float	X Size
20	4	Float	Y Size
24	4	Float	Z Size

12: matrix3x3

Offset	Len	Type	Description
4	4	Float	[0][0]
8	4	Float	[0][1]
12	4	Float	[0][2]
16	4	Float	[1][0]
20	4	Float	[1][1]
24	4	Float	[1][2]
28	4	Float	[2][0]
32	4	Float	[2][1]
36	4	Float	[2][2]

13: transform (matrix 4x3)

Offset	Len	Type	Description
4	4	Float	[0][0]
8	4	Float	[0][1]
12	4	Float	[0][2]
16	4	Float	[1][0]
20	4	Float	[1][1]
24	4	Float	[1][2]
28	4	Float	[2][0]
32	4	Float	[2][1]
36	4	Float	[2][2]
40	4	Float	[3][0]
44	4	Float	[3][1]
48	4	Float	[3][2]

14: color

Offset	Len	Type	Description
4	4	Float	Red (0..1)
8	4	Float	Green (0..1)
12	4	Float	Blue (0..1)
16	4	Float	Alpha (0..1)

15: image

Offset	Len	Type	Description
4	4	Integer	Format (see FORMAT_* in “Image”:class_image
8	4	Integer	Mip-Maps (0 means no mip-maps).
12	4	Integer	Width (Pixels)
16	4	Integer	Height (Pixels)
20	4	Integer	Data Length
24..24+DataLength	1	Byte	Image Data

This field is padded to 4 bytes.

16: node path

Offset	Len	Type	Description
4	4	Integer	String Length, or New Format (val&0x80000000!=0 and NameCount=val&0x7FFFFFFF)

For old format:

Offset	Len	Type	Description
8	X	Bytes	UTF-8 Encoded String

Padded to 4 bytes.

For new format:

Offset	Len	Type	Description
4	4	Integer	Sub-Name Count
8	4	Integer	Flags (absolute: val&1 != 0)

For each Name and Sub-Name

Offset	Len	Type	Description
X+0	4	Integer	String Length
X+4	X	Bytes	UTF-8 Encoded String

Every name string is padded to 4 bytes.

17: rid (unsupported)

18: object (unsupported)

19: input event

20: dictionary

Offset	Len	Type	Description
4	4	Integer	val&0x7FFFFFFF = elements, val&0x80000000 = shared (bool)

Then what follows is, for amount of "elements", pairs of key and value, one after the other, using this same format.

21: array

Offset	Len	Type	Description
4	4	Integer	val&0x7FFFFFFF = elements, val&0x80000000 = shared (bool)

Then what follows is, for amount of "elements", values one after the other, using this same format.

22: byte array

Offset	Len	Type	Description
4	4	Integer	Array Length (Bytes)
8..8+length	1	Byte	Byte (0..255)

The array data is padded to 4 bytes.

23: int array

Offset	Len	Type	Description
4	4	Integer	Array Length (Integers)
8..8+length*4	4	Integer	32 Bits Signed Integer

24: float array

Offset	Len	Type	Description
4	4	Integer	Array Length (Floats)
8..8+length*4	4	Integer	32 Bits IEE 754 Float

25: string array

Offset	Len	Type	Description
4	4	Integer	Array Length (Strings)

For each String:

Offset	Len	Type	Description
X+0	4	Integer	String Length
X+4	X	Bytes	UTF-8 Encoded String

Every string is is padded to 4 bytes.

26: vector2 array

Offset	Len	Type	Description
4	4	Integer	Array Length
8..8+length*8	4	Float	X Coordinate
8..12+length*8	4	Float	Y Coordinate

27: vector3 array

Offset	Len	Type	Description
4	4	Integer	Array Length
8..8+length*12	4	Float	X Coordinate
8..12+length*12	4	Float	Y Coordinate
8..16+length*12	4	Float	Z Coordinate

28: color array

Offset	Len	Type	Description
4	4	Integer	Array Length
8..8+length*16	4	Float	Red (0..1)
8..12+length*16	4	Float	Green (0..1)
8..16+length*16	4	Float	Blue (0..1)
8..20+length*16	4	Float	Alpha (0..1)

13.3 Misc

13.3.1 Frequently asked questions

GDScript? Why your own scripting language? Why not Lua, Javascript, C#, etc.?

The short answer is, we'd rather a programmer does the small effort to learn GDScript so he or she later has a seamless experience, than attracting him or her with a familiar programming language that results in a worse experience. We are OK if you would rather not give Godot a chance because of this, but we strongly encourage you to try it and see the benefits yourself.

The official languages for Godot are GDScript and C++.

GDScript is designed to integrate from the ground to the way Godot works, more than any other language, and is very simple and easy to learn. Takes at most a day or two to get comfortable and it's very easy to see the benefits once you do. Please do the effort to learn GDScript, you will not regret it.

Godot C++ API is also efficient and easy to use (the entire Godot editor is made with this API), and an excellent tool to optimize parts of a project, but trying to use it instead of GDScript for an entire game is, in most cases, a waste of time.

Yes, for more than a decade we tried in the past integrating several VMs (and even shipped games using them), such as Python, Squirrel and Lua (in fact we authored tolua++ in the past, one of the most popular C++ binders). None of them worked as well as GDScript does now.

More information about getting comfortable with GDScript or dynamically typed languages can be found in the *GDScript more efficiently* tutorial.

For the more technically versed, proceed to the next item.

I don't believe you. What are the technical reasons for the item above?

The main reasons are:

1. No good thread support in most script VMs, and Godot uses threads (Lua, Python, Squirrel, JS, AS, etc.)
2. No good class extending support in most script VMs, and adapting to the way Godot works is highly inefficient (Lua, Python, JS)
3. Horrible interface for binding to C++, results in large amount of code, bugs, bottlenecks and general inefficiency (Lua, Python, Squirrel, JS, etc.)
4. No native vector types (vector3,matrix4,etc.), resulting in highly reduced performance when using custom types (Lua, Python, Squirrel, JS, AS, etc.)
5. Garbage collector results in stalls or unnecessarily large memory usage (Lua, Python, JS, AS, etc.)
6. Difficulty to integrate with the code editor for providing code completion, live editing, etc. (all of them). This is very well supported by GDScript.

GDScript was designed to solve the issues above, and performs very well in all the above scenarios. Please learn GDScript, and enjoy from a very smooth integration of scripting with the game engine (yes, it's a rare but very enjoyable situation when things just work). It's worth it, give it a try!

Why is FBX not supported for import?

FBX SDK has a very restrictive license, that is incompatible with the open license provided by Godot.

That said, Godot's Collada support is really good, please use the OpenCollada exporter for maximum compatibility if you are using Maya or 3DS Max. If you are use Blender, take a look at our own Better Collada Exporter.

Will [Insert closed SDK such as PhysX, GameWorks, etc.] be supported in Godot?

No, the aim of Godot is to create a complete open source engine licensed under MIT, so you have complete control about over single piece of it. Open versions of functionality or features from such SDKs may be eventually added though.

That said, because it is open source, and modular, nothing prevents you or anyone else interested into adding those libraries as a module and ship your game using them, as either open or closed source. Everything is allowed.

How should assets be created to handle multiple resolutions and aspect ratios?

This question pops up often and it's probably thanks to the misunderstanding created by Apple when they originally doubled the resolution of their devices. It made people think that having the same assets in different resolutions was a good idea, so many continued towards that path. That originally worked to a point and only for Apple devices, but then several Android and Apple devices with different resolutions and aspect ratios were created, with a very wide range of sizes an DPIs.

The most common and proper way to this is to, instead, is to use a single base resolution for the game and only handle different screen aspects. This is mostly needed for 2D, as in 3D it's just a matter of Cameara XFov or YFov.

1. Choose a single base resolution for your game. Even if there are devices that go up to 2K and devices that go down to 400p, regular hardware scaling in your device will take care of this at little or no performance cost. Most common choices are either near 1080p (1920x1080) or 720p (1280x720). Keep in mind the higher the resolution, the larger your assets, the more memory they will take and the longer the time it will take for loading.
2. Use the stretch options in Godot, 2D stretching with keeping aspect works best. Check the *Resoluciones múltiples* tutorial on how to achieve this.
3. Determine a minimum resolution and then decide if you want your game to stretch vertically or horizontally for different aspect ratios, or whether there is a minimum one and you want black bars to appear instead. This is also explained in the previous step.
4. For user interfaces, use the *anchoring* to determine where controls should stay and move. If UIs are more complex, consider learning about Containers.

And that's it! Your game should work in multiple resolutions.

If there really is a desire to make your game also work on ancient devices with tiny screens (less than 300 pixels in width), you can use the *export option* to shrink images, and set that build to be used for certain screen sizes in the App Store or Google Play.

I have a great idea that will make Godot better, What do you think?

Your idea will most certainly be ignored. Examples of stuff that is ignored by the developers:

- Let's do this because it will make Godot better
- Let's do this in Godot because another game engine does it
- Let's remove this because I think it's not needed
- Let's remove clutter and bloat and make Godot look nicer
- Let's add an alternative workflow for people who prefer it

Developers are always willing to talk to you and listen to your feedback very openly, to an extent rarely seen in open source projects, but they will care mostly about real issues you have while using Godot, not ideas solely based on personal belief. Developers are interested in (for example):

- Your experience using the software and the problems you have, (we care about this much more than ideas on how to improve it)
- The features you would like to see implemented because you need them for your project.
- The concepts that were difficult to understand in order to learn the software.
- The parts of your workflow you would like to see optimized.

Once one of the above points is stated, we can work together on a solution and this is where your ideas and suggestions are most valuable and welcome, they need to be in context of a real issue.

As such, please don't feel that your ideas for Godot are unwelcome. Instead, try to reformulate them as a problem first, so developers and the community have a base ground to discuss first.

Examples of how NOT to state problems generally are like this:

- Certain feature is ugly
- Certain workflow is slow

- Certain feature needs optimization
- Certain aspect of the UI looks cluttered

Associating something with an adjective will not get you much attention and developers will most likely not understand you. Instead, try to reformulate your problem as a story such as:

- I try to move objects around but always end up picking the wrong one
- I tried to make a game like Battlefield but I'm not managing to understand how to get lighting to look the same.
- I always forget which script I was editing, and it takes me too many steps to go back to it.

This will allow you to convey what you are thinking much better and set a common ground for discussion. Please try your best to state your problems as stories to the developers and the community, before discussing any idea.

13.3.2 Command line tutorial

Some developers like using the command line extensively. Godot is designed to be friendly to them, so here are the steps for working entirely from the command line. Given the engine relies on little to no external libraries, initialization times are pretty fast, making it suitable for this workflow.

Path

It is recommended that your godot binary is in your PATH environment variable, so it can be executed easily from any place by typing `godot`. You can do so on Linux by placing the Godot binary in `/usr/local/bin` and making sure it is called `godot`.

Creating a project

Creating a project from the command line is simple, just navigate the shell to the desired place and just make an engine.cfg file exist, even if empty.

```
user@host:~$ mkdir newgame
user@host:~$ cd newgame
user@host:~/newgame$ touch engine.cfg
```

That alone makes for an empty Godot project.

Running the editor

Running the editor is done by executing godot with the `-e` flag. This must be done from within the project directory, or a subdirectory, otherwise the command is ignored and the project manager appears.

```
user@host:~/newgame$ godot -e
```

If a scene has been created and saved, it can be edited later by running the same code with that scene as argument.

```
user@host:~/newgame$ godot -e scene.xml
```

Erasing a scene

Godot is friends with your filesystem, and will not create extra metadata files, simply use `rm` to erase a file. Make sure nothing references that scene, or else an error will be thrown upon opening.

```
user@host:~/newgame$ rm scene.xml
```

Running the game

To run the game, simply execute Godot within the project directory or subdirectory.

```
user@host:~/newgame$ godot
```

When a specific scene needs to be tested, pass that scene to the command line.

```
user@host:~/newgame$ godot scene.xml
```

Debugging

Catching errors in the command line can be a difficult task because they just fly by. For this, a command line debugger is provided by adding `-d`. It works for both running the game or a simple scene.

```
user@host:~/newgame$ godot -d
```

```
user@host:~/newgame$ godot -d scene.xml
```

Exporting

Exporting the project from the command line is also supported. This is specially useful for continuous integration setups. The version of Godot that is headless (server build, no video) is ideal for this.

```
user@host:~/newgame$ godot -export "Linux X11" /var/builds/project
user@host:~/newgame$ godot -export Android /var/builds/project.apk
```

The platform names recognized by the `-export` switch are the same as displayed in the export wizard of the editor. To get a list of supported platforms from the command line, just try exporting to a non-recognized platform and the full listing of platforms your configuration supports will be shown.

To export a debug version of the game, use the `-export_debug` switch instead of `-export`. Their parameters and usage are the same.

Running a script

It is possible to run a simple .gd script from the command line. This feature is specially useful in very large projects, for batch conversion of assets or custom import/export.

The script must inherit from SceneTree or MainLoop.

Here is a simple example of how it works:

```
#sayhello.gd
extends SceneTree

func _init():
    print("Hello!")
    quit()
```

And how to run it:

```
user@host:~/newgame$ godot -s sayhello.gd
Hello!
```

If no engine.cfg exists at the path, current path is assumed to be the current working directory (unless `-path` is specified).

13.3.3 Changing editor fonts

Godot allows changing the font for the editor, and the font for the code editor. Both need to be in .fnt format, so they need to be imported somewhere using the *font import tool*.

Then copy or do whatever you want with the font, as long as the location does not change, and set the relevant property in Editor Settings. Code editor font is refreshed automatically, but the editor needs to be restarted for the new global font to take effect.

13.3.4 Services for iOS

At the moment, there are 2 iOS APIs partially implemented, GameCenter and Storekit. Both use the same model of asynchronous calls explained below.

Asynchronous methods

When requesting an asynchronous operation, the method will look like this:

```
Error purchase(Variant p_params);
```

The parameter will usually be a Dictionary, with the information necessary to make the request, and the call will have 2 phases. First, the method will immediately return an Error value. If the Error is not 'OK', the call operation is completed, with an error probably caused locally (no internet connection, API incorrectly configured, etc). If the error value is 'OK', a response event will be produced and added to the 'pending events' queue. Example:

```
func on_purchase_pressed():
    var result = InAppStore.purchase( { "product_id": "my_product" } )
    if result == OK:
        animation.play("busy") # show the "waiting for response" animation
    else:
        show_error()

# put this on a 1 second timer or something
func check_events():
    while InAppStore.get_pending_event_count() > 0:
        var event = InAppStore.pop_pending_event()
        if event.type == "purchase":
            if event.result == "ok":
                show_success(event.product_id)
```

```
            else:
                show_error()
```

Remember that when a call returns OK, the API will *always* produce an event through the pending_event interface, even if it's an error, or a network timeout, etc. You should be able to, for example, safely block the interface waiting for a reply from the server. If any of the APIs don't behave this way it should be treated as a bug.

The pending event interface consists of 2 methods:

- `get_pending_event_count()` Returns the number of pending events on the queue.
- `Variant pop_pending_event()` Pops the first event from the queue and returns it.

Store Kit

Implemented in platform/iphone/in_app_store.mm

The Store Kit API is accessible through the "InAppStore" singleton (will always be available from gdscript). It is initialized automatically. It has 2 methods for purchasing:

- `Error purchase(Variant p_params);`
- `Error request_product_info(Variant p_params);`

and the pending_event interface

```
int get_pending_event_count();
Variant pop_pending_event();
```

purchase

Purchases a product id through the Store Kit API.

Parameters

Takes a Dictionary as a parameter, with one field, `product_id`, a string with your product id. Example:

```
var result = InAppStore.purchase( { "product_id": "my_product" } )
```

Response event

The response event will be a dictionary with the following fields:

On error:

```
{
  "type": "purchase",
  "result": "error",
  "product_id": "the product id requested"
}
```

On success:

```
{
  "type": "purchase",
  "result": "ok",
  "product_id": "the product id requested"
}
```

request_product_info

Requests the product info on a list of product IDs.

Parameters

Takes a Dictionary as a parameter, with one field, `product_ids`, a string array with a list of product ids. Example:

```
var result = InAppStore.request_product_info( { "product_ids": ["my_product1", "my_
↪product2"] } )
```

Response event

The response event will be a dictionary with the following fields:

```
{
  "type": "product_info",
  "result": "ok",
  "invalid_ids": [ list of requested ids that were invalid ],
  "ids": [ list of ids that were valid ],
  "titles": [ list of valid product titles (corresponds with list of valid ids) ],
  "descriptions": [ list of valid product descriptions ] ,
  "prices": [ list of valid product prices ],
  "localized_prices": [ list of valid product localized prices ],
}
```

Game Center

Implemented in platform/iphone/game_center.mm

The Game Center API is available through the "GameCenter" singleton. It has 6 methods:

- `Error post_score(Variant p_score);`
- `Erroraward_achievement(Variant p_params);`
- `Error reset_achievements();`
- `Error request_achievements();`
- `Error request_achievement_descriptions();`
- `Error show_game_center(Variant p_params);`

plus the standard pending event interface.

post_score

Posts a score to a Game Center leaderboard.

Parameters

Takes a Dictionary as a parameter, with 2 fields:

- `score` a float number
- `category` a string with the category name

Example:

```
var result = GameCenter.post_score( { "value": 100, "category": "my_leaderboard", } )
```

Response event

The response event will be a dictionary with the following fields:

On error:

```
{
  "type": "post_score",
  "result": "error",
  "error_code": the value from NSError::code,
  "error_description": the value from NSError::localizedDescription,
}
```

On success:

```
{
  "type": "post_score",
  "result": "ok",
}
```

award_achievement

Modifies the progress of a Game Center achievement.

Parameters

Takes a Dictionary as a parameter, with 3 fields:

- `name` (string) the achievement name
- `progress` (float) the achievement progress from 0.0 to 100.0 (passed to `GKAchievement::percentComplete`)
- `show_completion_banner` (bool) whether Game Center should display an achievement banner at the top of the screen

Example:

```
var result = award_achievement( { "name": "hard_mode_completed", "progress": 6.1 } )
```

Response event

The response event will be a dictionary with the following fields:

On error:

```
{
  "type": "award_achievement",
  "result": "error",
  "error_code": the error code taken from NSError::code,
}
```

On success:

```
{
  "type": "award_achievement",
  "result": "ok",
}
```

reset_achievements

Clears all Game Center achievements. The function takes no parameters.

Response event

The response event will be a dictionary with the following fields:

On error:

```
{
  "type": "reset_achievements",
  "result": "error",
  "error_code": the value from NSError::code
}
```

On success:

```
{
  "type": "reset_achievements",
  "result": "ok",
}
```

request_achievements

Request all the Game Center achievements the player has made progress on. The function takes no parameters.

Response event

The response event will be a dictionary with the following fields:

On error:

```
{
  "type": "achievements",
  "result": "error",
  "error_code": the value from NSError::code
}
```

On success:

```
{
  "type": "achievements",
  "result": "ok",
  "names": [ list of the name of each achievement ],
  "progress": [ list of the progress made on each achievement ]
}
```

request_achievement_descriptions

Request the descriptions of all existing Game Center achievements regardless of progress. The function takes no parameters.

Response event

The response event will be a dictionary with the following fields:

On error:

```
{
  "type": "achievement_descriptions",
  "result": "error",
  "error_code": the value from NSError::code
}
```

On success:

```
{
  "type": "achievement_descriptions",
  "result": "ok",
  "names": [ list of the name of each achievement ],
  "titles": [ list of the title of each achievement ]
  "unachieved_descriptions": [ list of the description of each achievement when it is␣
↪unachieved ]
  "achieved_descriptions": [ list of the description of each achievement when it is␣
↪achieved ]
  "maximum_points": [ list of the points earned by completing each achievement ]
  "hidden": [ list of booleans indicating whether each achievement is initially␣
↪visible ]
  "replayable": [ list of booleans indicating whether each achievement can be earned␣
↪more than once ]
}
```

show_game_center

Displays the built in Game Center overlay showing leaderboards, achievements, and challenges.

Parameters

Takes a Dictionary as a parameter, with 2 fields:

- `view` (string) (optional) the name of the view to present. Accepts "default", "leaderboards", "achievements", or "challenges". Defaults to "default".
- `leaderboard_name` (string) (optional) the name of the leaderboard to present. Only used when "view" is "leaderboards" (or "default" is configured to show leaderboards). If not specified, Game Center will display the aggregate leaderboard.

Examples:

```
var result = show_game_center( { "view": "leaderboards", "leaderboard_name": "best_
↪time_leaderboard" } )
var result = show_game_center( { "view": "achievements" } )
```

Response event

The response event will be a dictionary with the following fields:

On close:

```
{
  "type": "show_game_center",
  "result": "ok",
}
```

Multi-platform games

When working on a multi-platform game, you won't always have the "GameCenter" singleton available (for example when running on PC or Android). Because the gdscript compiler looks up the singletons at compile time, you can't just query the singletons to see and use what you need inside a conditional block, you need to also define them as valid identifiers (local variable or class member). This is an example of how to work around this in a class:

```
var GameCenter = null # define it as a class member

func post_score(p_score):
    if GameCenter == null:
        return
    GameCenter.post_score( { "value": p_score, "category": "my_leaderboard" } )

func check_events():
    while GameCenter.get_pending_event_count() > 0:
        # do something with events here
        pass

func _ready():
    # check if the singleton exists
    if Globals.has_singleton("GameCenter"):
```

```
        GameCenter = Globals.get_singleton("GameCenter")
        # connect your timer here to the "check_events" function
```

13.3.5 Packaging Godot

Starting with 2.0, Godot has features to make it easier to package it for application repositories.

Default behaviour

Per default, Godot stores all settings and installed templates in a per-user directory. First Godot checks the `APPDATA` environment variable. If it exists, the per-user directory is the "Godot" subdirectory of `$APPDATA`. If `APPDATA` doesn't exist, Godot checks the `HOME` environment variable. The per-user directory is then the ".godot" subdir of `$HOME`.

This meets common operating system standards.

Global template path (Unix only)

The `unix_global_settings_path` build variable is meant for Unix/Linux distro packagers who want to package export templates together with godot. It allows to put the export templates on a hardcoded path.

To use it, pass the desired path via the scons `unix_global_settings_path` build variable when building the editor. The export templates then live at the "templates" subdirectory of the path specified.

Templates installed at the per-user location still override the system wide templates.

This option is only available on unix based platforms.

Self contained mode

The self contained mode can be used to package godot for distribution systems where godot doesn't live at a fixed location. If the godot editor finds a `._sc_` file in the directory the executable is located, godot will continue in "self contained mode".

In self contained mode, all config files are located next to the executable in a directory called `editor_data`. Godot doesn't read or write to the per-user location anymore.

The contents of the `._sc_` file (when not empty) are read with the ConfigFile api (same format as `engine.cfg`, etc). So far it can contain a list of pre-loaded project in this format:

```
[init_projects]
list=["demos/2d/platformer", "demos/2d/isometric"]
```

The paths are relative to the executable location, and will be added to the file `editor_settings.xml` when this is created for the first time.

CAPÍTULO 14

Contributing

14.1 Bug triage guidelines

This page describes the typical workflow of the bug triage team aka bugsquad when handling issues and pull requests on Godot's GitHub repository. It is bound to evolve together with the bugsquad, so do not hesitate to propose modifications to the following guidelines.

14.1.1 Issues management

GitHub proposes three features to manage issues:

- Set one or several labels from a predefined list
- Set one milestone from a predefined list
- Define one contributor as "assignee" among the Godot engine organization members

As the Godot engine organization on GitHub currently has a restricted number of contributors and we are not sure yet to what extent we will use it or OpenProject instead, we will not use assignees extensively for the time being.

Labels

The following labels are currently defined in the Godot repository:

Categories:

- *Archived*: either a duplicate of another issue, or invalid. Such an issue would also be closed.
- *Bug*: describes something that is not working properly.
- *Confirmed*: has been confirmed by at least one other contributor than the bug reporter (typically for *Bug* reports). The purpose of this label is to let developers know which issues are still reproducible when they want to select what to work on. It is therefore a good practice to add in a comment on what platform and what version or commit of Godot the issue could be reproduced; if a developer looks at the issue one year later, the *Confirmed* label may not be relevant anymore.

- *Enhancement*: describes a proposed enhancement to an existing functionality.
- *Feature request*: describes a wish for a new feature to be implemented.
- *High priority*: the issue should be treated in priority (typically critical bugs).
- *Needs discussion*: the issue is not consensual and needs further discussion to define what exactly should be done to address the topic.

The categories are used for general triage of the issues. They can be combined in some way when relevant, e.g. an issue can be labelled *Bug*, *Confirmed* and *High priority* at the same time if it's a critical bug that was confirmed by several users, or *Feature request* and *Needs discussion* if it's a non-consensual feature request, or one that is not precise enough to be worked on.

Topics:

- *Buildsystem*: relates to building issues, either linked to the SCons buildsystem or to compiler peculiarities.
- *Core*: anything related to the core engine. It might be further split later on as it's a pretty big topic.
- *Demos*: relates to the official demos.
- *GDScript*: relates to GDScript.
- *Porting*: relates to some specific platforms.
- *Rendering engine*: relates to the 2D and 3D rendering engines.
- *User interface*: relates to the UI design.

Issues would typically correspond to only one topic, though it's not unthinkable to see issues that fit two bills. The general idea is that there will be specialized contributors teams behind all topics, so they can focus on the issues labelled with their team topic.

Bug reports concerning the website or the documentation should not be filed in GitHub but in the appropriate tool in OpenProject, therefore such issues should be closed and archived once they have been moved to their rightful platform.

Platforms: *Android*, *HTML5*, *iOS*, *Linux*, *OS X*, *Windows*

By default, it is assumed that a given issue applies to all platforms. If one of the platform labels is used, it is the exclusive and the previous assumption doesn't stand anymore (so if it's a bug on e.g. Android and Linux exclusively, select those two platforms).

Milestones

Milestones correspond to planned future versions of Godot for which there is an existing roadmap. Issues that fit in the said roadmap should be filed under the corresponding milestone; if they don't correspond to any current roadmap, they should be set to *Later*. As a rule of thumb, an issue corresponds to a given milestone if it concerns a feature that is new in the milestone, or a critical bug that can't be accepted in any future stable release, or anything that Juan wants to work on right now :)

14.2 Documentation and localisation guidelines

This page describes the rules to follow if you want to contribute Godot Engine by writing documentation or translating existing documentation.

14.2.1 What is a good documentation?

A good documentation is well written in plain English and well-formed sentences. It is clear and objective.

A documentation page is not a tutorial page. We differentiate these concepts by these definitions :

- tutorial : a page aiming at explaining how to use one or more concepts in Godot Editor in order to achieve a specific goal with a learning purpose (ie. “make a simple 2d Pong game”, “apply forces to an object”...)
- documentation : a page describing precisely one and only one concept at the time, if possible exhaustively (ie. the list of methods of the Sprite class for example).

You are free to write the kind of documentation you wish, as long as you respect the following rules.

14.2.2 Create a new wiki page

TODO: Needs review for Sphinx doc

Creating a new documentation page or tutorial page is easy. The following rules must be respected:

- Choose a short and explicit title
- Respect the grammar and orthography
- Make use of the doc_wiki_syntax

Try to structure your page in order to enable users to include a page directly in another page or even forum posts using the include wiki syntax. For example, the syntax to include the page you are reading is :

```
:ref:`the cool documentation guidelines <doc_doc_and_l10n_guidelines>`
```

Titles

Please always begin pages with their title and a reference based on the file name (which should ideally be the same as the page title):

```
.. _insert_your_title_here:

Insert your title here
======================
```

Also, avoid American CamelCase titles: titles’ first word should begin with a capitalized letter, and every following word should not. Thus, this is a good example:

- Insert your title here And this is a bad example:
- Insert Your Title Here

Only project names (and people names) should have capitalized first letter. This is good:

- Starting up with Godot Engine and this is bad:
- Starting up with godot engine

14.2.3 Note for non-English authors

For the moment, we will not pull contributed pages that have no English counterpart. We aim at providing a tool helping translators and writers to determine whether certain languages have pages that do not exist in other languages, but this is not done yet. When it is done, we will open the documentation to new contributions.

Please be patient, we are working on it ;) .

14.2.4 Translating existing pages

New guidelines will come soon !

14.2.5 Important changes and discussions

You are welcome to correct mistakes or styles to respect these guidelines. However, in case of important changes, please do not start a discussion on this page: use the forum, create a new topic with a link to the incriminated page and start discussing there about your remarks.

14.2.6 Licence

This wiki and every page it contains is published under the terms of the Creative Commons BY 3.0 license.

14.3 Updating the class reference

Godot Engine provides an extensive panel of nodes and singletons that you can use with GDScript to develop your games. All those classes are listed and documented in the *class reference*, which is available both in the online documentation and offline from within the engine.

The class reference is however not 100 % complete. Some methods, constants and signals have not been described yet, while others may have their implementation changed and thus need an update. Updating the class reference is a tedious work and the responsibility of the community: if we all partake in the effort, we can fill in the blanks and ensure a good documentation level in no time!

Important notes:

- To coordinate the effort and have an overview of the current status, we use a collaborative pad. Please follow the instructions there to notify the other documentation writers about what you are working on.
- We aim at completely filling the class reference in English first. Once it nears 90-95 % completion, we could start thinking about localisating in other languages.

14.3.1 Workflow for updating the class reference

The source file of the class reference is an XML file in the main Godot source repository on GitHub, doc/base/classes.xml. As of now, it is relatively heavy (more than 1 MB), so it can't be edited online using GitHub's text editor.

The workflow to update the class reference is therefore:

- Fork the upstream repository and clone your fork.
- Edit the `doc/base/classes.xml` file, commit your changes and push to your fork.
- Make a pull request on the upstream repository.

The following section details this workflow, and gives also additional information about how to synchronise the XML template with the current state of the source code, or what should be the formatting of the added text.

Important: The class reference is also *available in the online documentation*, which is hosted alongside the rest of the documentation in the godot-docs git repository. The rST files should however **not be edited directly**, they are generated via a script from the `doc/base/classes.xml` file described above.

14.3.2 Getting started with GitHub

This section describes step-by-step the typical workflow to fork the git repository, or update an existing local clone of your fork, and then prepare a pull request.

Fork Godot Engine

First of all, you need to fork the Godot Engine on your own GitHub repository.

You will then need to clone the master branch of Godot Engine in order to work on the most recent version of the engine, including all of its features.

```
git clone https://github.com/your_name/godot.git
```

Then, create a new git branch that will contain your changes.

```
git checkout -b classref-edit
```

The branch you just created is identical to current master branch of Godot Engine. It already contains a `doc/` folder, with the current state of the class reference. Note that you can set whatever name you want for the branch, `classref-edit` is just an example.

Keeping your local clone up-to-date

If you already have a local clone of your own fork, it might happen pretty fast that it will diverge with the upstream git repository. For example other contributors might have updated the class reference since you last worked on it, or your own commits might have been squashed together when merging your pull request, and thus appear as diverging from the upstream master branch.

To keep your local clone up-to-date, you should first add an `upstream` git *remote* to work with:

```
git remote add upstream https://github.com/godotengine/godot
git fetch upstream
```

You only need to run this once to define this remote. The following steps will have to be run each time you want to sync your branch to the state of the upstream repo:

```
git pull --rebase upstream/master
```

This command would reapply your local changes (if any) on top of the current state of the upstream branch, thus bringing you up-to-date with your own commits on top. In case you have local commits that should be discarded (e.g. if your previous pull request had 5 small commits that were all merged into one bigger commit in the upstream branch), you need to *reset* your branch:

```
git fetch upstream
git reset --hard upstream/master
```

Warning: The above command will reset your branch to the state of the `upstream/master` branch, i.e. it will discard all changes which are specific to your local branch. So make sure to run this *before* making new changes and not afterwards.

Alternatively, you can also keep your own master branch (`origin/master`) up-to-date and create new branches when wanting to commit changes to the class reference:

```
git checkout master
git branch -d my-previous-doc-branch
git pull --rebase upstream/master
git checkout -b my-new-doc-branch
```

In case of doubt, ask for help on our IRC channels, we have some git gurus there.

Updating the documentation template

When classes are modified in the source code, the documentation template might become outdated. To make sure that you are editing an up-to-date version, you first need to compile Godot (you can follow the *Introduction to the buildsystem* page), and then run the following command (assuming 64-bit Linux):

```
./bin/godot.x11.tools.64 -doctool doc/base/classes.xml
```

The doc/base/classes.xml should then be up-to-date with current Godot Engine features. You can then check what changed using the `git diff` command. If there are changes to other classes than the one you are planning to document, please commit those changes first before starting to edit the template:

```
git add doc/base/classes.xml
git commit -m "Sync classes reference template with current code base"
```

You are now ready to edit this file to add stuff.

Note: If this has been done recently by another contributor, you don't forcefully need to go through these steps (unless you know that the class you plan to edit *has* been modified recently).

Push and request a pull of your changes

Once your modifications are finished, push your changes on your GitHub repository:

```
git add doc/base/classes.xml
git commit -m "Explain your modifications."
git push
```

When it's done, you can ask for a Pull Request via the GitHub UI of your Godot fork.

14.3.3 Editing the doc/base/classes.xml file

This file is generated and updated by Godot Engine. It is used by the editor as base for the Help section.

You can edit this file using your favourite text editor. If you use a code editor, make sure that it won't needlessly change the indentation behaviour (e.g. change all tabs to spaces).

Formatting of the XML file

Here is an example with the Node2D class:

```
<class name="Node2D" inherits="CanvasItem" category="Core">
    <brief_description>
    Base node for 2D system.
    </brief_description>
    <description>
    Base node for 2D system. Node2D contains a position, rotation and scale, which is␣
↪used to position and animate. It can alternatively be used with a custom 2D␣
↪transform ([Matrix32]). A tree of Node2Ds allows complex hierarchies for animation␣
↪and positioning.
    </description>
    <methods>
        <method name="set_pos">
            <argument index="0" name="pos" type="Vector2">
            </argument>
            <description>
            Set the position of the 2d node.
            </description>
        </method>
        <method name="set_rot">
            <argument index="0" name="rot" type="float">
            </argument>
            <description>
            Set the rotation of the 2d node.
            </description>
        </method>
        <method name="set_scale">
            <argument index="0" name="scale" type="Vector2">
            </argument>
            <description>
            Set the scale of the 2d node.
            </description>
        </method>
        <method name="get_pos" qualifiers="const">
            <return type="Vector2">
            </return>
            <description>
            Return the position of the 2D node.
            </description>
        </method>
        <method name="get_rot" qualifiers="const">
            <return type="float">
            </return>
            <description>
            Return the rotation of the 2D node.
            </description>
        </method>
        <method name="get_scale" qualifiers="const">
            <return type="Vector2">
            </return>
            <description>
            Return the scale of the 2D node.
            </description>
        </method>
        <method name="rotate">
            <argument index="0" name="degrees" type="float">
            </argument>
            <description>
        </description>
```

```
        </method>
        <method name="move_local_x">
            <argument index="0" name="delta" type="float">
            </argument>
            <argument index="1" name="scaled" type="bool" default="false">
            </argument>
            <description>
            </description>
        </method>
        <method name="move_local_y">
            <argument index="0" name="delta" type="float">
            </argument>
            <argument index="1" name="scaled" type="bool" default="false">
            </argument>
            <description>
            </description>
        </method>
        <method name="get_global_pos" qualifiers="const">
            <return type="Vector2">
            </return>
            <description>
            Return the global position of the 2D node.
            </description>
        </method>
        <method name="set_global_pos">
            <argument index="0" name="arg0" type="Vector2">
            </argument>
            <description>
            </description>
        </method>
        <method name="set_transform">
            <argument index="0" name="xform" type="Matrix32">
            </argument>
            <description>
            </description>
        </method>
        <method name="set_global_transform">
            <argument index="0" name="xform" type="Matrix32">
            </argument>
            <description>
            </description>
        </method>
        <method name="edit_set_pivot">
            <argument index="0" name="arg0" type="Vector2">
            </argument>
            <description>
            </description>
        </method>
    </methods>
    <constants>
    </constants>
</class>
```

As you can see, some methods in this class have no description (i.e. there is no text between their marks). This can also happen for the `description` and `brief_description` of the class, but in our example they are already filled. Let's edit the description of the `rotate()` method:

```
<method name="rotate">
    <argument index="0" name="degrees" type="float">
    </argument>
    <description>
    Rotates the node of a given number of degrees.
    </description>
</method>
```

That's all!

You simply have to write any missing text between these marks:

- <description></description>
- <brief_description></brief_description>
- <constant></constant>
- <member></member>
- <signal></signal>

Describe clearly and shortly what the method does, or what the constant, member variable or signal mean. You can include an example of use if needed. Try to use grammatically correct English, and check the other descriptions to get an impression of the writing style.

For setters/getters, the convention is to describe in depth what the method does in the setter, and to say only the minimal in the getter to avoid duplication of the contents.

Tags available for improved formatting

For more control over the formatting of the help, Godot's XML documentation supports various BBcode-like tags which are interpreted by both the offline in-editor Help, as well as the online documentation (via the reST converter).

Those tags are listed below. See existing documentation entries for more examples of how to use them properly.

Tag	Effect	Usage	Result
[Class]	Link a class	Move the [Sprite].	Move the *Sprite*.
[method methodname]	Link a method of this class	See [method set_pos].	See *set_pos*.
[method Class.methodname]	Link a method of another class	See [method Node2D.set_pos].	See *set_pos*.
[b] [/b]	Bold	Some [b]bold[/b] text.	Some **bold** text.
[i] [/i]	Italic	Some [i]italic[/b] text.	Some *italic* text.
[code] [/code]	Monospace	Some [code]monospace[/code] text.	Some `monospace` text.
[codeblock] [/codeblock]	Multiline preformatted block	*See below.*	*See below.*

The `[codeblock]` is meant to be used for pre-formatted code block, using spaces as indentation (tabs will be removed by the reST converter). For example:

```
[codeblock]
func _ready():
    var sprite = get_node("Sprite")
    print(sprite.get_pos())
[/codeblock]
```

Which would be rendered as:

```
func _ready():
    var sprite = get_node("Sprite")
    print(sprite.get_pos())
```

I don't know what this method does!

Not a problem. Leave it behind for now, and don't forget to notify the missing methods when you request a pull of your changes. Another editor will take care of it.

If you wonder what a method does, you can still have a look at its implementation in Godot Engine's source code on GitHub. Also, if you have a doubt, feel free to ask on the Forums and on IRC (freenode, #godotengine).