
gsage Documentation

Release 1.0

Artem Chernyshev

Oct 12, 2019

CONTENTS

1	Tutorials	3
1.1	Conan Build Instructions	3
1.2	Introduction	4
1.3	Game Configuration	9
1.4	Saving and Loading Game State	12
1.5	Custom Start Script	14
1.6	Entity	16
1.7	Internals	18
1.8	Custom Systems	18
1.9	Lua Bindings	20
1.10	Plugin System	22
1.11	Custom Input Handler	25
1.12	Serialize Classes	25
1.13	Engine Abstraction Layer	27
1.14	Custom Window Manager	30
1.15	Using Another Render System	31
1.16	Ogre Plugin	31
1.17	Dear ImGUI Plugin	32
1.18	LibRocket Plugin	35
1.19	SDL Plugin	35
1.20	Recast Navigation Plugin	36
1.21	Class list	36
1.22	Struct list	48
1.23	Namespace list	48
1.24	File list	55
2	Links	161
	Index	163

Contents:

- [search](#)

TUTORIALS

1.1 Conan Build Instructions

It is possible to build the engine using Conan. It should be relatively simple, though it's necessary to have some prerequisites.

1.1.1 Before you start

Important: You should have Makefile, gcc/XCode/VS2015 and Conan installed

On Windows you can install Anaconda <https://anaconda.org/anaconda/python> . Then you can open Anaconda prompt and install conan:

```
conda install conan
```

Set up VC environment using `vcvarsall.bat` and you're all set.

1.1.2 Build And Run

To build the project run `make build` from the root directory.

Other convenient build targets:

1. `make run` starts the game.
2. `make editor` run editor.
3. `make unit` runs unit tests.
4. `make functional` runs lua functional tests.

If you do not want to use make, you can run several build commands manually:

```
# add gsage conan repository to Conan
conan remote add gsage https://api.bintray.com/conan/gsage/main --insert

# install Conan dependencies
conan install -g cmake -o gsage:with_ogre=1.9.0 -o gsage:with_librocket=True -o with_
lua_version:luajit-2.0.5 --build=outdated .

# build the project
conan build .
```

1.2 Introduction

This tutorial will teach you how to create some basic scene using Gsage engine.

If you have not built the engine yet, check out these build tutorials:

Conan Build Instructions

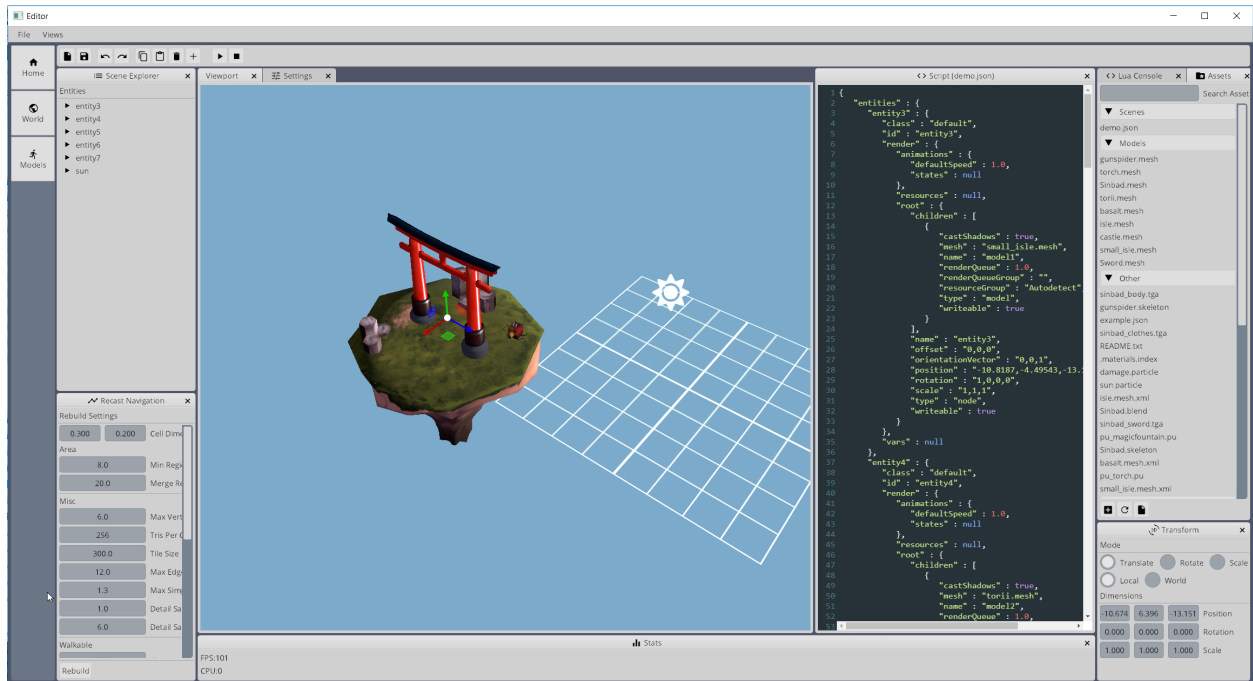
After you build the engine you will have the following file structure:

- `game/game.app/game.exe` executable file should be in the **build/bin** folder. This file is the main game executable.
- All resources are located in `./resources` folder. This folder is symlinked to **game.app/Contents/Resources** on the Mac OS X systems.
- `editor/editor.app/editor.exe` should also be in the game folder.

Old example of game executable in action:



editor:



1.2.1 Modifying Level

Note: Information below is outdated, use editor application instead. Manuals for editor will be available later.

If you open **resources/levels** folder, you will find the file named **exampleLevel.json** there. This file provides some basic level example.

You can modify it the way you want:

1. Add some model to **resources/models** folder (world.mesh).
2. Change entities list, and add this model to position "x: 0, y: 0, z:0":

```
entities: [
...
{
  "id": "hello",
  "render": {
    "root": {
      "position": "0,0,0",
      "rotation": "1,0,-1,0",
      "scale": "1,1,1",
      "children": [{
        "type": "model",
        "mesh": "world.mesh",
        "castShadows": true
      }]
    }
  }
}
...
]
```

If you run **game**, you should see your model on the scene. MovementSystem should automatically calculate walkable areas for this mesh.

For more information about entities format refer to *Entity Format*.

Modifying level demo:

1.2.2 Creating Characters

If you open **resources/bundles/characters** folder, you will see couple entities defined there:

- **sinbad.json** which is controlled by player.
- **ninja.json** which are hostile NPC.

You can use these characters as a reference and create a new character:

1. Grab model with some animations and add it to models folder.
2. Create new json file in the **resources/characters** folder.
3. Now you should be able to configure the character:

3.1. Write Render Component

```
"render":
{
  "resources":
  {
    "Mob":
    [
      "Zip:models/packs/mob.zip" // pack file
      // or you can do
      "FileSystem:models/mob/"
      // or you can omit this section and add resource folder in global settings
    ]
  },
  "root":
  {
    "scale": "1,1,1",
    "rotation": "1,0,1,0",
    "children":
    [
      {
        "type": "model",
        // this is important, otherwise this entity will be treated as part of_
        ↪ level
        "query": "dynamic",
        "name": "modelName",
        "mesh": "mob.mesh",
        "castShadows": true
      }
    ]
  },
  "animations":
  {
    "states":
    {
      // animation is configured as <model_name>.<animation_name>
      "walk": { "body": "modelName.WalkAnimation" },
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    "idle": {"body": "modelName.IdleAnimation"},
    "attack": {"top": "modelName.AttackAnimation"},
  },
  "defaultState": "idle",
  // animation speed. Adjust if necessary
  "defaultSpeed": 1
}
}

```

3.2. Write Movement Component

```

"movement":
{
  // movement speed
  "speed": 10,
  // animation to use for movement
  "moveAnimation": "walk",
  // animation/speed ratio to apply
  "animSpeedRatio": 0.15
},

```

3.3. Write Stats Component

```

// stats component is not limited by param types at all
"stats": {
  "anything": 123
}

```

3.4. Write Script Component

Create script file **characters/scripts/mob.lua**. This file will be used as a setup script.

Startup script should return a function that accepts one parameter. This way the function will get subject entity as the first parameter.

```

-- log information on startup
return function(self) print("I am alive! (" .. self.id .. ")") end

```

Then write behaviour:

```

local function moveRandomly(self, context)
  local position = Vector3:new(
    self:render().position.x + math.random(30) - 15,
    0,
    self:render().position.z + math.random(30) - 15
  )
  self:movement():go(position)
end

local function createTree()
  return Repeat (
    Delay(Leaf(moveRandomly), function() return math.random(700)/100 + 3 end)
  )
end

btree.register("walker", createTree)

```

Saving it as **behaviours/trees/walker.lua**.

Then you will be able to define script component:

```
"script":
{
  "setupScript": "@File:characters/scripts/mob.lua",
  "behavior": "walker"
}
```

4. Add character to scene. Edit **scripts/start.lua** file, add:

```
entity.create("mob")
```

This will create NPC. Or you can use lua console in GsageExe.

Console can be invoked by F9 key. Type the same line there, and NPC will appear.

1.2.3 Modifying UI

UI integration is managed by classes, derived from `Gsage::UIManager` interface. Engine can have several UI libraries running at the same time.

LibRocket

RocketUI plugIn should be installed.

LibRocket looks like a dead project, but it can be configured very nicely using rml and rcss files.

And also it supports lua bindings out of the box, so can have very organic connection with other parts of the engine.

All librocket ui files are stored in the `resources/ui` folder. Currently it's in the mess, but it will be cleaned up very soon.

ImGui

ImGUI plugIn should be installed.

ImGui views can be registered in lua using:

```
-- render method
function render()
  imgui.ShowTestDialog()
end

-- render view class for stateful UI
View = class(function()
end)

function view:__call()
  imgui.ShowTestDialog()
end

local view = View()

imgui.manager.addView("viewID", view)
```

(continues on next page)

(continued from previous page)

```

imgui.manager:addView("viewID2", render)

-- remove
success = imgui.manager:removeView("viewID")

```

1.3 Game Configuration

Gsage engine has one global configuration file. It is loaded by `Gsage::GsageFacade::initialize()`.

This file has initial configs for all engine systems, plugins list, startup script path and data manager configuration.

It can be encoded in json and msgpack and can look like this:

```

{
  "dataManager":
  {
    "extension": "json",
    "charactersFolder": "characters",
    "levelsFolder": "levels",
    "savesFolder": "templates"
  },
  "startupScript": "scripts/start.lua",
  "inputHandler": "ois",
  "systems": ["ogre"],

  "plugins":
  [
    "PlugIns/Plugin_ParticleUniverseFactory"
  ],

  "packager": {
    "deps": [
      "tween"
    ]
  },

  "windowManager": {
    "type": "SDL"
  },

  "render":
  {
    "pluginsFile": "plugins.cfg",
    "configFile": "ogreConfig.cfg",
    "globalResources":
    {
      "General":
      [
        "FileSystem:materials/",
        "FileSystem:programs/",
        "FileSystem:particles/PU",
        "FileSystem:particles/Ogre"
      ],
      "Rocket":
      [

```

(continues on next page)

(continued from previous page)

```
        "FileSystem:fonts/",
        "FileSystem:ui/"
    ]
},

"window":
{
    "name": "Game",
    "width": 1280,
    "height": 800,
    "params":
    {
        "FSAA": 4,
        "displayFrequency": 50,
        "vsync": false
    }
}
}
```

1.3.1 Gsage::GameDataManager Config

Gsage::GameDataManager settings are stored in "dataManager" section. There can be 4 kinds of variables:

- "extension" extension of all data files. "json" is recommended.
- "charactersFolder" folder where to search characters construction data.
- "levelsFolder" folder where to search levels data.
- "savesFolder" folder where to keep saves.

1.3.2 Plugins List

"plugins" stores list of plugins to be loaded on engine startup. Plugins are c++ dynamic libraries: *.so/*.dylib/*.dll.

Note: Plugin should be specified without extension. Engine will add appropriate extension for each platform itself.

Each defined plugin will be installed in the order defined in the list.

1.3.3 Systems Configs

Systems can be either registered statically, by calling Gsage::GsageFacade::addSystem() or they can be created by SystemFactory in the runtime in GsageFacade::initialize function.

SystemFactory reads systems array in the configuration file. For example:

```
...
"systems": ["ogre", "lua", "dynamicStats"]
...
```

- lua and dynamicStats are preinstalled systems.

- `ogre` and `recast` are registered by the `OgrePlugin`.

Each system has two identifiers:

- **implementation** id.
- **functional** id.

Implementation id is used by `SystemFactory` to create a system. **Functional** id defines system purpose and is used to identify it's components.

For example, there is `render` system that is using `ogre` underneath.

When the system is added to the engine it can read the configuration from the global configuration file. System configuration must be stored on the root level of global configuration file or scene file under **functional** id.

For example:

```
{
...
  "movement": {
    "cacheFolder": "./"
  }
  "coolSystem": {
    "setMeUP": 1
  }
...
}
```

Engine will inject each system configuration placed under system **functional** id. The system will get a `Gsage::DataProxy` object and will get all system specific parameters from it.

See [Custom Systems](#) for more information how to add new types of systems into Gsage engine.

1.3.4 Input

Input is configured by `inputHandler` field. It should have string identifier of input factory, which is installed into the Gsage Facade.

Currently it supports two kinds of inputHandlers:

- `SDL` (preferred).
- `ois` (may be removed in future releases).

You can implement your own input handler and install it into the Gsage Facade. See [Custom Input Handler](#) to get more info how to implement your own input handler.

1.3.5 Window Management

`windowManager` section can be used to configure window management system. It has one mandatory field and one optional:

"`type`" is mandatory and defines window manager type to use. "`windows`" is optional and may contain the list of windows that should be created by the window manager.

Elements of this list should be objects and may vary depending on the implementation fo the window manager.

1.3.6 Log Config

`logConfig` can be used to define path to log configuration file. Refer to [easylogging++ documentation](#) for more details.

1.3.7 Packager

This packager can install any lua dependencies using `luarocks`. `deps` array should contain the list of dependencies. Each entry of this array support version pinning and version query operators.

1.3.8 Plug-Ins

Global config file can contain any additional configuration, which are relevant to installed plugins.

1.4 Saving and Loading Game State

1.4.1 Level File Format

Static scene objects are defined by `render` component of an entity. Only difference for static object is the **static** flag, defined in the entity. Static scene object can have any other components defined as well, such as script or sound.

Level is described by simple json or msgpack:

```
{
  "settings": {
    "render": {
      "resources": {
        "levelID": [
          "Zip:models/packs/levelResourcePack.zip"
        ]
      },
      "colourBackground": "0xc0c0c",
      "colourAmbient": "0x7F7F7F",
      "colourDiffuse": "0xc0c0c"
    },
    "script": {
      "hooks": {
        "camera": "setOrbitalCam()",
        "damageEmitter": "emitter.create('damage') "
      }
    }
  },
  "entities": [{
    "id": "castle",
    "render": {
      "root": {
        "position": "0,0,0",
        "rotation": "1,0,-1,0",
        "scale": "1,1,1",
        "children": [{
          "type": "model",
          "mesh": "castle.mesh",
          "name": "castle",
```

(continues on next page)

(continued from previous page)

```

        "castShadows": true
    }
}
}
]
}

```

"settings" section is used to reconfigure all engine systems for the particular level. It works in the similar way as the global config. In this example, "render" and script systems are configured, and have the following settings:

- "resources" contains list of resources, that are required for the level. Each resource list is identified by unique id to make shared resources reusable between different levels.
- "script" has some level startup script hooks defined.

"entities" section describes entity list to be used as static scene elements.

1.4.2 Loading Levels

All levels are stored in the same folder. Level can be loaded by calling `Gsage::GameDataManager::loadArea()`. This will initialize all static entities and reconfigure systems.

By default, level folder path is `resources/levels`. Game data manager is configured in the `gameConfig.json` file, see [Gsage::GameDataManager Config](#) for more information.

Levels information should be considered as constant. Game data loader is designed with thoughts that levels information will be stored in the single archive, which won't be changed.

To provide ability to modify levels information and save dynamic entities states, save files are used.

Note: Currently save file copies the whole level information. There is plan to save only difference between initial data and save

To load save file `Gsage::GameDataManager::loadSave()` function can be used. Save file format differs from level file format a bit:

- save file saves level information in the field `area`.

Several new Fields:

- `characters` saves all characters information.
- `placement` saves positions of all characters on all locations.
- `settings` saves global configs for systems (can be used for render system settings customizations).

Note: Save file format might change in the future. `settings` is likely to be removed.

1.4.3 Saving Levels

Each engine component and system supports serialization. All settings, that can be read from configs also can be dumped to config files.

This allows `Gsage::GameDataManager` to iterate all systems and entities and dump their state to file.

Save file can be created by calling `Gsage::GameDataManager::dumpSave()`.

Note: There is no method to modify level file itself yet. It will be created for editor purposes later.

1.4.4 Lua Bindings

- `game:loadSave("save1234")`
- `game:dumpSave("save1234")`

1.5 Custom Start Script

Note: Start scripts are still in a mess. Common code parts will be bundled in some module later.

It is possible to write any kind of code in the start script. But it should be set up properly.

Firstly, it is required add other lua scripts folders to `package.path`.

```
package.path = package.path .. ';' .. getResourcePath('scripts/?lua') ..      --  
↳ main scripts folder, contains generic core logic                          ;' .. getResourcePath('behaviors/trees/?lua') .. --  
↳ behavior trees folder                                                    ;' .. getResourcePath('behaviors/?lua') .. ";"
```

1.5.1 Setting Up ImGui

Check if ImGui UI manager is installed and register a view:

```
local ImGuiInterface = require 'ImGui.base'  
  
if ImGuiInterface:available() then  
    -- simple function  
    ImGuiInterface:addView("window", function()  
        ImGui.TextWrapped("Hello world")  
    end, true)  
end
```

Refer to *Creating ImGui Views in Lua* for more details about registering ImGui views in Lua.

1.5.2 Setting Up LibRocket

Librocket support can be checked by verifying if there is `onRocketContext` event handler. Then it is possible to subscribe to that callback:

```
-- librocket initialization  
if event.onRocketContext ~= nil then  
    event:onRocketContext(core, RocketContextEvent.CREATE, initLibrocket)  
end
```

Librocket views, fonts loading should be done in `initLibrocket`:

```
function initLibrocket(event)
  local ctx = rocket.contexts[event.name]
  if not rocketInitialized then
    local fonts =
    {
      "Delicious-Roman.otf",
      "Delicious-BoldItalic.otf",
      "Delicious-Bold.otf",
      "Delicious-Italic.otf",
      "lucida.ttf"
    }
    for _, font in pairs(fonts) do
      resource.loadFont(font)
    end
  end
end

main = resource.loadDocument(ctx, "minimal.rml")
cursor = resource.loadCursor(ctx, "cursor.rml")

main:Show()
end
```

`initLibrocket` will be called for each context initialized.

`event.name` can be used to distinguish different contexts and render different set of views for each of them.

1.5.3 Subscribing For Key Events

```
function handleKeyEvent(e)
  if e.type == KeyboardEvent.KEY_UP then
    -- handle any key up
  end

  if e.key == Keys.KC_T and e.type == KeyboardEvent.KEY_DOWN then
    -- handle pressing T key
  end
end

event:onKeyboard(core, KeyboardEvent.KEY_DOWN, handleKeyEvent)
event:onKeyboard(core, KeyboardEvent.KEY_UP, handleKeyEvent)
```

1.5.4 Handling Scene Object Selection

```
local function onSelect(e)
  local target = eal.getEntity(e.entity)
  if not target then
    return
  end

  ogreView:setGizmoTarget(target)
```

(continues on next page)

(continued from previous page)

```
end

-- generic approach
event:bind(core, SelectEvent.OBJECT_SELECTED, onSelect)
```

1.5.5 Pipeline Setup

TBD: when ogre 2.1 support is added

1.6 Entity

1.6.1 Entity Format

Each entity consists of several components.

Each component can communicate with adjacent component through entity. Each component belongs to one system and stores state of the entity for that system.

For example, for render system:

```
"render": {
  "root": {
    "position": "0,0,0",
    "rotation": "1,0,-1,0",
    "scale": "1,1,1",
    "children": [{
      "type": "model",
      "mesh": "castle.mesh",
      "name": "castle",
      "castShadows": true
    }]
  }
}
```

The component stores information about nodes: "root" is always root node of the component. It has "position", "rotation" and other props, typical for render system. "children" here can store the list of various visual children:

- models.
- billboards.
- particle systems.
- and others.

There are different kinds of systems. Script component data can look like this:

```
"script":
{
  "setupScript": "@File:characters/scripts/ninja.lua",
  "behavior": "dumbMonster"
}
```

The engine is highly extensible, so it's easy to add any new kind of system. Each system is installed in the engine with the unique string id. The same id is used to match the system where the entity should be created.

That way, engine will find system by the name "script" and will create new component there using parameters, which are defined for it in the entity description.

For example for the **script** component type it will call `Gsage::Engine::createComponent()` with `type = "script"` and dict:

```
{
  "setupScript": "@File:characters/scripts/ninja.lua",
  "behavior": "dumbMonster"
}
```

Then it will add the pointer to the created component to the entity object.

1.6.2 Lifecycle

On each update, each component state can be updated. Engine iterates and updates all the systems and each system iterates and updates each component it owns.

Each component can be altered during engine operation. For example, render component can update it's position or change model. Movement can change speed.

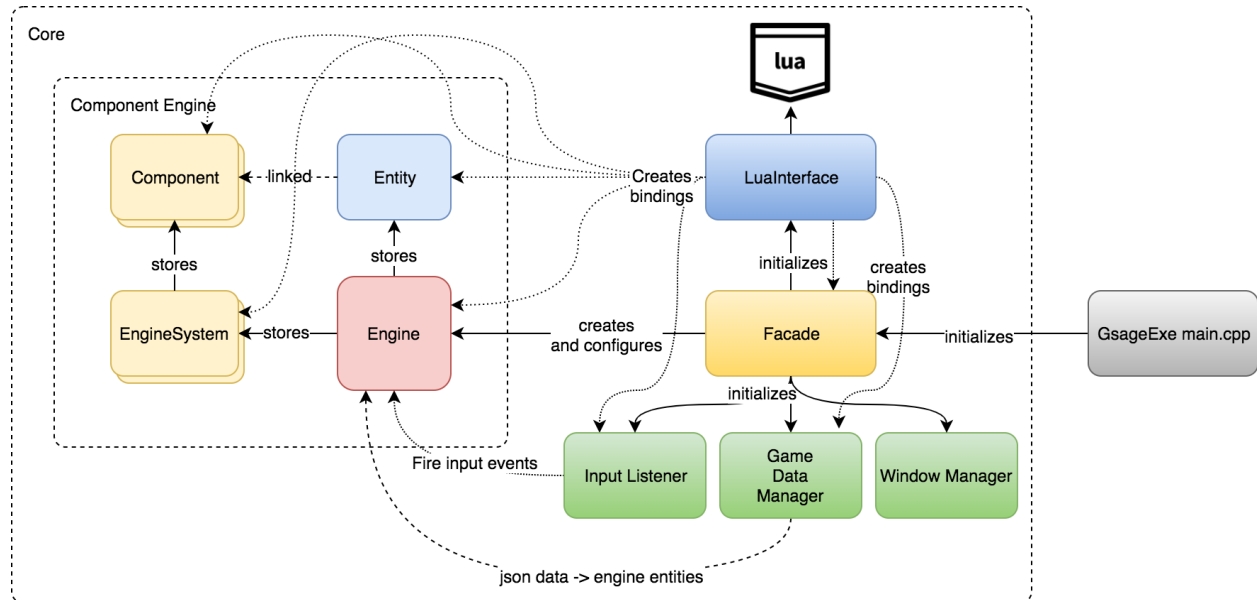
1.6.3 Entity Add Flow

1. `Gsage::Engine` creates `Gsage::Entity` in the pool of entities. All entities are stored in the `Gsage::Engine`.
2. Engine iterates through the list of keys of `Gsage::Entity`, finds appropriate system by string `id` and allocates a component there.
3. Each system allocates the component in the pool and configures created component with `Gsage::DataProxy` that came from `Gsage::Engine`.

1.6.4 Entity Remove Flow

1. `Gsage::Engine` finds `Gsage::Entity` by `id` and iterates through all it's components.
2. `Gsage::Engine` removes each component by pointer it got from `Gsage::Entity`.
3. `Gsage::ComponentStorage` deallocates component.
4. `Gsage::Engine` removes entity from pool when all components are deleted.

1.7 Internals



1.7.1 Terminology

- **component** — building block for the `Gsage::Entity`. Constructing `Entity` from different components will make it have different features.
- **system** — part of the `Gsage::Engine`. Implements some gaming logic, manages corresponding **component** pool.
- **manager** — logic, that goes out of **system** boundary, is wrapped into a **manager**. For example: UI, Input, Windowing.

1.8 Custom Systems

1.8.1 Writing a Component

You should create a new class, which is derived from `Gsage::EntityComponent`, then define static `SYSTEM` field in the created component class. This field will define the id which will be used for this component and the system.

The component is derived from the `Gsage::Serializable`, so it can be easily configured to read and write it's state into `json/mspack`. More information about serialization [Serialize Classes](#).

If everything is defined properly, `Gsage::GameDataManager` will handle state saving and restoring for the newly created component.

1.8.2 Engine System

Each Gsage Engine system should implement at least `Gsage::EngineSystem` interface.

To make system work, you should implement several methods.

Abstract Methods

- `Gsage::EngineSystem::update()` - called on each engine update.
- `Gsage::EngineSystem::createComponent()` - create component in the system.
- `Gsage::EngineSystem::removeComponent()` - remove component from the system.
- `Gsage::EngineSystem::unloadComponents()` - unload all components from the system.

This interface should be convenient to use for all systems that have components. It's not if you want system without components.

For that, you can use `Gsage::UpdateListener` interface.

If you want to add which does not need update but has components, then you'll have to make stub implementation for the `Gsage::EngineSystem::update()` method.

It would be nice to have a separate interface & pool for such systems so there is [a task](#) for that.

Also, each system must have static field `ID`, which defines it's string identifier for the `Gsage::SystemManager`.

Optional Methods

- `Gsage::EngineSystem::initialize()` - handle initial configs in this method.
- `Gsage::EngineSystem::configure()` - can be called if any level has different configs.

Don't forget to call base class implementation in each override, otherwise `Gsage::EngineSystem::mConfig` will be unset.

Fields

- `Gsage::EngineSystem::mEngine` - engine instance.
- `Gsage::EngineSystem::mConfig` - dictionary the current system configs.

1.8.3 Component Storage

There is also another class, which can be used as a base class for the system: `ComponentStorage`.

This class helps you to handle component allocation, iteration, initialization.

It has only one pure virtual method `Gsage::ComponentStorage::updateComponent()`. This method is called for each component in the system.

Optional Methods

- `Gsage::ComponentStorage::prepareComponent()` - call it for some precondition logic handling.
- `Gsage::ComponentStorage::fillComponentData()` - this method can be used to configure the component.

1.8.4 Registering a New System

Newly created system can be registered in the facade by a simple call. Just call `Gsage::GsageFacade::addSystem()` with the new system. You can do it at any time and engine will initialize this system properly.

Example:

```
facade.addSystem<Gsage::LuaScriptSystem>();
```

There is also another way to register new type of the system by using `Gsage::GsageFacade::registerSystemFactory`.

```
facade.registerSystemFactory<Gsage::LuaScriptSystem>("luaSystem");
```

After registering system this way, it will be possible to tell engine to create it using game config `systems` field:

```
...
"systems": ["luaSystem"]
...
```

Important: This solution is more flexible as it allows engine to create such systems at the runtime.

1.8.5 Further steps

- After you've created the new system, you may want to expose some methods to the lua. See [Lua Bindings](#), [Bind Engine Systems](#) and [Bind Entity Components](#) for more details.
- You may also want to wrap this new system into a plugin. See [Plugin System](#) for more details.

1.9 Lua Bindings

`Gsage::GsageFacade::initialize()` initializes Lua state in the engine. This is a single global Lua state for the whole engine (At least for now it's single).

`Gsage::LuaInterface` binds the Core bindings:

- bindings for entity and engine wrappers.
- `Gsage::GsageFacade` bindings like `Gsage::GsageFacade::shutdown()`.
- `Gsage::GameDataManager` bindings.
- and others.

`Sol2` library is used for Lua bindings.

Besides bindings, `LuaInterface` initializes global variables:

- `core` - `Gsage::Engine` instance.
- `game` - `Gsage::GsageFacade` instance.
- `data` - `Gsage::GameDataManager` instance.
- `log` - provides `easylogging++` API to lua.

Any script can be executed by calling `Gsage::LuaInterface::runScript()`. `startupScript` configuration variable can be used to define script which will be executed after `Gsage::GsageFacade` initialization.

`LuaInterface` also runs packager script.

1.9.1 Bindings Guidelines

All Core bindings are located in the `LuaInterface` file. If any plug-in needs to add some Lua bindings, it should override `Gsage::IPlugin::setupLuaBindings` method. Plug-in can access Lua state by using `mLuaInterface->getSolState()` function. It will return pointer to `sol::state_view`.

Note: Note that the pointer to Sol2 state can be 0 if the `LuaInterface` was not initialized by calling `Gsage::LuaInterface::initialize()`.

Bind Engine Systems

If you create a new engine system, you may want to access it from Lua. Systems can be added dynamically at any point and `Gsage::Engine` functions should be updated in the runtime:

```
lua["Engine"]["script"] = &Engine::getSystem<LuaScriptSystem>;
```

This way, when a plugin registers a new system, it will also update `Gsage::Engine` to have getter for this system: `core:script()`.

So, if you add a new system, you will need to create a new binding like this:

```
lua.new_usertype<KittySystem>("KittySystem"
    "getKitten", &KittySystem::getKitten
);

lua["Engine"]["kitty"] = &Engine::getSystem<KittySystem>;
```

After you make this binding, you will be able to get the system instance:

```
s = core:kitty()
s:getKitten()
```

Bind Entity Components

When registering a new Component, you should also update `Gsage::Entity` functions and add the getter for the new Component in the same way as for the new System, but instead of `Engine` binding, you should use `Entity`:

```
lua.new_usertype<KittyComponent>("KittyComponent"
    "meow", &KittyComponent::meow
);

lua["Entity"]["kitty"] = &Entity::getComponent<KittyComponent>;
```

After that it will be possible to get Component from Entity instance by using newly registered getter:

```
e = eal:getEntity("cat")
cat.kitty:meow()
```

Bind Events

Events can be handled in Lua script in two ways:

- `event:bind(...)` will bind generic callback. You can use it if you do not need upcasting from `Gsage::Event` to derived event type.
- `event:<handlerID>(...)` will bind callback specifically for some concrete type of event.

If you use bind, you will not be able to access derived class methods or variables:

```
local onSelect = function(event)
    print(e.hasFlags) -- prints nil
end

event:bind(core, "objectSelected", onSelect)
```

To listen for any specific event type use `event:<handlerID>`.

`handlerID` is defined when binding a new event type:

```
registerEvent<SelectEvent>("SelectEvent",
    "onSelect", // <-- handlerID
    sol::base_classes, sol::bases<Event>(),
    "hasFlags", &SelectEvent::hasFlags,
    "entity", sol::property(&SelectEvent::getEntityId),
);
```

To handle `Gsage::SelectEvent` in Lua:

```
local onSelect = function(event)
    -- you will be able to access derived class methods
    print(e.hasFlags(OgreSceneNode.DYNAMIC))
end

event:onSelect(core, "objectSelected", onSelect)
```

1.10 Plugin System

Gsage Engine supports plugins, which can be installed at the runtime. Each plugin can be wrapped into dynamically linked library or shared library (.so/.dynlib/.dll)

Plugin loader is borrowed from the Ogre project `DynLib.h`.

All plugins installed into the system should have unique id.

- `Gsage::GsageFacade::loadPlugin()` can be used to install plugin from a shared library.
- `Gsage::GsageFacade::installPlugin()` can be called with instantiated plugin object.
- `Gsage::GsageFacade::uninstallPlugin()` removes installed plugin.
- `Gsage::GsageFacade::unloadPlugin()` unloads shared library.

1.10.1 Writing a Plugin

Each plugin should implement `IPlugin` interface.

Abstract Methods

- `Gsage::IPlugin::getName()` - this method should return unique string id of the plugin.
- `Gsage::IPlugin::installImpl()` - should contain plugin initialization logic.
- `Gsage::IPlugin::uninstallImpl()` - should contain plugin destruction logic.

`installImpl` can register UI/Window manager factories, add new SystemFactories or add a system to the Engine. `uninstallImpl` should properly uninstall plugin from the facade: remove the system, unregister lua bindings or unregister UI/Window manager.

Important: Though it is possible to define bindings in the `installImpl` method, there is another method for this: `Gsage::IPlugin::setupLuaBindings()`. Bindings may work properly if defined in `installImpl`, but if Lua state will be recreated by the `LuaInterface`, bindings will be lost.

While implementing a plugin, you should also add couple extern "C" methods, which will be called from the `GsageFacade` after plugin dynamic library was loaded:

- `dllStartPlugin.`
- `dllStopPlugin.`

Example:

```
// from OgrePlugin.h

...

#if GSAGE_PLATFORM == GSAGE_WIN32
#ifdef PLUGIN_EXPORT
#define PluginExport __declspec (dllexport)
#else
#define PluginExport __declspec (dllimport)
#endif
#else
#define PluginExport
#endif

...
```

```
// from OgrePlugin.cpp

...

bool OgrePlugin::installImpl()
{
    mEngine->addSystem<OgreRenderSystem>();
    mEngine->addSystem<RecastMovementSystem>();
    return true;
}

void OgrePlugin::uninstallImpl()
{
    mEngine->removeSystem("render");
    mEngine->removeSystem("movement");
}

OgrePlugin* ogrePlugin = NULL;
```

(continues on next page)

(continued from previous page)

```

extern "C" bool PluginExport dllStartPlugin(GsageFacade* facade)
{
    if(ogrePlugin != NULL)
    {
        return false;
    }
    ogrePlugin = new OgrePlugin();
    return facade->installPlugin(ogrePlugin);
}

extern "C" bool PluginExport dllStopPlugin(GsageFacade* facade)
{
    if(ogrePlugin == NULL)
        return true;

    bool res = facade->uninstallPlugin(ogrePlugin);
    if(!res)
        return false;
    delete ogrePlugin;
    ogrePlugin = NULL;
    return true;
}

...

```

Set Up Lua Bindings

- `Gsage::IPlugin::setupLuaBindings()` - should contain all Lua bindings.

This method will be called again if `lua_State` was recreated.

Example:

```

...

void OgrePlugin::setupLuaBindings() {
    if (mLuaInterface && mLuaInterface->getState())
    {
        sol::state_view lua = *mLuaInterface->getSolState();

        // Ogre Wrappers

        lua.new_usertype<OgreObject>("OgreObject",
            "type", sol::property(&OgreObject::getType)
        );

        ...

        lua.new_usertype<Ogre::Quaternion>("Quaternion",
            sol::constructors<sol::types<const Ogre::Real&, const Ogre::Real&, const
↪Ogre::Real&, const Ogre::Real&>, sol::types<const Ogre::Radian&, const
↪Ogre::Vector3&>>(),
            "w", &Ogre::Quaternion::w,
            "x", &Ogre::Quaternion::x,
            "y", &Ogre::Quaternion::y,

```

(continues on next page)

(continued from previous page)

```

        "z", &Ogre::Quaternion::z,
        sol::meta_function::multiplication,
↪ (Ogre::Quaternion(Ogre::Quaternion::*) (const Ogre::Quaternion&) const) &
↪Ogre::Quaternion::operator*
    );

    lua.new_usertype<Ogre::Radian>("Radian",
        sol::constructors<sol::types<float>>()
    );

    lua.new_usertype<OgreSelectEvent>("OgreSelectEvent",
        sol::base_classes, sol::bases<Event, SelectEvent>(),
        "intersection", sol::property(&OgreSelectEvent::getIntersection),
        "cast", cast<const Event&, const OgreSelectEvent&>
    );

    lua["Engine"]["render"] = &Engine::getSystem<OgreRenderSystem>;
    lua["Engine"]["movement"] = &Engine::getSystem<RecastMovementSystem>;

    lua["Entity"]["render"] = &Entity::getComponent<RenderComponent>;
    lua["Entity"]["movement"] = &Entity::getComponent<MovementComponent>;

    ...
}
}
...

```

1.11 Custom Input Handler

New handler should be registered in the `Gsage::GsageFacade` by calling `Gsage::GsageFacade::registerInputFactory()`. As you can get from the method specialization, you should pass class which implements interface `Gsage::AbstractInputFactory`.

Any concrete factory should create input handlers that implement interface `Gsage::InputHandler`.

Input factory will requested to create new handler for each created window. After creation, this handler will receive all resize and close events.

`Gsage::OisInputFactory` can be taken as a good starting point for a new input implementation.

```

InputHandler* OisInputFactory::create(size_t windowHandle, Engine* engine)
{
    return new OisInputListener(windowHandle, engine);
}

```

1.12 Serialize Classes

1.12.1 Bindings

`Gsage::Serializable` is a convenient base class which can be used to tell Gsage Engine how the cpp object, which does not have any reflection and dynamic properties/methods lookup, can be converted into a

`Gsage::DataProxy`.

`Gsage::DataProxy` can be converted to `json`, `msgpack` and `sol::table`. Lua can pass `sol::table` directly into the engine and it will be wrapped by the `Gsage::DataProxy`. `json` and `msgpack` serialization will automatically try to convert all complex types to primitive type. When converting to `sol::table`, no conversion will take place, so if it is required to use field that stores a complex type in lua, this type should be registered in lua bindings. When converting `sol::table` to `Gsage::DataProxy`, it will try convert primitive types to complex. However, it is better to initialize the fields as complex types in Lua, as it will work faster.

When you derive class from the `Gsage::Serializable`, you should tell it what kind of properties you want to dump and read and how.

There are several macros for that:

Warning: doxygendefine: Cannot find define “BIND_PROPERTY” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_PROPERTY_WITH_PRIORITY” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_PROPERTY_OPTIONAL” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_ACCESSOR” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_ACCESSOR_WITH_PRIORITY” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_ACCESSOR_OPTIONAL” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_GETTER” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_GETTER_OPTIONAL” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_SETTER_OPTIONAL” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_READONLY_PROPERTY” in doxygen xml output for project “gsage” from directory: xml/

Warning: doxygendefine: Cannot find define “BIND_WRITEONLY_PROPERTY” in doxygen xml output for project “gsage” from directory: xml/

There are various places in code, where you can check out how these methods are used. For example, from `Gsage::RenderComponent`:

```

BIND_ACCESSOR_OPTIONAL("resources", &RenderComponent::setResources, &
↪RenderComponent::getResources);
BIND_GETTER("root", &RenderComponent::getRootNode);
BIND_GETTER("animations", &RenderComponent::getAnimations);

```

Or `Gsage::SceneNodeWrapper`:

```

BIND_PROPERTY("offset", &mOffset);

BIND_ACCESSOR("orientationVector", &SceneNodeWrapper::setOrientationVector, &
↪SceneNodeWrapper::getOrientationVector);
BIND_ACCESSOR("name", &SceneNodeWrapper::createNode, &SceneNodeWrapper::getId);
BIND_ACCESSOR("position", &SceneNodeWrapper::setPosition, &
↪SceneNodeWrapper::getPosition);
BIND_ACCESSOR("scale", &SceneNodeWrapper::setScale, &SceneNodeWrapper::getScale);
BIND_ACCESSOR("rotation", &SceneNodeWrapper::setOrientation, &
↪SceneNodeWrapper::getOrientation);
BIND_ACCESSOR("children", &SceneNodeWrapper::readChildren, &
↪SceneNodeWrapper::writeChildren);

```

1.12.2 Read and Dump

After the bindings are defined, it will be possible to use `Gsage::Serializable::dump()` and `Gsage::Serializable::read()` functions.

- `Gsage::Serializable::read()` - will allow to convert `Gsage::DataProxy` to the class. It will return false if any of non-Optional field is missing from the dict.
- `Gsage::Serializable::dump()` - will allow to convert the class `Gsage::DataProxy`.

1.13 Engine Abstraction Layer

As systems, that are installed from plugins most definitely will have different set of Lua bindings, Gsage Engine has Lua abstraction layer, that can be used to unify interfaces for different kinds of systems.

EAL helps to tie Lua code to the engine entities. Besides that EAL allows extensive code reuse between different objects.

The idea is pretty simple, each engine entity can have defined:

- A single `class` type.
- Several `mixin s`.
- Each component can also trigger EAL to add more Lua logic.

1.13.1 Defining an Extension

Extension is a function that accepts a class prototype as a first parameter.

```
local function extension(cls)

end
```

Then it is possible any amount of additional methods in that function.

```
local function extension(cls)
    function cls:moveToOrigin()
        self.render:setPosition(0, 0, 0)
    end
end
```

Besides defining custom methods, it is also possible to define `setup` and `teardown` methods. They are pretty similar to constructor/destructor methods, but there can be more than one `setup` and `teardown`.

```
local function extension(cls)
    ...
    cls.onCreate(function(self)
        self.someVariable = 1
    end)

    cls.onDestroy(function(self)
        print(tostring(self.id) .. " was destroyed")
    end)
    ...
end
```

To add this extension to the system, it is required to call `extend` method.

```
eal:extend({mixin = "extension"}, extension)
```

Now everything combined will look like that:

```
local eal = require 'eal.manager'

local function extension(cls)
    cls.onCreate(function(self)
        self.someVariable = 1
    end)

    cls.onDestroy(function(self)
        print(tostring(self.id) .. " was destroyed")
    end)

    function cls:moveToOrigin()

```

(continues on next page)

(continued from previous page)

```

        self.render:setPosition(0, 0, 0)
    end
end

eal:extend({mixin = "extension"}, extension)

```

In this example, extension is created as a mixin. Then, to use this extension for an entity, it is required to modify it's JSON data.

```

{
  "id": "something",
  "props": {
    "mixins": ["extension"] // adding a mixin
  }
  "render": {
    "root": {
    }
  }
}

```

After all these manipulations you should be able to use this EAL interface:

```

local e = eal:getEntity("something")
-- our extension method
e:moveToOrigin()
-- the variable set in set up should be accessible
assert(e.someVariable == 1)

```

1.13.2 Supported Types of Extensions

System

System wide extensions. Allows applying extension for all entities that have a component of a system `system` with subtype `type`.

Extending EAL:

```
eal:extend({system="render", type="ogre"}, extension)
```

There is no need to modify entity data as this extension will be applied system wide: each entity with component of system `render` with subtype `ogre` will have this extension applied.

Class

When there is no need to stick to any particular system type, but it's still required to distinguish different system subtype, it is better to use the class extension. Though it is also possible to define a class without a strict requirement of system type.

```

-- enable this extension only when render system type is "ogre"
eal:extend({class = {name = "camera", requires = {render = "ogre"}}}, extension)

```

Using it in the entity data:

```
{
  ...
  "props": {
    "class": "camera"
  }
  ...
}
```

Mixin

Mixin allows defining multiple different extensions for a single entity that are not tied to any specific system. It is better to define only the highest level logic in the mixin. Do not create too many mixins as it may hit the performance.

Important: As it is possible to make a composition of extensions of different kinds, it is necessary to know the order they are applied. First go system level extensions. Then class extension. Then mixins in order, defined in the json array.

1.14 Custom Window Manager

Gsage Engine allows you to implement new windowing managers.

Current main windowing system is SDL.

WindowManager is optional, it is possible to let render system create windows on it's own.

Steps to register new WindowManager:

1. Inherit `Gsage::WindowManager` interface. Implement abstract methods:

- `Gsage::WindowManager::initialize()`.
- `Gsage::WindowManager::createWindow()`.
- `Gsage::WindowManager::destroyWindow()`.

2. Inherit `Gsage::Window` interface. Implement abstract methods:

- `Gsage::WindowManager::getWindowHandle()` should return native OS window handle.
- `Gsage::WindowManager::getGLContext()` should return open GL pointer. Return 0 if not needed.

3. Add register, unregister methods in the plugin:

```
...
bool MyPlugin::installImpl() {
    mFacade->registerWindowManager<MyWindowManager>("MyWindowManager");
    return true;
}
...
```

(continues on next page)

(continued from previous page)

```
void MyPlugin::uninstallImpl() {
    mFacade->removeWindowManager("MyWindowManager");
}
```

4. Make Gsage load the Plugin and switch to the new WindowManager:

```
...
"windowManager": {
    "type": "MyWindowManager"
}

...
"plugins":
[
    ...
    "MyPlugin"
    ...
]
```

1.15 Using Another Render System

Writing the plugin which introduces another `RenderSystem` is pretty complicated thing to do, because many other plugins usually depend on `RenderSystem` implementation.

The most annoying thing is to write all `UIManager` adapters for newly implemented `RenderSystem`. Besides that, it is required to wrap all scene objects into serializable classes.

TODO: describe what interfaces to implement TODO: describe how to use generic 3D primitives

1.15.1 Implementing 2D Render System

TODO: never done that yet, so hard to describe it yet

1.15.2 Using EAL

EAL can help to simplify migration from one render system to another. Or make Lua code written for one `RenderSystem` work with another `RenderSystem` without any changes.

TODO: more details

1.15.3 Implement Factories

If it is impossible to use the same data structure for render component of the new system, it is possible to achieve compatibility by using different set of factories for different systems.

This way it will be possible to make it use the same interface, having different implementation underneath.

1.16 Ogre Plugin

Plugin Name: `OgrePlugin`

Only one version of OGRE can be built simultaneously. E.g. you can't build both 2.1 and 1.9 plugins at the same time.

1.16.1 Common Features

1.16.2 1.9.0

This version is still used by default, no additional build parameters are required to select this version.

Supported rendering subsystems:

- Direct3D9 Rendering Subsystem
- OpenGL Rendering Subsystem

Custom pipeline setup is not there yet. Editor starts, but it crashes when trying to load the scene:

- Direct3D11 Rendering Subsystem

1.16.3 1.10.0

Will be used instead of 1.9.0 in future. 1.9.0 support will be dropped.

1.16.4 2.1.0

To build against OGRE 2.1.

Using makefile:

```
# bash
OGRE_VERSION=2.1.0 make build

# windows shell
set OGRE_VERSION=2.1.0
make.exe build
```

Conan commands:

```
conan install -g cmake -s build_type=Release -o gsage:with_ogre=2.1.0 --
↪build=outdated .
conan build .
```

Supported rendering subsystems:

- OpenGL 3+ Rendering Subsystem
- Metal Rendering Subsystem (OSX only)

1.17 Dear ImGui Plugin

Plugin Name ImGuiPlugin

ImGui is used only for game development utilities. It may be possible to use it for the game interface but it may lack some functional.

Important: built-in `ImGui::Image(...)` is not supported. There is no support for images yet, but it should be possible to implement images basing on `Gsage::OgreView`.

1.17.1 Creating ImGui Views in Lua

All ImGui views manipulation must be done using `ImGuiInterface`. Registering a simple view is pretty straightforward:

```
local ImGuiInterface = require 'ImGui.base'

if ImGuiInterface.available() then

    -- simple function
    ImGuiInterface.addView("window", function()
        ImGui.TextWrapped("Hello world")
    end, true)
end
```

It is possible to use Lua class for some complicated views:

```
local ImGuiInterface = require 'ImGui.base'

-- ImGui engine stats view
Stats = class(ImGuiWindow, function(self, title, docked, open)
    ImGuiWindow.init(self, title, docked, open)
    self.fps = 0
    self.frames = 0
    self.elapsedTime = 0
    self.monitor = ResourceMonitor.new(0.1)
    self.stats = self.monitor.stats

    self.handleTime = function(delta)
        self.frames = self.frames + 1
        self.elapsedTime = self.elapsedTime + delta
        if self.elapsedTime >= 1 then
            self.fps = self.frames
            self.frames = 0
            self.elapsedTime = 0
        end
    end
end)
game.addUpdateListener(self.monitor)
time.addHandler("stats", self.handleTime, true)
end)

-- render stats
function Stats:__call()
    if self:ImGuiBegin() then
        ImGui.Text("FPS:" .. self.fps)
        ImGui.Text("CPU:" .. math.floor(self.stats.lastCPU * 100))
        self:ImGuiEnd()
    end
end

if ImGuiInterface.available() then
```

(continues on next page)

(continued from previous page)

```
local stats = Stats("stats", true)
ImGuiInterface:addWidget("window", stats)
end
```

Note that this view inherits `ImGuiWindow`. This allows this view to be configured as dockable window by setting `docked` parameter to `true`.

ImGui Lua Interface

ImGui interface for Lua differs from the what there is for C++. ImGui relies on pointers to bool, float and other types, but there is no way to pass pointer to primitive types from Lua to C++.

Refer to `PlugIns/ImGui/include/ImGuiLuaInterface.h` file to see the list of all available methods.

There are some additional utility classes for Ogre:

class `OgreView` : public `EventSubscriber<OgreView>`
Allows displaying ogre camera into imgui window.

Lua usage example:

```
-- create ogre viewport
viewport = ImGui.createOgreView("#000000FF")

local textureID = "myOgreView"

-- render camera to texture
local cam = camera:create("free")
cam:renderToTexture(textureID, {
    autoUpdated = false
})

-- set ogre view texture
viewport:setTextureID(textureID)
-- update on each imgui render call
viewport:render(320, 240)
```

class `Gizmo` : public `EventSubscriber<Gizmo>`
Transformation *Gizmo*, supports move, scale and rotate operations.

Lua usage example:

```
-- create gizmo
gizmo = ImGui.createGizmo()

-- setting target
local render = eal:getEntity("test").render
if render == nil then
    exit(1)
end
gizmo:addTarget(render.root)

-- enable
gizmo:enable(true)

-- render on each ImGui cycle
gizmo:render(0, 0, 320, 240)
```

(continues on next page)

(continued from previous page)

```

-- changing mode
gizmo.mode = ImGui.gizmo.WORLD

-- changing operation
gizmo.operation = ImGui.gizmo.ROTATE

-- draw coordinates editor for this gizmo
-- it is separate to make it possible to draw it in the separate window
gizmo:drawCoordinatesEditor(1, 0, 0, "%.3f", 1,
"position",
"scale",
"rotation")

```

class ImGuiDockspaceRenderer

Dockspace for ImGui.

Provides alternate methods for Begin and End. Docked view should be surrounded by BeginDock and EndDock methods.

```

local flags = 0
local active, open = ImGui.BeginDockOpen("test", true, flags)
if active and open then
    -- render some view here
    ImGui.TextWrapped("Hello World!")
end
ImGui.EndDock()

```

Saving and loading dockspace state:

```

-- save dock state (will get lua table)
local savedState = ImGui.GetDockState()

...

-- restore dock state
ImGui.SetDockState(savedState)

```

Additional ImGui global variables:

- `ImGui.render` *Gsage::ImGuiRenderer* instance.
- `ImGui.dockspace` *Gsage::ImGuiDockspaceRenderer* instance.

1.18 LibRocket Plugin

Plugin Name: RocketUIPlugin

TODO: examples, more description

1.19 SDL Plugin

Plugin Name: SDLPlugin

SDL Plugin registers *Gsage::SDLWindowManager* and *Gsage::SDLInputListener*.

1.20 Recast Navigation Plugin

Plugin Name: `RecastNavigationPlugin`

All render systems should inherit `Gsage::RenderSystem` and implement `Gsage::RenderSystem::getGeometry()` function.

When `Gsage::RecastNavigationPlugin` tries to build navmesh, it gets this raw 3D scene information from the `RenderSystem`.

TODO: describe some examples.

1.21 Class list

1.21.1 Class `Gsage::Animation`

class Animation

Class that wraps ogre animation state, adds speed definition

1.21.2 Class `Gsage::AnimationController`

class AnimationController

Animation controller

1.21.3 Class `Gsage::AnimationGroup`

class AnimationGroup

Container for several animations

1.21.4 Class `Gsage::AnimationScheduler`

class AnimationScheduler : public Serializable<AnimationScheduler>

Class that reads all animation information, configures states, then manages all animations playback

1.21.5 Class `Gsage::BillboardSetWrapper`

class BillboardSetWrapper : public Gsage::MovableObjectWrapper<OgreV1::BillboardSet>

1.21.6 Class `Gsage::BillboardWrapper`

class BillboardWrapper : public Serializable<BillboardWrapper>

1.21.7 Class `Gsage::CameraWrapper`

class CameraWrapper : public Gsage::MovableObjectWrapper<Ogre::Camera>, public Listener

1.21.8 Class Gsage::CustomPass

class CustomPass : public CompositorPass
 Subclassed by *Gsage::OverlayPass*

1.21.9 Class Gsage::CustomPassDef

class CustomPassDef : public CompositorPassDef
 Subclassed by *Gsage::OverlayPassDef*

1.21.10 Class Gsage::CustomPassProvider

class CustomPassProvider : public CompositorPassProvider

1.21.11 Class Gsage::Dock

class Dock

1.21.12 Class Gsage::EntityWrapper

class EntityWrapper : public *Gsage::MovableObjectWrapper*<OgreV1::Entity>

1.21.13 Class Gsage::Gizmo

class Gizmo : public EventSubscriber<*Gizmo*>
 Transformation *Gizmo*, supports move, scale and rotate operations.
 Lua usage example:

```
-- create gizmo
gizmo = imgui.createGizmo()

-- setting target
local render = eal.getEntity("test").render
if render == nil then
    exit(1)
end
gizmo:addTarget(render.root)

-- enable
gizmo:enable(true)

-- render on each ImGui cycle
gizmo:render(0, 0, 320, 240)

-- changing mode
gizmo.mode = imgui.gizmo.WORLD

-- changing operation
gizmo.operation = imgui.gizmo.ROTATE
```

(continues on next page)

(continued from previous page)

```
-- draw coordinates editor for this gizmo
-- it is separate to make it possible to draw it in the separate window
gizmo:drawCoordinatesEditor(1, 0, 0, "%.3f", 1,
"position",
"scale",
"rotation")
```

1.21.14 Class Gsage::GsageOgrePlugin

```
class GsageOgrePlugin : public IPlugin
```

1.21.15 Class Gsage::HlmsUnlit

```
class HlmsUnlit : public HlmsUnlit
    Extends Ogre standard HlmsUnlit
```

1.21.16 Class Gsage::HlmsUnlitDatablock

```
class HlmsUnlitDatablock : public HlmsUnlitDatablock
```

1.21.17 Class Gsage::IMovableObjectWrapper

```
class IMovableObjectWrapper : public Gsage::OgreObject
    Provides templateless base class for MovableObjectWrapper
```

Subclassed by *Gsage::MovableObjectWrapper< T >*, *Gsage::MovableObjectWrapper< Ogre::Camera >*, *Gsage::MovableObjectWrapper< Ogre::Item >*, *Gsage::MovableObjectWrapper< Ogre::Light >*, *Gsage::MovableObjectWrapper< Ogre::ManualObject >*, *Gsage::MovableObjectWrapper< OgreV1::BillboardSet >*, *Gsage::MovableObjectWrapper< OgreV1::Entity >*

1.21.18 Class Gsage::ImGUIRenderable

```
class ImGUIRenderable : public Renderable
```

1.21.19 Class Gsage::ImGuiDockspace

```
class ImGuiDockspace
    ImGui dockspace
```

1.21.20 Class Gsage::ImGuiDockspaceRenderer

```
class ImGuiDockspaceRenderer
    Dockspace for ImGui.
```

Provides alternate methods for *Begin* and *End*. Docked view should be surrounded by *BeginDock* and *EndDock* methods.

```

local flags = 0
local active, open = ImGui.BeginDockOpen("test", true, flags)
if active and open then
    -- render some view here
    ImGui.TextWrapped("Hello World!")
end
ImGui.EndDock()

```

Saving and loading dockspace state:

```

-- save dock state (will get lua table)
local savedState = ImGui.GetDockState()

...

-- restore dock state
ImGui.SetDockState(savedState)

```

1.21.21 Class `Gsage::ImageRenderer`

```

class ImageRenderer : public Gsage::Renderer
    Really simple image renderer

```

1.21.22 Class `Gsage::ImGuiDockspaceView`

```

class ImGuiDockspaceView : public Gsage::ImGuiViewCollection
    Renders window collection into workspace

```

1.21.23 Class `Gsage::ImGuiEvent`

```

class ImGuiEvent : public Event

```

1.21.24 Class `Gsage::ImGuiImage`

```

class ImGuiImage

```

1.21.25 Class `Gsage::ImGuiLuaInterface`

```

class ImGuiLuaInterface

```

1.21.26 Class `Gsage::ImGuiManager`

```

class ImGuiManager : public UIManager, public EventSubscriber<ImGuiManager>, public Gsage::ImGuiViewCollection

```

1.21.27 Class `Gsage::ImGuiMovableObject`

```

class ImGuiMovableObject : public MovableObject

```

1.21.28 Class Gsage::ImGuiMovableObjectFactory

class ImGuiMovableObjectFactory : public MovableObjectFactory
Factory object for creating *ImGuiMovableObject* instances

1.21.29 Class Gsage::ImGuiOgrePlugin

class ImGuiOgrePlugin : public IPlugin

1.21.30 Class Gsage::ImGuiOgreRenderer

class ImGuiOgreRenderer : public EventSubscriber<ImGuiOgreRenderer>, public Gsage::ImGuiRenderer
Subclassed by *Gsage::ImGuiRendererV1*, *Gsage::ImGuiRendererV2*

1.21.31 Class Gsage::ImGuiPlugin

class ImGuiPlugin : public IPlugin

1.21.32 Class Gsage::ImGuiRenderable

class ImGuiRenderable : public Renderable

1.21.33 Class Gsage::ImGuiRenderer

class ImGuiRenderer
Subclassed by *Gsage::ImGuiOgreRenderer*

1.21.34 Class Gsage::ImGuiRendererV1

class ImGuiRendererV1 : public Gsage::ImGuiOgreRenderer

1.21.35 Class Gsage::ImGuiRendererV2

class ImGuiRendererV2 : public Gsage::ImGuiOgreRenderer

1.21.36 Class Gsage::ImGuiTextBuffer

class ImGuiTextBuffer
ImGui buffer that can be used by lua

1.21.37 Class Gsage::ImGuiViewCollection

class ImGuiViewCollection
Subclassed by *Gsage::ImGuiDockspaceView*, *Gsage::ImGuiManager*

1.21.38 Class Gsage::ItemWrapper

```
class ItemWrapper : public Gsage::MovableObjectWrapper<Ogre::Item>
```

1.21.39 Class Gsage::LightWrapper

```
class LightWrapper : public Gsage::MovableObjectWrapper<Ogre::Light>
```

1.21.40 Class Gsage::ManualObjectWrapper

```
class ManualObjectWrapper : public Gsage::MovableObjectWrapper<Ogre::ManualObject>, public Gsage::MovableOb
```

1.21.41 Class Gsage::ManualTextureManager

```
class ManualTextureManager : public Listener
```

This class is used to handle manual texture management and update

1.21.42 Class Gsage::MaterialBuilder

```
class MaterialBuilder
```

1.21.43 Class Gsage::MaterialLoader

```
class MaterialLoader : public EventDispatcher, public EventSubscriber<MaterialLoader>
```

TODO: INDEXER

```
{ "filename": { "checksum": "...", "changed": "...", "materials": [] } }
```

```
for(auto& file : filesystem->ls(folder)) { if(filesystem->extension(file) != "material") { continue; } } Custom  
OGRE material loader
```

1.21.44 Class Gsage::MovableObjectWrapper

```
template<typename T>
```

```
class MovableObjectWrapper : public Gsage::IMovableObjectWrapper
```

1.21.45 Class Gsage::ObjectMutation

```
class ObjectMutation
```

1.21.46 Class Gsage::OgreGeom

```
class OgreGeom : public Geom
```

1.21.47 Class Gsage::OgreLogRedirect

class **OgreLogRedirect** : **public** LogListener
 Class, used to redirect all ogre output to the easylogging++

1.21.48 Class Gsage::OgreObject

class **OgreObject** : **public** Serializable<*OgreObject*>
 Abstract ogre object
 Subclassed by *Gsage::IMovableObjectWrapper*, *Gsage::ParticleSystemWrapper*, *Gsage::SceneNodeWrapper*

1.21.49 Class Gsage::OgreObjectManager

class **OgreObjectManager** : **public** EventDispatcher

1.21.50 Class Gsage::OgreObjectManagerEvent

class **OgreObjectManagerEvent** : **public** Event
 Event related to factory lifecycle

1.21.51 Class Gsage::OgreObjectManager::ConcreteOgreObjectPool

template<typename C>
class **ConcreteOgreObjectPool** : **public** *Gsage::OgreObjectManager::OgreObjectPool*

1.21.52 Class Gsage::OgreObjectManager::OgreObjectPool

class **OgreObjectPool**

1.21.53 Class Gsage::OgreRenderComponent

class **OgreRenderComponent** : **public** EventDispatcher, **public** RenderComponent
Ogre render system component

1.21.54 Class Gsage::OgreRenderSystem

class **OgreRenderSystem** : **public** ComponentStorage<*OgreRenderComponent*>, **public** RenderQueueListener, **public** E

1.21.55 Class Gsage::OgreSelectEvent

class **OgreSelectEvent** : **public** SelectEvent

1.21.56 Class Gsage::OgreTexture

class **OgreTexture** : **public** Texture, **public** Listener
 Implements abstract texture class Texture

1.21.57 Class `Gsage::OgreTexture::AllocateScalingPolicy`

```
class AllocateScalingPolicy : public Gsage::OgreTexture::ScalingPolicy
```

1.21.58 Class `Gsage::OgreTexture::DefaultScalingPolicy`

```
class DefaultScalingPolicy : public Gsage::OgreTexture::ScalingPolicy
```

1.21.59 Class `Gsage::OgreTexture::ScalingPolicy`

```
class ScalingPolicy
```

Subclassed by `Gsage::OgreTexture::AllocateScalingPolicy`, `Gsage::OgreTexture::DefaultScalingPolicy`

1.21.60 Class `Gsage::OgreView`

```
class OgreView : public EventSubscriber<OgreView>
```

Allows displaying ogre camera into imgui window.

Lua usage example:

```
-- create ogre viewport
viewport = imgui.createOgreView("#000000FF")

local textureID = "myOgreView"

-- render camera to texture
local cam = camera:create("free")
cam:renderToTexture(textureID, {
    autoUpdated = false
})

-- set ogre view texture
viewport:setTextureID(textureID)
-- update on each imgui render call
viewport:render(320, 240)
```

1.21.61 Class `Gsage::OverlayPass`

```
class OverlayPass : public Gsage::CustomPass
```

1.21.62 Class `Gsage::OverlayPassDef`

```
class OverlayPassDef : public Gsage::CustomPassDef
```

1.21.63 Class `Gsage::ParticleSystemWrapper`

```
class ParticleSystemWrapper : public Gsage::OgreObject
```

1.21.64 Class Gsage::RenderEvent

class RenderEvent : public Event
Event that is used to update any UI overlay system

1.21.65 Class Gsage::RenderSystemWrapper

class RenderSystemWrapper
Subclassed by *Gsage::RocketOgreWrapper*

1.21.66 Class Gsage::RenderTarget

class RenderTarget : public EventSubscriber<RenderTarget>
Subclassed by *Gsage::RttRenderTarget*, *Gsage::WindowRenderTarget*

1.21.67 Class Gsage::RenderTargetFactory

class RenderTargetFactory
Factory creates Ogre::RenderTarget wrapped into Gsage wrapper

1.21.68 Class Gsage::RenderTargetType

class RenderTargetType

1.21.69 Class Gsage::Renderer

class Renderer
Subclassed by *Gsage::ImageRenderer*

1.21.70 Class Gsage::RendererFactory

class RendererFactory

1.21.71 Class Gsage::ResourceManager

class ResourceManager

1.21.72 Class Gsage::RocketContextEvent

class RocketContextEvent : public Event

1.21.73 Class Gsage::RocketOgreWrapper

class RocketOgreWrapper : public EventSubscriber<RocketOgreWrapper>, public Gsage::RenderSystemWrapper

1.21.74 Class Gsage::RocketUIManager

```
class RocketUIManager : public UIManager, public EventSubscriber<RocketUIManager>
```

1.21.75 Class Gsage::RocketUIPlugin

```
class RocketUIPlugin : public IPlugin
```

1.21.76 Class Gsage::RttRenderTarget

```
class RttRenderTarget : public Gsage::RenderTarget
    Wraps Ogre RT target
```

1.21.77 Class Gsage::SDLAudioSystem

```
class SDLAudioSystem
```

1.21.78 Class Gsage::SDLCore

```
class SDLCore : public UpdateListener
```

1.21.79 Class Gsage::SDLEventListener

```
class SDLEventListener
    Subclassed by Gsage::SDLInputListener
```

1.21.80 Class Gsage::SDLInputFactory

```
class SDLInputFactory : public AbstractInputFactory
```

1.21.81 Class Gsage::SDLInputListener

```
class SDLInputListener : public InputHandler, public EventDispatcher, public Gsage::SDLEventListener
```

1.21.82 Class Gsage::SDLPlugin

```
class SDLPlugin : public IPlugin
```

1.21.83 Class Gsage::SDLRenderer

```
class SDLRenderer
    Wraps SDL renderer and call updates on each engine update
```

1.21.84 Class Gsage::SDLWindow

```
class SDLWindow : public Window
```

1.21.85 Class Gsage::SDLWindowManager

```
class SDLWindowManager : public WindowManager
    SDL Window Manager implementation
```

1.21.86 Class Gsage::SceneNodeWrapper

```
class SceneNodeWrapper : public Gsage::OgreObject, public EventSubscriber<SceneNodeWrapper>
```

1.21.87 Class Gsage::ViewportRenderData

```
class ViewportRenderData
```

1.21.88 Class Gsage::WindowEventListener

```
class WindowEventListener : public WindowEventListener
    Proxy ogre window resize events to the engine
```

1.21.89 Class Gsage::WindowRenderTarget

```
class WindowRenderTarget : public Gsage::RenderTarget
    Wrapped Ogre::RenderWindow
```

1.21.90 Class Gsage::WorkspaceEvent

```
class WorkspaceEvent : public Event
    Fired when a new Ogre::CompositorWorkspace is created
```

1.21.91 Class MOC::CollisionTools

```
class CollisionTools
```

1.21.92 Class Ogre::ManualMovableTextRenderer

```
class ManualMovableTextRenderer : public ParticleSystemRenderer
    Renderer that is used to create floating text particles, like damage
```

1.21.93 Class Ogre::ManualMovableTextRendererFactory

```
class ManualMovableTextRendererFactory : public ParticleSystemRendererFactory
```

1.21.94 Class `Ogre::ManualMovableTextRenderer::CmdFontName`

```
class CmdFontName : public ParamCommand
```

1.21.95 Class `Ogre::MeshInformation`

```
class MeshInformation
    A single mesh information
```

1.21.96 Class `Ogre::MeshTools`

```
class MeshTools : public Ogre::Singleton<MeshTools>
    Provides cached access to raw mesh data
```

1.21.97 Class `Ogre::MovableText`

```
class MovableText : public MovableObject, public Renderable, public FrameListener, public MovableObject, public
```

1.21.98 Class `Ogre::MovableTextFactory`

```
class MovableTextFactory : public MovableObjectFactory
    Factory object for creating MovableText instances
```

1.21.99 Class `Ogre::MovableTextValue`

```
class MovableTextValue : public ParticleVisualData
    Movable text string
```

1.21.100 Class `Ogre::TextNode`

```
class TextNode
    Class that represents created movable text node
```

1.21.101 Class `RenderInterfaceOgre3D`

```
class RenderInterfaceOgre3D : public RenderInterface
    A sample render interface for Rocket into Ogre3D.

    Modified by Brett Didemus to work with programable pipeline

    Author Peter Curry
```

1.21.102 Class `SystemInterfaceOgre3D`

```
class SystemInterfaceOgre3D : public SystemInterface
    A sample system interface for Rocket into Ogre3D.

    Author Peter Curry
```

1.22 Struct list

1.22.1 Struct Gsage::ImGuiDockspaceState

```
struct ImGuiDockspaceState
```

1.22.2 Struct Gsage::ImGuiDockspaceState::Dockstate

```
struct Dockstate
```

1.22.3 Struct Gsage::ImGuiDockspaceStyle

```
struct ImGuiDockspaceStyle
```

1.22.4 Struct Gsage::ImGuiRenderer::Context

```
struct Context
```

1.22.5 Struct Gsage::MaterialLoader::FileInfo

```
struct FileInfo
```

1.22.6 Struct Gsage::OgreObject::PendingPropertyUpdate

```
struct PendingPropertyUpdate
```

1.22.7 Struct ImGui::Gradient

```
struct Gradient
```

Subclassed by *ImGui::VerticalGradient*

1.22.8 Struct ImGui::VerticalGradient

```
struct VerticalGradient : public ImGui::Gradient
```

1.23 Namespace list

1.23.1 Namespace Gsage

```
namespace Gsage
```

Typedefs

```
typedef RenderTarget *RenderTargetPtr
typedef std::shared_ptr<Renderer> RendererPtr
typedef std::shared_ptr<Dock> DockPtr
```

Enums

```
enum DockSlotPreviewStyle
    Values:
        DockSlotPreviewFill = 0
        DockSlotPreviewOutline = 1
enum ImGuiDockFlags
    Values:
        ImGuiDock_NoTitleBar = 1 << 0
        ImGuiDock_NoResize = 1 << 1
```

Functions

```
static const Ogre::Vector3 GsageVector3ToOgreVector3 (const Gsage::Vector3 &vector)
static const Gsage::Vector3 OgreVector3ToGsageVector3 (const Ogre::Vector3 &vector)
static const Ogre::Quaternion GsageQuaternionToOgreQuaternion (const
                                                                Gsage::Quaternion
                                                                &quaternion)
static const Gsage::Quaternion OgreQuaternionToGsageQuaternion (const
                                                                Ogre::Quaternion
                                                                &quaternion)
static const Ogre::AxisAlignedBox BoundingBoxToAxisAlignedBox (const    Bounding-
                                                                Box &bbox)
TYPE_CASTER (OgreDegreeCaster, Ogre::Degree, std::string)
TYPE_CASTER (OgreColourValueCaster, Ogre::ColourValue, std::string)
TYPE_CASTER (OgreVector3Caster, Ogre::Vector3, std::string)
TYPE_CASTER (OgreQuaternionCaster, Ogre::Quaternion, std::string)
TYPE_CASTER (OgreFloatRectCaster, Ogre::FloatRect, std::string)
TYPE_CASTER (OgrePixelFormatCaster, Ogre::PixelFormat, std::string)
TYPE_CASTER (RenderOperationTypeCaster, Ogre::RenderOperation::OperationType, std::string)
TYPE_CASTER (RenderTargetTypeCaster, RenderTargetType::Type, std::string)
unsigned long WindowContentViewHandle (SDL_SysWMInfo &info)
static DataProxy dumpState (ImGuiDockspaceState state)
```

```
static ImGuiDockspaceState loadState (DataProxy dp)
```

```
class Animation
```

```
    #include <AnimationScheduler.h> Class that wraps ogre animation state, adds speed definition
```

```
class AnimationController
```

```
    #include <AnimationScheduler.h> Animation controller
```

```
class AnimationGroup
```

```
    #include <AnimationScheduler.h> Container for several animations
```

```
class AnimationScheduler : public Serializable<AnimationScheduler>
```

```
    #include <AnimationScheduler.h> Class that reads all animation information, configures states, then manages all animations playback
```

```
class CustomPass : public CompositorPass
```

```
    Subclassed by Gsage::OverlayPass
```

```
class CustomPassDef : public CompositorPassDef
```

```
    Subclassed by Gsage::OverlayPassDef
```

```
class Gizmo : public EventSubscriber<Gizmo>
```

```
    #include <Gizmo.h> Transformation Gizmo, supports move, scale and rotate operations.
```

Lua usage example:

```
-- create gizmo
gizmo = imgui.createGizmo()

-- setting target
local render = eal:getEntity("test").render
if render == nil then
    exit(1)
end
gizmo:addTarget(render.root)

-- enable
gizmo:enable(true)

-- render on each ImGui cycle
gizmo:render(0, 0, 320, 240)

-- changing mode
gizmo.mode = imgui.gizmo.WORLD

-- changing operation
gizmo.operation = imgui.gizmo.ROTATE

-- draw coordinates editor for this gizmo
-- it is separate to make it possible to draw it in the separate window
gizmo:drawCoordinatesEditor(1, 0, 0, "%.3f", 1,
    "position",
    "scale",
    "rotation")
```

```
class HlmsUnlit : public HlmsUnlit
```

```
    #include <HlmsUnlit.h> Extends Ogre standard HlmsUnlit
```

```
class ImageRenderer : public Gsage::Renderer
    #include <Image.hpp> Really simple image renderer
```

```
class ImGuiDockspace
    #include <ImGuiDockspace.h> ImGui dockspace
```

```
class ImGuiDockspaceRenderer
    #include <ImGuiDockspace.h> Dockspace for ImGui.
```

Provides alternate methods for Begin and End. Docked view should be surrounded by BeginDock and EndDock methods.

```
local flags = 0
local active, open = ImGui.BeginDockOpen("test", true, flags)
if active and open then
    -- render some view here
    ImGui.TextWrapped("Hello World!")
end
ImGui.EndDock()
```

Saving and loading dockspace state:

```
-- save dock state (will get lua table)
local savedState = ImGui.GetDockState()

...

-- restore dock state
ImGui.SetDockState(savedState)
```

```
class ImGuiDockspaceView : public Gsage::ImGuiViewCollection
    #include <ImGuiManager.h> Renders window collection into workspace
```

```
class ImGuiMovableObjectFactory : public MovableObjectFactory
    #include <ImGuiMovableObject.h> Factory object for creating ImGuiMovableObject instances
```

```
class ImGuiOgreRenderer : public EventSubscriber<ImGuiOgreRenderer>, public Gsage::ImGuiRenderer
    Subclassed by Gsage::ImGuiRendererV1, Gsage::ImGuiRendererV2
```

```
class ImGuiRenderer
    Subclassed by Gsage::ImGuiOgreRenderer
```

```
class ImGuiTextBuffer
    #include <ImGuiLuaInterface.h> ImGui buffer that can be used by lua
```

```
class ImGuiViewCollection
    Subclassed by Gsage::ImGuiDockspaceView, Gsage::ImGuiManager
```

```
class IMovableObjectWrapper : public Gsage::OgreObject
    #include <MovableObjectWrapper.h> Provides templateless base class for MovableObjectWrapper

    Subclassed by Gsage::MovableObjectWrapper< T >, Gsage::MovableObjectWrapper< Ogre::Camera >,
    Gsage::MovableObjectWrapper< Ogre::Item >, Gsage::MovableObjectWrapper< Ogre::Light >,
    Gsage::MovableObjectWrapper< Ogre::ManualObject >, Gsage::MovableObjectWrapper< OgreV1::BillboardSet >,
    Gsage::MovableObjectWrapper< OgreV1::Entity >
```

```
class ManualTextureManager : public Listener
    #include <ManualTextureManager.h> This class is used to handle manual texture management and update
```

```
class MaterialLoader : public EventDispatcher, public EventSubscriber<MaterialLoader>
    #include <MaterialLoader.h> TODO: INDEXER

    { "filename": { "checksum": "...", "changed": "...", "materials": [] } }

    for(auto& file : filesystem->ls(folder)) { if(filesystem->extension(file) != "material") { continue; } } Custom OGRE material loader

class OgreLogRedirect : public LogListener
    #include <OgreRenderSystem.h> Class, used to redirect all ogre output to the easylogging++

class OgreObject : public Serializable<OgreObject>
    #include <OgreObject.h> Abstract ogre object

    Subclassed by Gsage::IMovableObjectWrapper, Gsage::ParticleSystemWrapper,
    Gsage::SceneNodeWrapper

class OgreObjectManagerEvent : public Event
    #include <OgreObjectManager.h> Event related to factory lifecycle

class OgreRenderComponent : public EventDispatcher, public RenderComponent
    #include <OgreRenderComponent.h> Ogre render system component

class OgreTexture : public Texture, public Listener
    #include <ManualTextureManager.h> Implements abstract texture class Texture

class OgreView : public EventSubscriber<OgreView>
    #include <OgreView.h> Allows displaying ogre camera into imgui window.
```

Lua usage example:

```
-- create ogre viewport
viewport = imgui.createOgreView("#000000FF")

local textureID = "myOgreView"

-- render camera to texture
local cam = camera:create("free")
cam:renderToTexture(textureID, {
    autoUpdated = false
})

-- set ogre view texture
viewport:setTextureID(textureID)
-- update on each imgui render call
viewport:render(320, 240)
```

```
class Renderer
    Subclassed by Gsage::ImageRenderer

class RenderEvent : public Event
    #include <RenderEvent.h> Event that is used to update any UI overlay system

class RenderSystemWrapper
    Subclassed by Gsage::RocketOgreWrapper

class RenderTarget : public EventSubscriber<RenderTarget>
    Subclassed by Gsage::RttRenderTarget, Gsage::WindowRenderTarget
```

```

class RenderTargetFactory
    #include <RenderTarget.h> Factory creates Ogre::RenderTarget wrapped into Gsage wrapper

class RttRenderTarget : public Gsage::RenderTarget
    #include <RenderTarget.h> Wraps Ogre RT target

class SDLEventListener
    Subclassed by Gsage::SDLInputListener

class SDLRenderer
    #include <SDLRenderer.h> Wraps SDL renderer and call updates on each engine update

class SDLWindowManager : public WindowManager
    #include <SDLWindowManager.h> SDL Window Manager implementation

class WindowEventListener : public WindowEventListener
    #include <WindowEventListener.h> Proxy ogre window resize events to the engine

class WindowRenderTarget : public Gsage::RenderTarget
    #include <RenderTarget.h> Wrapped Ogre::RenderWindow

class WorkspaceEvent : public Event
    #include <WorkspaceEvent.h> Fired when a new Ogre::CompositorWorkspace is created

```

1.23.2 Namespace ImGui

```
namespace ImGui
```

Functions

```

IMGUI_API bool ImGui::Spinner(const char * label, float radius, int thickness, const I

static void AddConvexPolyFilled (ImDrawList *drawList, const ImVec2 *points, const int
    points_count, const Gradient &gradient)

static void PathFillConvex (ImDrawList *drawList, const Gradient &gradient)

struct Gradient
    Subclassed by ImGui::VerticalGradient

```

1.23.3 Namespace ImGuizmo

```
namespace ImGuizmo
```

Enums

```

enum OPERATION
    Values:

    TRANSLATE

    ROTATE

    SCALE

    TRANSLATE

```

```
    ROTATE
    SCALE
    BOUNDS
enum MODE
    Values:
    LOCAL
    WORLD
    LOCAL
    WORLD
enum OPERATION
    Values:
    TRANSLATE
    ROTATE
    SCALE
    TRANSLATE
    ROTATE
    SCALE
    BOUNDS
enum MODE
    Values:
    LOCAL
    WORLD
    LOCAL
    WORLD
```

Functions

```
ImGui_API void ImGui::BeginFrame()
ImGui_API void ImGui::EndFrame()
ImGui_API bool ImGui::IsOver()
ImGui_API bool ImGui::IsUsing()
ImGui_API void ImGui::Enable(bool enable)
ImGui_API void ImGui::DecomposeMatrixToComponents(const float * matrix, float * tra
ImGui_API void ImGui::RecomposeMatrixFromComponents(const float * translation, cons
ImGui_API void ImGui::SetRect(float x, float y, float width, float height)
ImGui_API void ImGui::DrawCube(const float * view, const float * projection, float
ImGui_API void ImGui::Manipulate(const float * view, const float * projection, OPER
ImGui_API void ImGui::SetDrawlist()
```

```

IMGUI_API void ImGuizmo::SetOrthographic(bool isOrthographic)
IMGUI_API void ImGuizmo::DrawCube(const float * view, const float * projection, const
IMGUI_API void ImGuizmo::DrawGrid(const float * view, const float * projection, const

```

1.23.4 Namespace MOC

namespace MOC

1.23.5 Namespace Ogre

namespace Ogre

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

```

class ManualMovableTextRenderer : public ParticleSystemRenderer
    #include <ManualMovableTextRenderer.h> Renderer that is used to create floating text particles, like damage

class MeshInformation
    #include <MeshTools.h> A single mesh information

class MeshTools : public Ogre::Singleton<MeshTools>
    #include <MeshTools.h> Provides cached access to raw mesh data

class MovableTextFactory : public MovableObjectFactory
    #include <MovableText.h> Factory object for creating MovableText instances

class MovableTextValue : public ParticleVisualData
    #include <ManualMovableTextRenderer.h> Movable text string

class TextNode
    #include <ManualMovableTextRenderer.h> Class that represents created movable text node

```

1.23.6 Namespace sol

namespace sol

1.24 File list

1.24.1 File AnimationScheduler.h

Defines

DEFAULT_FADE_SPEED

LOOP

namespace Ogre

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

namespace Gsage

class Animation

#include <AnimationScheduler.h> Class that wraps ogre animation state, adds speed definition

Public Functions

Animation()

virtual ~Animation()

void **initialize** (OgreV1::AnimationState **state*)

void **update** (double *time*)

Calls Ogre::AnimationState addTime with adjusted speed

Parameters

- *time*: Time in seconds

void **enable** (double *time* = DEFAULT_FADE_SPEED)

Enables animation

Parameters

- *time*: Time to fade in

void **disable** (double *time* = DEFAULT_FADE_SPEED)

Disables animation

Parameters

- *time*: Time to fade out

float **getTimePosition** ()

Gets current time position of anim

void **setTimePosition** (double *value*)

Sets current time position of anim

Parameters

- *value*: Time in seconds

```
float getLength ()
    Gets animation length

bool getEnabled ()
    Gets enabled

void setEnabled (bool value)
    Change enabled animation state immediately

Parameters
    • value: Enable anim

bool getLoop ()
    Gets loop enabled

void setLoop (bool value)
    Sets loop value

Parameters
    • value:

bool hasEnded ()
    Gets animation finished (if not looping)

double getSpeed ()
    Gets animation speed

void setSpeed (float speed)
    Sets animation speed

bool isInitialized ()
    Animation has anim state

bool isEnding ()
    Animation is finite and fading out

bool isFadingOut ()
    Animation is disabled but is still fading

void rewind (double offset = 0)
    Rewind animation to the beginning, depends on direction

Parameters
    • offset: If speed >= 0: 0 + offset, else: length - offset
```

Private Members

```
OgreV1::AnimationState *mAnimationState

bool mFadeIn

bool mFadeOut

float mFadeTime

float mSpeed
```

Friends

friend Gsage::AnimationScheduler

class AnimationController
#include <AnimationScheduler.h> Animation controller

Public Functions

AnimationController (*Animation* &animation, double *speed* = 0, double *offset* = 0)

AnimationController (const *AnimationController* &other)

virtual ~AnimationController ()

void **start** ()
Starts animation with predefined parameters

void **finish** ()
Stops animation

bool **hasEnded** ()
Check that wrapped animation ended

AnimationController **operator=** (const *AnimationController* &other)

bool **operator==** (*Animation* &anim)

bool **operator!=** (*Animation* &anim)

Private Members

Animation &mAnimation

float mOffset

float mSpeed

class AnimationGroup
#include <AnimationScheduler.h> Container for several animations

Public Functions

AnimationGroup ()

AnimationGroup (*OgreRenderComponent* *c)

virtual ~AnimationGroup ()

bool **initialize** (const DataProxy &dict, *Ogre::SceneManager* *sceneManager)
Initialize animation groups

double **getSpeed** ()
Gets animations speed

void **setSpeed** (float *value*)

Sets animations speed

Parameters

- *value*: 1 is normal speed

bool **hasEnded** ()

Checks that all animations ended

Private Types

typedef std::map<std::string, *Animation*> **Animations**

Private Members

Animations **mAnimations**

OgreRenderComponent ***mRenderComponent**

float **mSpeed**

Friends

friend Gsage::AnimationScheduler

class AnimationScheduler : public Serializable<*AnimationScheduler*>

#include <AnimationScheduler.h> Class that reads all animation information, configures states, then manages all animations playback

Public Functions

AnimationScheduler ()

virtual ~AnimationScheduler ()

void **setRenderComponent** (*OgreRenderComponent* **c*)

bool **initialize** (**const** DataProxy &*dict*, *Ogre::SceneManager* **sceneManager*, *OgreRenderComponent* **renderComponent*)

Read dict and configure groups, read anim states from entities

Parameters

- *dict*: DataProxy to get settings from
- *sceneManager*: *Ogre::SceneManager* to get entities from
- *renderComponent*: target render component

bool **adjustSpeed** (**const** std::string &*name*, double *speed*)

Adjust animation speed

Return true if animation group exists

Parameters

- *name*: *Animation* group name
- *speed*: *Animation* speed, 1 is normal speed

bool **play** (**const** std::string &name, int times = LOOP, double speed = 0, double offset = 0, bool reset = false)

Play animations from animations group

Return true if animation group exists

Parameters

- name: *Animation* group name
- times: Repeat times, -1 means loop
- speed: *Animation* speed, 1 is normal speed
- offset: *Animation* start offset
- reset: Starts animation immediately

void **resetState** ()

Reset animation state

void **update** (double time)

Update animations

Parameters

- time: Time delta in seconds

Private Types

typedef std::map<std::string, *Animation* *> **Animations**

typedef std::map<std::string, *AnimationGroup*> **AnimationGroups**

typedef std::queue<*AnimationController*> **AnimationQueue**

typedef std::map<std::string, *AnimationQueue*> **AnimationQueues**

Private Functions

void **playDefaultAnimation** ()

Plays default animation, if present

bool **queueAnimation** (*AnimationGroup* &animation, double speed, double offset, bool reset)

Queue animation

Parameters

- animation: *Animation* group to play
- speed: *Animation* speed
- offset: *Animation* offset
- reset: interrupt current animation

bool **isQueued** (**const** std::string &group, *Animation* &anim)

Check that animation is in queue

Parameters

- group: *Animation* group, queue name
- anim: *Animation* instance

void **setStates** (**const** DataProxy &dict)

Set animation states

Parameters

- dict: DataProxy with settings

const DataProxy &**getStates** () **const**

Get animation states serialized

Private Members

```

Ogre::SceneManager *mSceneManager
AnimationQueues mAnimationQueues
AnimationGroups mAnimationGroups
Animations mAnimations
float mDefaultAnimationSpeed
std::string mDefaultAnimation
std::string mCurrentAnimation
DataProxy mAnimationStatesDict
OgreRenderComponent *mRenderComponent
bool mInitialized

```

1.24.2 File BillboardWrapper.h

Defines

```

BBT_POINT_ID
BBT_ORIENTED_COMMON_ID
BBT_ORIENTED_SELF_ID
BBT_PERPENDICULAR_SELF_ID
BBT_PERPENDICULAR_COMMON_ID
namespace Gsage

```

```

class BillboardSetWrapper : public Gsage::MovableObjectWrapper<OgreV1::BillboardSet>

```

Public Functions

```

BillboardSetWrapper ()

virtual ~BillboardSetWrapper ()

bool read (const DataProxy &dict)
    Override default logic values reading

void setCommonUpVector (const Ogre::Vector3 &vector)
    Set common up vector of the billboard
    Parameters
    • vector: Vector3

const Ogre::Vector3 &getCommonUpVector () const
    Get common up vector of the billboard

void setCommonDirection (const Ogre::Vector3 &vector)
    Set common direction vector of the billboard

```

Parameters

- `vector`: Vector3

const *Ogre::Vector3* &**getCommonDirection** () **const**

Get direction vector of the billboard

void **setBillboardType** (**const** std::string &*type*)

Set billboard type

See *Ogre::BillboardType* for the list of supported ids. It is the same as it is in the enum

Parameters

- `type`: Type id

std::string **getBillboardType** ()

Get billboard type

void **setMaterialName** (**const** std::string &*id*)

Set billboard material

Parameters

- `id`: Material id

const std::string &**getMaterialName** () **const**

Get billboard material

void **setBillboards** (**const** DataProxy &*dict*)

Add billboards to the billboard set

DataProxy **getBillboards** ()

Get billboards to the billboard set

Public Static Attributes

const std::string **TYPE**

Private Types

typedef std::vector<*BillboardWrapper*> **Billboards**

Private Members

OgreV1::BillboardSet ***mBillboardSet**

Billboards **mBillboards**

std::string **mMaterialName**

Private Static Functions

static std::string **mapBillboardType** (**const** OgreV1::BillboardType *type*)

Convert ogre type enum to string

Parameters

- `type`: Enum value of the type

static OgreV1::BillboardType **mapBillboardType** (**const** std::string &*type*)

Convert string type id to ogre type enum

Parameters

- type: String id

```
class BillboardWrapper : public Serializable<BillboardWrapper>
```

Public Functions

```
BillboardWrapper()
```

```
virtual ~BillboardWrapper()
```

```
bool initialize(const DataProxy &dict, OgreV1::BillboardSet *billboardSet)
```

Initialize billboard

Return false if fails

Parameters

- dict: DataProxy with values
- billboardSet: Ogre::BillboardSet to create billboard into

```
void setPosition(const Ogre::Vector3 &position)
```

Set billboard position

Parameters

- position: Position of the billboard object

```
const Ogre::Vector3 &getPosition() const
```

Get billboard position

```
void setWidth(const float &value)
```

Set billboard width

Parameters

- value: Billboard width

```
float getWidth()
```

Get billboard width

```
void setHeight(const float &value)
```

Set billboard height

Parameters

- value: Billboard height

```
float getHeight()
```

Get billboard height

```
void setColour(const Ogre::ColourValue &value)
```

Set billboard colour

Parameters

- value: Billboard colour ARGB

```
const Ogre::ColourValue &getColour() const
```

Get billboard colour

```
void setRotation(const Ogre::Degree &value)
```

Set billboard rotation

Parameters

- value: Billboard rotation

```
Ogre::Degree getRotation()
```

Get billboard rotation

```
void setTexcoordRect (const Ogre::FloatRect &rect)
    Set Texcoord Rect

const Ogre::FloatRect &getTexcoordRect () const
    Get Texcoord Rect

void setTexcoordIndex (const unsigned int &index)
    Set Texcoord Index

unsigned int getTexcoordIndex ()
    Get Texcoord Index
```

Private Members

```
OgreV1::BillboardSet *mBillboardSet
OgreV1::Billboard *mBillboard
float mWidth
float mHeight
```

1.24.3 File CameraWrapper.h

namespace Ogre

File: MovableText.h

Description: This creates a billboarding object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

namespace Gsage

```
class CameraWrapper : public Gsage::MovableObjectWrapper<Ogre::Camera>, public Listener
```

Public Functions

```
CameraWrapper ()

virtual ~CameraWrapper ()

void setOrientation (const Ogre::Quaternion &orientation)
    Sets camera orientation

Parameters
    • orientation: Ogre::Quaternion

const Ogre::Quaternion &getOrientation () const
    Gets camera orientation
```

```

void attach (Ogre::Viewport *viewport)
    Activate this camera

void attach (RenderTargetPtr renderTarget)
    Attach the camera to render target

Parameters
    • renderTarget: RenderTarget

void detach ()
    Detach camera from the render target

void createCamera (const std::string &name)
    Create camera
Parameters
    • name: Camera name

Ogre::Camera *getCamera ()
    Get camera ogre object

const Ogre::Camera *getCamera () const
    Get camera ogre object

const std::string &getName () const
    Get camera name

void setClipDistance (const float &value)
    Set camera clip distance
Parameters
    • value: Clip distance

float getClipDistance ()
    Get camera clip distance

void setBgColour (const Ogre::ColourValue &colour)
    Set viewport background colour
Parameters
    • colour: ColourValue

const Ogre::ColourValue &getBgColour () const
    Get viewport background colour

bool isActive () const
    Check if camera is active

const Ogre::Matrix4 &getViewMatrix () const
    Get view matrix

const Ogre::Matrix4 &getProjectionMatrix () const
    Get projection matrix

void destroy ()
    Remove camera

void objectDestroyed (Ogre::MovableObject *cam)

RenderTargetPtr getRenderTarget ()
    Get render target camera attached to

```

Public Static Attributes

const std::string **TYPE**

Private Members

std::string **mTarget**

Ogre::ColourValue **mBgColour**

Ogre::Viewport ***mViewport**

Ogre::RenderWindow ***mWindow**

bool **mIsActive**

RenderTargetPtr **mRenderTarget**

1.24.4 File CollisionTools.h

namespace MOC

class CollisionTools

Public Functions

CollisionTools (*Ogre::SceneManager* *sceneMgr)

~CollisionTools ()

bool **raycastFromCamera** (int width, int height, *Ogre::Camera* *camera, **const** *Ogre::Vector2* &mousecoords, *Ogre::Vector3* &result, *Ogre::MovableObject* *&target, float &closest_distance, **const** *Ogre::uint32* queryMask = 0xFFFFFFFF)

bool **raycastFromCamera** (int width, int height, *Ogre::Camera* *camera, **const** *Ogre::Vector2* &mousecoords, *Ogre::Vector3* &result, *OgreV1::Entity* *&target, float &closest_distance, **const** *Ogre::uint32* queryMask = 0xFFFFFFFF)

bool **collidesWithEntity** (**const** *Ogre::Vector3* &fromPoint, **const** *Ogre::Vector3* &toPoint, **const** float collisionRadius = 2.5f, **const** float rayHeightLevel = 0.0f, **const** *Ogre::uint32* queryMask = 0xFFFFFFFF)

void **calculateY** (*Ogre::SceneNode* *n, **const** bool doTerrainCheck = true, **const** bool doGridCheck = true, **const** float gridWidth = 1.0f, **const** *Ogre::uint32* queryMask = 0xFFFFFFFF)

float **getTSMHeightAt** (**const** float x, **const** float z)

bool **raycastFromPoint** (**const** *Ogre::Vector3* &point, **const** *Ogre::Vector3* &normal, *Ogre::Vector3* &result, *Ogre::MovableObject* *&target, float &closest_distance, **const** *Ogre::uint32* queryMask = 0xFFFFFFFF)

```

bool raycastFromPoint (const Ogre::Vector3 &point, const Ogre::Vector3 &normal,
                      Ogre::Vector3 &result, OgreV1::Entity *&target, float &closest_distance, const Ogre::uint32 queryMask = 0xFFFFFFFF)

bool raycast (const Ogre::Ray &ray, Ogre::Vector3 &result, Ogre::MovableObject *&target,
              float &closest_distance, const Ogre::uint32 queryMask = 0xFFFFFFFF)

bool raycast (const Ogre::Ray &ray, Ogre::Vector3 &result, OgreV1::Entity *&target, float
              &closest_distance, const Ogre::uint32 queryMask = 0xFFFFFFFF)

void setHeightAdjust (const float heightadjust)

float getHeightAdjust (void)

```

Public Members

```

Ogre::RaySceneQuery *mRaySceneQuery
Ogre::RaySceneQuery *mTSMRaySceneQuery
Ogre::SceneManager *mSceneMgr

```

Private Members

```
float _heightAdjust
```

1.24.5 File CustomPassProvider.h

```
namespace Gsage
```

```
class CustomPassProvider : public CompositorPassProvider
```

Public Functions

```

CustomPassProvider ()

virtual ~CustomPassProvider ()

void initialize (Engine *engine)

Ogre::CompositorPass *addPass (const Ogre::CompositorPassDef *definition, Ogre::Camera
                              *defaultCamera, Ogre::CompositorNode *parentNode, const
                              Ogre::CompositorChannel &target, Ogre::SceneManager
                              *sceneManager)

Ogre::CompositorPassDef *addPassDef (Ogre::CompositorPassType passType, Ogre::IdString
                                     customId, Ogre::CompositorTargetDef *parentTarget-
                                     Def, Ogre::CompositorNodeDef *parentNodeDef)

template<class C, class D>
bool registerPassDef (const std::string &customId)

```

Private Types

```
typedef std::function<Ogre::CompositorPassDef * (Ogre::CompositorTargetDef *,
                                                Ogre::CompositorNodeDef *) >
        DefFactoryFunc

typedef std::map<Ogre::IdString, DefFactoryFunc> DefFactoryFuncs

typedef std::function<Ogre::CompositorPass * (const Ogre::CompositorPassDef *definition,
                                              const Ogre::CompositorChannel &target,
                                              Ogre::CompositorNode *parentNode) >
        PassFactoryFunc

typedef std::map<std::string, PassFactoryFunc> PassFactoryFuncs
```

Private Members

```
Engine *mEngine
DefFactoryFuncs mDefFactories
PassFactoryFuncs mPassFactories
```

1.24.6 File CustomPass.h

```
namespace Gsage
```

```
class CustomPass : public CompositorPass
    Subclassed by Gsage::OverlayPass
```

Public Functions

```
CustomPass (Engine *engine, const Ogre::CompositorPassDef *def, const
            Ogre::CompositorChannel &target, Ogre::CompositorNode *parentNode)

virtual ~CustomPass ()
```

Protected Attributes

```
Engine *mEngine
```

```
class CustomPassDef : public CompositorPassDef
    Subclassed by Gsage::OverlayPassDef
```

Public Functions

```
CustomPassDef (const std::string &id, Engine *engine, Ogre::CompositorTargetDef *target,
               Ogre::CompositorNodeDef *node)

virtual ~CustomPassDef ()

const std::string &getID () const
    Get pass def identifier
```

Protected Attributes

```
const std::string mID
Engine *mEngine
```

1.24.7 File Definitions.h

Defines

```
GSAGE_OGRE_PLUGIN_API
```

```
OgreV1
```

```
GET_IV_DATA (variable)
```

1.24.8 File EntityWrapper.h

```
namespace Ogre
```

```
File: MovableText.h
```

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

```
namespace Gsage
```

```
class EntityWrapper : public Gsage::MovableObjectWrapper<OgreV1::Entity>
```

Public Types

```
enum Query
```

```
Values:
```

```
    STATIC = 0x01
```

```
    DYNAMIC = 0x02
```

```
    UNKNOWN = 0x04
```

Public Functions

```
EntityWrapper ()
```

```
virtual ~EntityWrapper ()
```

```
void setQueryFlags (const std::string &type)
```

```
Set model entity flags, used for raycasting and querying entities
```

```
Parameters
```

- `type`: static means that entity is a part of location, dynamic means that entity is some actor

const std::string &**getQueryFlags** () **const**

Get query flags of the model

void **setResourceGroup** (**const** std::string &*name*)

Set model resource group

Parameters

- `name`: Resource group name, e.g. General

const std::string &**getResourceGroup** () **const**

Get resource group name

void **setMesh** (**const** std::string &*mesh*)

Set model mesh (this function creates entity)

Parameters

- `mesh`: Mesh file name

const std::string &**getMesh** () **const**

Get mesh file name

void **setCastShadows** (**const** bool &*value*)

Set cast shadows

Parameters

- `value`: Cast shadows

bool **getCastShadows** ()

Get cast shadows

void **setRenderQueue** (**const** unsigned int &*queue*)

Set render queue

Parameters

- `queue`: queue id

unsigned int **getRenderQueue** ()

Get render queue

void **attachToBone** (**const** DataProxy &*params*, **const** std::string &*entityId*, DataProxy *movableObjectData*)

Attach another entity to the bone

Parameters

- `params`: should contain boneID field
- `entityId`: Id of new entity
- `movableObjectData`: MovableObject to create and attach

bool **attach** (*Ogre::MovableObject *object*, **const** DataProxy &*params*)

Attach another entity to the bone

Parameters

- `params`: must contain boneID

OgreV1::Entity ***getEntity** ()

Get underlying entity

void **createCloneWithMaterial** (*Ogre::MaterialPtr material*, *Ogre::uint8 renderQueue*)

Clone and assign entity material

Parameters

- `material`:

```
void removeClone ()
    Remove cloned entity
```

```
Ogre::AxisAlignedBox getAabb () const
    Get AABB
```

Public Static Attributes

```
const std::string TYPE
```

Private Types

```
typedef std::vector<OgreObject *> AttachedEntities
```

Private Members

```
OgreV1::SkeletonAnimationBlendMode mAnimBlendMode
```

```
std::string mMeshName
```

```
std::string mQueryString
```

```
std::string mResourceGroup
```

```
Query mQuery
```

```
AttachedEntities mAttachedEntities
```

```
OgreV1::Entity *mClone
```

1.24.9 File Factory.hpp

```
namespace Gsage
```

```
class RendererFactory
```

Public Functions

```
RendererPtr create (const DataProxy &params, SDL_Renderer *renderer, SDLCore *core,
    WindowPtr window)
```

1.24.10 File Gizmo.h

```
namespace Ogre
```

```
File: MovableText.h
```

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

namespace Gsage

class Gizmo : public EventSubscriber<*Gizmo*>

#include <Gizmo.h> Transformation *Gizmo*, supports move, scale and rotate operations.

Lua usage example:

```
-- create gizmo
gizmo = imgui.createGizmo()

-- setting target
local render = eal:getEntity("test").render
if render == nil then
    exit(1)
end
gizmo:addTarget(render.root)

-- enable
gizmo:enable(true)

-- render on each ImGui cycle
gizmo:render(0, 0, 320, 240)

-- changing mode
gizmo.mode = imgui.gizmo.WORLD

-- changing operation
gizmo.operation = imgui.gizmo.ROTATE

-- draw coordinates editor for this gizmo
-- it is separate to make it possible to draw it in the separate window
gizmo:drawCoordinatesEditor(1, 0, 0, "%.3f", 1,
    "position",
    "scale",
    "rotation")
```

Public Functions

Gizmo (*OgreRenderSystem* *rs)

virtual ~Gizmo ()

void **render** (float x, float y, **const** std::string &rttName)
Render gizmo UI

void **enable** (bool value)
Enable/disable gizmo

void **addTarget** (*SceneNodeWrapper* *target)
Set target

void **removeTarget** (**const** *SceneNodeWrapper* *target)
Remove target

```

void resetTargets ()
    Reset all targets in Gizmo

void setOperation (ImGizmo::OPERATION value)
    Set gizmo operation

ImGizmo::OPERATION getOperation () const
    Get gizmo operation

void setMode (ImGizmo::MODE value)
    Set gizmo mode

ImGizmo::MODE getMode () const
    Get gizmo mode

bool drawCoordinatesEditor (float v_speed = 1.0f, float v_min = 0.0f, float v_max = 0.0f,
                             const std::string &display_format = "%.3f", float power
                             = 1.0f, const std::string &posLabel = "position", const
                             std::string &scaleLabel = "scale", const std::string &rotationLabel = "rotation")
    Draw coordinates editor

```

Private Types

```

typedef std::vector<SceneNodeWrapper *> Targets
typedef std::map<const SceneNodeWrapper *, Ogre::Vector3> Positions

```

Private Functions

```

const Ogre::Vector3 &getPosition ()

const Ogre::Vector3 &getScale ()

const Ogre::Quaternion &getOrientation ()

bool onTargetDestroyed (EventDispatcher *sender, const Event &event)

void updateTargetNode (bool reset = true)

void rotate (float delta[3])

void setScale (float scale[3])

bool extractMatrix (Ogre::Matrix4, float *dest, int length)

```

Private Members

```

OgreRenderSystem *mRenderSystem
Targets mTargets
Positions mPositions
Positions mInitialPositions
Positions mInitialScales

```

```
float mModelMatrix[16]
Ogre::Vector3 mPosition
Ogre::Vector3 mScale
Ogre::Quaternion mOrientation
ImGuizmo::OPERATION mOperation
ImGuizmo::MODE mMode
bool mEnabled
bool mUsing
```

1.24.11 File GsageOgrePlugin.h

```
namespace Gsage
```

```
class GsageOgrePlugin : public IPlugin
```

Public Functions

```
GsageOgrePlugin ()
```

```
virtual ~GsageOgrePlugin ()
```

```
virtual const std::string &getName () const
    Get ogre plugin name
```

```
virtual bool installImpl ()
    Registers OgreRenderSystem and RecastNavigationSystem
```

```
virtual void uninstallImpl ()
    Unregisters OgreRenderSystem and RecastNavigationSystem
```

```
virtual void setupLuaBindings ()
    Set up lua bindings
```

1.24.12 File HlmsUnlitDatablock.h

```
namespace Gsage
```

```
class HlmsUnlitDatablock : public HlmsUnlitDatablock
```

Public Functions

```
HlmsUnlitDatablock (Ogre::IdString name, HlmsUnlit *creator, const
Ogre::HlmsMacroblock *macroblock, const Ogre::HlmsBlendblock
*blendblock, const Ogre::HlmsParamVec &params)
```

Friends

```
friend Gsage::HlmsUnlit
```

1.24.13 File HlmsUnlit.h

```
namespace Gsage
```

```
class HlmsUnlit : public HlmsUnlit
    #include <HlmsUnlit.h> Extends Ogre standard HlmsUnlit
```

Public Functions

```
HlmsUnlit (Ogre::Archive *dataFolder, Ogre::ArchiveVec *libraryFolders)
```

```
HlmsUnlit (Ogre::Archive *dataFolder, Ogre::ArchiveVec *libraryFolders, Ogre::HlmsTypes
    type, const Ogre::String &typeName)
```

```
virtual ~HlmsUnlit ()
```

```
virtual Ogre::uint32 fillBuffersForV2 (const Ogre::HlmsCache *cache, const
    Ogre::QueuedRenderable &queuedRenderable,
    bool casterPass, Ogre::uint32 lastCacheHash,
    Ogre::CommandBuffer *commandBuffer)
```

```
void setUseCustomProjectionMatrix (Ogre::IdString name, const Ogre::Matrix4 &ma-
    trix)
```

Make *HlmsUnlit* use the same projection matrix for all renderables which are using datablock with the specified name.

Parameters

- name: Datablock name
- matrix: Ogre::Matrix4 to use

Public Static Functions

```
static void getDefaultPaths (Ogre::String &outDataFolderPath, Ogre::StringVector &out-
    LibraryFoldersPaths)
```

Protected Types

```
typedef std::map<Ogre::IdString, Ogre::Matrix4> CustomProjectionMatrices
```

Protected Functions

```
virtual Ogre::HlmsDatablock *createDatablockImpl (Ogre::IdString datablockName,
    const Ogre::HlmsMacroblock
    *macroblock, const
    Ogre::HlmsBlendblock
    *blendblock, const
    Ogre::HlmsParamVec &paramVec)
```

Protected Attributes

CustomProjectionMatrices `mCustomProjectionMatrices`

1.24.14 File ImGuiConverters.h

1.24.15 File ImGuiDockspaceState.h

`namespace Gsage`

Functions

`static DataProxy dumpState (ImGuiDockspaceState state)`

`static ImGuiDockspaceState loadState (DataProxy dp)`

1.24.16 File ImGuiDockspace.h

Defines

`IMGUI_DEFINE_MATH_OPERATORS`

`namespace Gsage`

Typedefs

`typedef std::shared_ptr<Dock> DockPtr`

Enums

`enum DockSlotPreviewStyle`

Values:

`DockSlotPreviewFill = 0`

`DockSlotPreviewOutline = 1`

`enum ImGuiDockFlags`

Values:

`ImGuiDock_NoTitleBar = 1 << 0`

`ImGuiDock_NoResize = 1 << 1`

`class Dock`

Public Types

`enum Layout`

Values:

`None = 0`

```

    Vertical = 1
    Horizontal = 2
enum Location
    Values:
    Unspecified = 10
    Left = 1
    Right = Left ^ 1
    Top = 3
    Bottom = Top ^ 1
    Tab = 5
    Root = 6

```

Public Functions

```

Dock (const char *label, const ImGuiDockspaceStyle &mStyle)
virtual ~Dock ()
void reset ()
    Called when dock is detached
void activateOther ()
    Activate any other tab if there is one Descending order is preferable
bool isContainer () const
    Check if dock is container should have any children present
bool hasBothChildren () const
    Check if container has both children open
bool bothChildrenVisible () const
    Check if both children are visible
void updateChildren ()
    Update children in container
    Update dimensions, positions and draw splits
void setDimensions (ImVec2 pos, ImVec2 size)
    Update dock dimensions
    Parameters
    • pos: Position
    • size: Size
float getRatio () const
    Get first dock scale ratio
void setRatio (float ratio)
    Set first dock scale ratio
    Parameters
    • ratio: from 0 to 1

```

const ImVec2 &getPosition () const
 Get position

const ImVec2 &getSize () const
 Get size

bool **getOpened () const**
 Check if dock is opened

bool **anyTabOpen () const**
 Check if this tab or any next tab is open

void **setOpened** (bool *value*)
 Change opened state for dock

Parameters

- *value*:

bool **getActive () const**
 Check if dock is active (tab wise)

void **setActive** (bool *value*)
 Set dock as active

Parameters

- *value*:

bool **visible () const**
 Check if the dock should be drawn

bool **docked () const**
 Is docked to anything

void **addChild** (*DockPtr child*)
 Add child to location defined in child itself

Parameters

- *child*: DockPtr

void **addChildAt** (*DockPtr child, Location location*)
 Add child to container at location

Parameters

- *child*: DockPtr
- *location*: Location

void **removeChildAt** (*Location location*)
 Remove child from location

Parameters

- *location*: Location

DockPtr **getChildAt** (*Location location*)
 Get child at specified location

DockPtr **getChildAt** (int *index*)
 Get child at specified index

Dock::Layout **getLayout () const**
 Get dock layout

Dock::Location **getLocation () const**
Get dock location

DockPtr **getParent ()**
Get dock parent

DockPtr **getNextTab ()**
Get next tab

DockPtr **getPreviousTab ()**
Get previous tab

const char *getLabel () const
Get dock label

const char *getTitle () const
Get dock title

const ImRect &getBoundingRect () const
Get bounding rect

float **calculateRatio** (*Dock::Layout* layout, **const** ImVec2 &containerSize)
Calculate ratio for the dock, basing on dock location

Parameters

- container: destination container

Private Functions

int **locationToIndex** (*Location* location) **const**

Private Members

char ***mLabel**

char ***mTitle**

ImVec2 **mPos**

ImVec2 **mSize**

ImRect **mBounds**

DockPtr **mChildren[2]**

DockPtr **mNextTab**

DockPtr **mPreviousTab**

DockPtr **mParent**

Layout **mLayout**

Location **mLocation**

bool **mActive**

bool **mOpened**

bool **mRendered**

bool **mResized**

```
bool mDirty
float mRatio
int mFlags
const ImGuiDockspaceStyle &mStyle
```

Friends

```
friend Gsage::ImGuiDockspace
friend Gsage::ImGuiDockspaceRenderer
```

```
class ImGuiDockspace
    #include <ImGuiDockspace.h> ImGui dockspace
```

Public Functions

```
ImGuiDockspace ()
```

```
ImGuiDockspace (const ImGuiDockspaceStyle &style)
```

```
virtual ~ImGuiDockspace ()
```

```
ImGuiDockspaceState getState ()
```

Get dockspace state

Return current *ImGuiDockspaceState*

```
void setState (ImGuiDockspaceState state)
```

Set dockspace state

Parameters

- state: *ImGuiDockspaceState*

```
void setDimensions (ImVec2 pos, ImVec2 size)
```

Update dockspace position and dimensions

Parameters

- pos: Position
- size: Size

```
void updateLayout ()
```

Update layout

```
bool dockTo (const char *label, DockPtr dock, Dock::Location location)
```

Dock window

Parameters

- label: parent dock label
- location: location to dock to

```
bool undock (DockPtr dock, bool destroy = false)
```

Undock dock from any kind of container

Parameters

- dock: *DockPtr*
- destroy: removes dock object completely, clearing the dock state

bool **undock** (**const** char **label*, bool *destroy* = false)

Undock dock from any kind of container

Parameters

- *label*: dock label
- *destroy*: removes dock object completely, clearing the dock state

DockPtr **getRootDock** ()

Get root dock

DockPtr **getDock** (**const** char **label*)

Get window state calculated by `updateLayout` call

Parameters

- *label*: identifies window uniquely

DockPtr **getDock** (**const** std::string &*label*)

ImGuiDockspace::getDock

DockPtr **createDock** (**const** char **label*, bool *opened* = true, bool *active* = true)

Create new dock

Parameters

- *label*: dock label

void **addChild** (*DockPtr* *container*, *DockPtr* *child*)

Add child dock to container

Parameters

- *container*:
- *child*:

void **reset** ()

Reset dockspace, remove all docks

DockPtr **getDockAt** (**const** ImVec2 &*position*)

Get dock under mouse position

Parameters

- ImVec2: position

Private Types

typedef std::map<std::string, *DockPtr*> **Docks**

typedef std::map<*Dock::Location*, *Dock::Layout*> **LocationToLayout**

Private Members

ImGuiDockspaceStyle **mStyle**

Docks **mDocks**

LocationToLayout **mLocationToLayout**

DockPtr **mRootDock**

ImVec2 **mSize**

Friends

friend `Gsage::ImGuiDockspaceRender`

class `ImGuiDockspaceRender`

#include <ImGuiDockspace.h> Dockspace for ImGui.

Provides alternate methods for `Begin` and `End`. Docked view should be surrounded by `BeginDock` and `EndDock` methods.

```
local flags = 0
local active, open = ImGui.BeginDockOpen("test", true, flags)
if active and open then
    -- render some view here
    ImGui.TextWrapped("Hello World!")
end
ImGui.EndDock()
```

Saving and loading dockspace state:

```
-- save dock state (will get lua table)
local savedState = ImGui.GetDockState()

...

-- restore dock state
ImGui.SetDockState(savedState)
```

Public Functions

`ImGuiDockspaceRender()`

`ImGuiDockspaceRender(const ImGuiDockspaceStyle &style)`

`virtual ~ImGuiDockspaceRender()`

`bool activateDock(const char *label)`

Activate dock

Return true if succeed

Parameters

- `label`: Dock id

`void beginWorkspace(ImVec2 pos, ImVec2 size)`

Must be called before rendering all windows

`bool begin(const char *label, bool *opened, ImGuiWindowFlags windowFlags)`

Begin rendering

Parameters

- `label`: Window label
- `opened`: Pointer to bool
- `windowFlags`: additional window flags

`bool begin(const char *label, const char *title, bool *opened, ImGuiWindowFlags windowFlags, int dockFlags)`

Begin rendering

Parameters

- `label`: Window label
- `opened`: Pointer to bool
- `windowFlags`: additional window flags
- `dockFlags`: additional dock flags

bool **begin**(const char **label*, const char **title*, bool **opened*, ImGuiWindowFlags *windowFlags*)

Begin rendering

Parameters

- `label`: Window label
- `title`: Actual window name
- `opened`: Pointer to bool
- `windowFlags`: additional window flags

void **end**()

End rendering

void **endWorkspace**(bool *cleanup* = false)

Must be called after rendering all windows

Parameters

- `cleanup`: do cleanup of docks that were not rendered this time

ImGuiDockspaceState **getState**()

Get workspace state

void **setState**(const *ImGuiDockspaceState* &*state*)

Set workspace state

Parameters

- `state`:

Private Types

enum **EndCommand**

Values:

End_None = 0

End_Child = 1

End_Window = 2

typedef std::map<std::string, bool> **Windows**

Private Functions

bool **tabbar**(*DockPtr* *dock*, bool *closeButton*)

void **title**(*DockPtr* *dock*)

void **dockWindows**()

void **splitters**()

void **handleDragging**()

```
bool dockSlots (DockPtr destDock, const ImVec2 &rect, const std::vector<Dock::Location>
               &locations)
```

Private Members

```
ImGuiDockspace mDockspace
```

```
ImGuiDockspaceStyle mStyle
```

```
EndCommand mEndCommand
```

```
ImVec2 mPos
```

```
ImVec2 mSize
```

```
bool mPopClipRect
```

```
Windows mDockableWindows
```

```
DockPtr mDraggedDock
```

```
bool mStateWasUpdated
```

```
struct ImGuiDockspaceState
```

Public Types

```
typedef std::map<std::string, Dockstate> Docks
```

Public Members

```
Docks docks
```

```
ImVec2 size
```

```
struct Dockstate
```

Public Functions

```
Dockstate ()
```

```
Dockstate (DockPtr dock)
```

```
virtual ~Dockstate ()
```

Public Members

```
float ratio
```

```
std::string children[2]
```

```
std::string next
```

```
std::string prev
```

```
std::string parent
```

```
Dock::Layout layout
```

```

    Dock::Location location
    bool active
    bool opened

struct ImGuiDockspaceStyle

```

Public Members

```

float splitterThickness
float tabbarHeight
float tabbarTextMargin
float tabbarPadding
float windowRounding
ImU32 windowBGColor
ImU32 tabInactiveColor
ImU32 tabActiveColor
ImU32 tabHoveredColor
ImU32 textColor
ImU32 splitterHoveredColor
ImU32 splitterNormalColor
ImU32 dockSlotHoveredColor
ImU32 dockSlotNormalColor
DockSlotPreviewStyle dockSlotPreview

```

1.24.17 File Image.hpp

```
namespace Gsage
```

```

class ImageRenderer : public Gsage::Renderer
    #include <Image.hpp> Really simple image renderer

```

Public Functions

```

ImageRenderer (const DataProxy &params, SDL_Renderer *renderer, SDLCore *core, Win-
    dowPtr window)

~ImageRenderer ()

void render ()

```

Private Members

```
SDL_Surface *mImage
SDL_Texture *mTexture
SDL_Renderer *mRenderer
SDL_Rect mDestRect
bool mCustomRect
```

1.24.18 File ImGuiDefinitions.h

Defines

```
RENDER_QUEUE_IMGUI
IMGUI_MATERIAL_NAME
```

1.24.19 File ImGuiEvent.h

```
namespace Gsage
```

```
class ImGuiEvent : public Event
```

Public Functions

```
ImGuiEvent (Event::ConstType type, const std::string &contextName)
virtual ~ImGuiEvent ()
```

Public Members

```
std::string mContextName
```

Public Static Attributes

```
const Event::Type CONTEXT_CREATED
```

1.24.20 File ImGuiExportHeader.h

Defines

```
IMGUI_PLUGIN_API
```

1.24.21 File ImGuiImage.h

namespace Gsage

class ImGuiImage

Public Functions

ImGuiImage (const std::string &name, RenderSystem *render)

virtual ~ImGuiImage ()

void **render** (unsigned int *width*, unsigned int *height*)
Render image

Parameters

- *width*: Image width
- *height*: Image height

TexturePtr **getTexture** ()
Get texture object

Private Members

TexturePtr **mTexture**

RenderSystem ***mRender**

std::string **mName**

1.24.22 File ImGuiLuaInterface.h

namespace Gsage

class ImGuiLuaInterface

Public Static Functions

static void **addLuaBindings** (sol::state_view &lua)
Add ImGui bindings to the lua state

Parameters

- *lua*: state to enrich with imgui bindings

class ImGuiTextBuffer

#include <ImGuiLuaInterface.h> ImGui buffer that can be used by lua

Public Functions

ImGuiTextBuffer (int *size*, **const** char **initialValue* = "")

virtual ~**ImGuiTextBuffer** ()

std::string **read** () **const**

Read the buffer

char ***read** ()

Read the buffer

bool **write** (**const** std::string &*value*)

Overwrite the buffer.

Parameters

- *value*: to write

int **size** () **const**

Returns buffer size.

Private Members

char ***mBuffer**

int **mSize**

1.24.23 File ImGuiLualterator.h

1.24.24 File ImGuiManager.h

namespace **Gsage**

class **ImGuiDockspaceView**: **public** *Gsage::ImGuiViewCollection*
#include <ImGuiManager.h> Renders window collection into workspace

Public Functions

ImGuiDockspaceView (*ImGuiManager* **manager*, **const** std::string &*name*)

void **render** (int *x*, int *y*, int *width*, int *height*)

Render views

Parameters

- *x*: dockspace x
- *y*: dockspace y
- *width*: dockspace width
- *height*: dockspace height

sol::table **getState** ()

Get dockspace state

void **setState** (*sol::table* *t*)

Set dockspace state

Parameters

- `t`: `sol::table` to read state data from

bool **activateDock** (const std::string &name)

Activate dock

Return true if dock was found

Parameters

- name: *Dock* name

virtual bool **addView** (const std::string &name, *sol::object* view)

Add new lua view to the imgui renderer.

Parameters

- name: view name
- view: lua function that will render everything.

virtual bool **removeView** (const std::string &name, *sol::object* view)

Remove view from dockspace

Private Members

ImGuiManager *mManager

DataProxy mState

std::unique_ptr<*ImGuiDockspaceRenderer*> mDockspace

std::string mName

class **ImGuiManager** : public UIManager, public EventSubscriber<*ImGuiManager*>, public *Gsage::ImGuiViewColl*

Public Types

typedef std::function<*ImGuiRenderer* * ()> **RendererFactory**

Public Functions

ImGuiManager ()

virtual ~**ImGuiManager** ()

virtual void **initialize** (GsageFacade *facade, lua_State *L = 0)

Initialize ui manager

Parameters

- called: by gsage facade on setup
- facade: Gsage Facade
- L: init with lua state

lua_State ***getLuaState** ()

Gets Rocket lua state

void **setLuaState** (lua_State *L)

Update load state

Parameters

- L: lua_State

void **setUp** ()

Configures rendering

void **tearDown** ()

Tear down imgui manager

bool **handleSystemChange** (EventDispatcher **sender*, **const** Event &*event*)

SystemChangeEvent::SYSTEM_ADDED and SystemChangeEvent::SYSTEM_REMOVED handler

bool **handleMouseEvent** (EventDispatcher **sender*, **const** Event &*event*)

Handle mouse event from engine

Parameters

- *sender*: EventDispatcher
- *event*: Event

const std::string &**getType** ()

void **addRendererFactory** (**const** std::string &*type*, *RendererFactory* *f*)

This method is called by render system interfaces plugins

Parameters

- *type*: Type of render system to use for. RenderSystem must properly return “type” field from `getSystemInfo` call
- *f*: Factory method to create renderables

void **removeRendererFactory** (**const** std::string &*type*)

Unregister renderer factory

ImGuiContext ***getImGuiContext** (std::string *name*, **const** ImVec2 &*initialSize*)

Get or create new ImGuiContext instance

Parameters

- *name*: Context name
- *initialSize*: Initial context size

virtual void **renderViews** (*ImGuiRenderer::Context* &*ctx*)

Render a single imgui frame

Public Static Attributes

const std::string **TYPE**

Private Types

typedef std::map<std::string, *RendererFactory*> **RendererFactories**

typedef std::vector<ImVector<ImWchar>> **GlyphRanges**

Private Functions

bool **handleKeyboardEvent** (EventDispatcher **sender*, **const** Event &*event*)

Handle keyboard event from engine

bool **handleInputEvent** (EventDispatcher **sender*, **const** Event &*event*)

Handle input event from engine

bool **doCapture** ()

Check if mouse event can be captured by any rocket element

bool **render** (EventDispatcher *dispatcher, const Event &e)

This function does not handle actual render system rendering, it only updates ImGui draw list

Private Members

ImGuiRenderer *mRenderer

ImFontAtlas *mFontAtlas

bool mIsSetUp

RendererFactories mRendererFactories

std::string mPendingSystemType

std::string mUsedRendererType

ImVector<ImFont *> mFonts

GlyphRanges mGlyphRanges

std::map<std::string, ImGuiContext *> mContexts

ImGuiDockspaceRenderer *mCurrentDockspace

Friends

friend **Gsage::ImGuiDockspaceView**

class ImGuiRenderer

Subclassed by *Gsage::ImGuiOgreRenderer*

Public Functions

ImGuiRenderer ()

virtual ~ImGuiRenderer ()

virtual void initialize (Engine *facade, lua_State *L)

Set engine, setup lua bindings

Parameters

- engine: Gsage CE
- L: lua state

void setMousePosition (const std::string &name, ImVec2 position)

Update mouse position for the render target

Parameters

- name: render target name
- position: mouse position

virtual void createFontTexture (unsigned char *pixels, int width, int height) = 0

Create font texture in the underlying render system

Parameters

- `pixels`: Raw texture
- `width`: Texture width
- `height`: Texture height

virtual void setImGuiContext (ImGuiContext **ctx*) = 0
 ImGui context is not shared across plugins, so pass it to renderer

Parameters

- `ctx`: *Context* to use

void **render** ()

Protected Functions

virtual Context *getContext (ImGuiContext **context*)

virtual Context *initializeContext (const std::string &*name*)

Protected Attributes

std::map<std::string, ImVec2> **mMousePositions**

Engine ***mEngine**

std::map<std::string, bool> **mRenderTargetWhitelist**

std::mutex **mContextLock**

std::map<std::string, Context> **mContexts**

std::map<ImGuiContext *, std::string> **mContextNames**

ImGuiManager ***mManager**

Friends

friend Gsage::ImGuiManager

struct Context

Public Members

ImVec2 **size**

ImGuiContext ***context**

class ImGuiViewCollection

Subclassed by *Gsage::ImGuiDockspaceView*, *Gsage::ImGuiManager*

Public Functions

virtual bool addView (const std::string &*name*, *sol::object* *view*)
 Add new lua view to the imgui renderer.

Parameters

- `name`: view name

- `view`: lua function that will render everything.

```
virtual bool removeView (const std::string &name, sol::object view)
    Remove view from collection
```

Protected Types

```
typedef std::function<sol::function_result ()> RenderView
typedef std::map<std::string, RenderView> Views
```

Protected Attributes

```
Views mViews
```

1.24.25 File `ImGuiMovableObject.h`

```
namespace Gsage
```

```
class ImGuiMovableObject : public MovableObject
```

Public Functions

```
ImGuiMovableObject (Ogre::IdType id, Ogre::ObjectMemoryManager *objectMemoryManager, Ogre::SceneManager *manager, Ogre::uint8 renderQueue)
```

```
virtual ~ImGuiMovableObject ()
```

```
virtual const Ogre::String &getMovableType (void) const
```

```
void updateVertexData (const ImDrawList *drawList, int offset)
    Update vertex data for each underlying renderable
```

Parameters

- `drawList`: ImDrawList
- `offset`: drawList list offset (z order)

```
void setDatablock (const Ogre::String &name)
    Set datablock to use for renderables
```

Parameters

- `name`: Datablock name

Private Members

```
Ogre::String mDatablockName
```

```
class ImGuiMovableObjectFactory : public MovableObjectFactory
    #include <ImGuiMovableObject.h> Factory object for creating ImGuiMovableObject instances
```

Public Functions

```
ImGuiMovableObjectFactory ()  
~ImGuiMovableObjectFactory ()  
  
const Ogre::String &getType (void) const  
  
void destroyInstance (Ogre::MovableObject *obj)
```

Public Static Attributes

```
Ogre::String FACTORY_TYPE_NAME
```

Protected Functions

```
virtual Ogre::MovableObject *createInstanceImpl (Ogre::IdType id,  
                                                Ogre::ObjectMemoryManager  
                                                *objectMemoryManager,  
                                                Ogre::SceneManager *manager,  
                                                const Ogre::NameValuePairList  
                                                *params = 0)
```

1.24.26 File ImGuiOgrePlugin.h

```
namespace Gsage
```

```
class ImGuiOgrePlugin : public IPlugin
```

Public Functions

```
ImGuiOgrePlugin ()  
  
virtual ~ImGuiOgrePlugin ()  
  
virtual const std::string &getName () const  
    Get rocket UI plugin name  
    Return ImGui string  
  
virtual bool installImpl ()  
    Install rocker ui manager  
    Return true if succesful  
  
virtual void uninstallImpl ()  
    Uninstall rocket ui manager  
  
virtual void setupLuaBindings ()  
    Set up lua bindings for imgui
```

1.24.27 File ImGuiOgreRenderer.h

namespace Gsage

class **ImGuiOgreRenderer** : **public** EventSubscriber<ImGuiOgreRenderer>, **public** Gsage::ImGuiRenderer
 Subclassed by *Gsage::ImGuiRendererV1*, *Gsage::ImGuiRendererV2*

Public Functions

ImGuiOgreRenderer ()

virtual **~ImGuiOgreRenderer** ()

virtual void **initialize** (Engine **facade*, lua_State **L*)
 Set engine, setup lua bindings

Parameters

- engine: Gsage CE
- L: lua state

virtual void **createFontTexture** (unsigned char **pixels*, int *width*, int *height*)
 Create font texture in the underlying render system

Parameters

- pixels: Raw texture
- width: Texture width
- height: Texture height

virtual void **updateVertexData** (Ogre::Viewport **vp*, ImVec2 *displaySize*) = 0

virtual void **setImGuiContext** (ImGuiContext **ctx*)
 ImGui context is not shared across plugins, so pass it to renderer

Parameters

- ctx: Context to use

Protected Functions

virtual void **updateFontTexture** ()

virtual void **createMaterial** () = 0

virtual bool **renderQueueEnded** (EventDispatcher **sender*, **const** Event &*event*)

Protected Attributes

Ogre::TexturePtr **mFontTex**

Ogre::SceneManager ***mSceneMgr**

unsigned char ***mFontPixels**

int **mFontTexWidth**

int **mFontTexHeight**

```
bool mUpdateFontTex
std::mutex mFontTexLock
```

1.24.28 File ImGuiPlugin.h

```
namespace Gsage
```

```
class ImGuiPlugin : public IPlugin
```

Public Functions

```
ImGuiPlugin ()
virtual ~ImGuiPlugin ()
virtual const std::string &getName () const
    Get rocket UI plugin name
    Return ImGui string
virtual bool installImpl ()
    Install rocker ui manager
    Return true if succesful
virtual void uninstallImpl ()
    Uninstall rocket ui manager
virtual void setupLuaBindings ()
    Set up lua bindings for imgui
```

Private Members

```
int mUIManagerHandle
ImGuiManager mUIManager
```

1.24.29 File ImGuiRendererV1.h

```
namespace Gsage
```

```
class ImGuiRendererV1 : public Gsage::ImGuiOgreRenderer
```

Public Functions

```
ImGuiRendererV1 ()
virtual ~ImGuiRendererV1 ()
void initialize (Engine *facade, lua_State *L)
    Set engine, setup lua bindings
```

Parameters

- engine: Gsage CE
- L: lua state

Protected Functions

void **updateVertexData** (*Ogre::Viewport *vp*, *ImVec2 displaySize*)

void **createMaterial** ()

void **updateFontTexture** ()

void **setFiltering** (*Ogre::TextureFilterOptions mode*)

Private Members

std::vector<*ImGuiRenderable* *> **mRenderables**

Ogre::Pass ***mPass**

Ogre::TextureUnitState ***mTexUnit**

1.24.30 File ImGuiRendererV2.h

namespace Gsage

class **ImGuiRendererV2** : public *Gsage::ImGuiOgreRenderer*

Public Functions

ImGuiRendererV2 (*Ogre::uint8 renderQueueGroup*)

virtual ~**ImGuiRendererV2** ()

void **initialize** (*Engine *facade*, *lua_State *L*)
Set engine, setup lua bindings

Parameters

- engine: Gsage CE
- L: lua state

virtual void **updateVertexData** (*Ogre::Viewport *vp*, *ImVec2 displaySize*)

Protected Functions

virtual void **createMaterial** ()

Private Functions

ImGuiMovableObject ***createImGuiMovableObject** ()

Private Members

```
std::vector<ImGuiMovableObject*> mImGuiMovableObjects
Ogre::uint8 mRenderQueueGroup
ImGuiMovableObjectFactory *mMovableObjectFactory
HlmsUnlit *mHlms
```

1.24.31 File ItemWrapper.h

namespace Ogre

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

namespace Gsage

```
class ItemWrapper : public Gsage::MovableObjectWrapper<Ogre::Item>
```

Public Types

enum Query

Values:

STATIC = 0x01

DYNAMIC = 0x02

UNKNOWN = 0x04

Public Functions

ItemWrapper()

virtual ~ItemWrapper()

void **setMesh** (const std::string &mesh)

Set model mesh (this function creates entity)

Parameters

- mesh: Mesh file name

const std::string &**getMesh** () const

Get mesh file name

void **setQueryFlags** (const std::string &type)

Set model entity flags, used for raycasting and querying entities

Parameters

- `type`: static means that entity is a part of location, dynamic means that entity is some actor

```
const std::string &getQueryFlags () const
    Get query flags of the model
```

```
Ogre::Item *getItem ()
    Get underlying entity
```

```
void setDatablock (const std::string &name)
    Set item datablock
```

Parameters

- `name`: datablock name (Hlms)

```
const std::string &getDatablock () const
    Get item datablock name
```

Public Static Attributes

```
const std::string TYPE
```

Private Members

```
std::string mMeshName
```

```
std::string mDatablock
```

```
std::string mQueryString
```

```
Query mQuery
```

1.24.32 File LightWrapper.h

```
namespace Gsage
```

```
class LightWrapper : public Gsage::MovableObjectWrapper<Ogre::Light>
```

Public Functions

```
LightWrapper ()
```

```
virtual ~LightWrapper ()
```

```
void create (const std::string &name)
    Create light instance
```

Parameters

- `light`: name

```
const std::string &getName () const
    Get light name
```

```
void setType (const std::string &type)
    Set light type
```

Parameters

- `type`: point/directional/spot

```

std::string getType ()
    Get light type

void setPosition (const Ogre::Vector3 &position)
    Set light position
Parameters
    • position: Ogre::Vector3 position

Ogre::Vector3 getPosition ()
    Get light position

void setDiffuseColour (const Ogre::ColourValue &value)
    Set light diffuse colour
Parameters
    • value: ColourValue

const Ogre::ColourValue &getDiffuseColour () const
    Get light diffuse colour

void setSpecularColour (const Ogre::ColourValue &value)
    Set light specular colour
Parameters
    • value: ColourValue

const Ogre::ColourValue &getSpecularColour () const
    Get light specular colour

void setDirection (const Ogre::Vector3 &value)
    Set light direction
Parameters
    • direction: Vector3 direction, non relevant for point light

Ogre::Vector3 getDirection ()
    Get light direction

void setCastShadows (const bool &value)
    Cast shadows from this light
Parameters
    • value: Cast shadows

bool getCastShadows ()
    Get cast shadows

void setRenderQueue (const unsigned int &queue)
    Set render queue
Parameters
    • queue: queue id

unsigned int getRenderQueue ()
    Get render queue

```

Public Static Attributes

```

const std::string TYPE

```

Private Functions

Ogre::Light::LightTypes **mapType** (**const** std::string &*type*)

Get ogre internal light type from string

Parameters

- *type*: point/directional/spotlight

std::string **mapType** (**const** *Ogre::Light::LightTypes* &*type*)

Convert *Ogre* light type to string type

Parameters

- *ogre*: *Ogre::Light::LightTypes*

1.24.33 File ManualMovableTextRenderer.h

namespace *Ogre*

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

class *ManualMovableTextRenderer* : **public** ParticleSystemRenderer

#include <*ManualMovableTextRenderer.h*> *Renderer* that is used to create floating text particles, like damage

Public Functions

ManualMovableTextRenderer (**const** std::string &*name*, ObjectMemoryManager **memoryManager*, SceneManager **sceneManager*)

virtual ~*ManualMovableTextRenderer* ()

void **setFontName** (**const** std::string &*name*)

Set font name to use

Parameters

- *name*: Id of the font

const std::string &**getFontName** () **const**

Get used font name

const std::string &**getType** () **const**

Get the type of this renderer

void **_updateRenderQueue** (RenderQueue **queue*, Camera **camera*, **const** Camera **lodCamera*, list<Particle *> &*currentParticles*, bool *cullIndividually*, RenderableArray &*outRenderables*)

Updates all nodes in the renderer

void **_notifyParticleQuota** (size_t *quota*)

Parameters

- `quota`: Number of nodes to be allocated

void **_notifyDefaultDimensions** (Real *width*, Real *height*)
Not used

void **setRenderQueueGroup** (uint8 *queueID*)
Not used

void **setRenderQueueGroupAndPriority** (uint8 *queueID*, ushort *priority*)
Not used

void **setKeepParticlesInLocalSpace** (bool *keepLocal*)
Not used

SortMode **_getSortMode** () **const**
Not used

void **_setDatablock** (HlmsDatablock **datablock*)
Not used

void **_setMaterialName** (**const** String &*matName*, **const** String &*resourceGroup*)
Not used

void **_notifyCurrentCamera** (**const** Camera **cam*)
Not used

void **_notifyAttached** (Node **parent*)
Not used

void **setRenderQueueSubGroup** (uint8 *queueID*)
Not used

void **_destroyVisualData** (ParticleVisualData **data*)
Not used

Protected Static Attributes

CmdFontName **msFontNameCmd**

Private Types

typedef std::vector<*TextNode*> **TextNodes**

typedef std::deque<*TextNode* *> **TextNodesPtrs**

typedef std::map<Particle *, *TextNode* *> **TextNodesMap**

Private Functions

void **adjustNodeCount** ()
Adjust particle count to the quota

Private Members

```

TextNodesMap mBusyNodes
TextNodesPtrs mFreeNodes
TextNodes mNodePool
size_t mQuota
std::string mName
std::string mFontName
int mPreviousParticleCount

class CmdFontName : public ParamCommand

```

Public Functions

```

std::string doGet (const void *target) const

void doSet (void *target, const std::string &val)

class ManualMovableTextRendererFactory : public ParticleSystemRendererFactory

```

Public Functions

```

ManualMovableTextRendererFactory ()

virtual ~ManualMovableTextRendererFactory ()

const String &getType () const

ParticleSystemRenderer *createInstance (const String &name)

void destroyInstance (ParticleSystemRenderer *ptr)

```

Private Members

```

int mCreatedRenderersCounter

class MovableTextValue : public ParticleVisualData
    #include <ManualMovableTextRenderer.h> Movable text string

```

Public Functions

```

MovableTextValue (const std::string &value, SceneNode *attachTo)

const std::string &getValue () const
    Get text value for particle

void setNode (TextNode *node)
    Set node that is using this movable text value
Parameters
    • node: Pointer to the node

```

TextNode *getNode ()

Node that is using this movable text value

Return 0 if no node present

SceneNode *getNodeToAttachTo ()

Get parent node to use for the text node

Private Members

std::string mValue

TextNode *mNode

SceneNode *mSceneNode

class TextNode

#include <ManualMovableTextRenderer.h> Class that represents created movable text node

Public Functions

TextNode (IdType *id*, ObjectMemoryManager *memoryManager, SceneManager *sceneManager)

virtual ~TextNode ()

void **activate** (*MovableTextValue* *value, **const** std::string &fontName)

Activate text node

Parameters

- parent: Node to attach text to
- value: Text value to display

void **deactivate** ()

Deactivate text node this deletes movable text value that was attached to this node

void **setPosition** (**const** Vector3 &position)

Set node position

Parameters

- position: Position

void **setColour** (**const** ColourValue &colour)

Set text colour

void **setHeight** (**const** float value)

Set text height

Private Members

MovableTextValue *mValue

SceneNode *mSceneNode

MovableText *mView

IdType mId

ObjectMemoryManager *mObjectManager

SceneManager *mSceneManager

1.24.34 File ManualTextureManager.h

namespace Gsage

class ManualTextureManager : public Listener

#include <ManualTextureManager.h> This class is used to handle manual texture management and update

Public Functions

ManualTextureManager (*OgreRenderSystem *renderSystem*)

virtual ~ManualTextureManager ()

void **reset** ()

Destroy all texture objects

TexturePtr **createTexture** (RenderSystem::TextureHandle *handle*, DataProxy *params*)

Create manual texture

List of possible texture parameters:

- **group** resource group to use (optional). Defaults to DEFAULT_RESOURCE_GROUP_NAME.
- **textureType** texture type (optional). Defaults to TEX_TYPE_2D.
- **width** initial texture width (required).
- **height** initial texture height (required).
- **depth** The depth of the texture (optional). Defaults to 0.
- **numMipmaps** The number of pre-filtered mipmaps to generate. If left to MIP_DEFAULT then the TextureManager's default number of mipmaps will be used (see setDefaultNumMipmaps()). If set to MIP_UNLIMITED mipmaps will be generated until the lowest possible level, 1x1x1.
- **pixelFormat** texture pixel format (optional). Defaults to PF_R8G8B8A8.
- **usage** usage type (optional). Defaults to TU_DEFAULT.
- **hwGammaCorrection** use gamma correction (optional). Defaults to false.
- **fsaa** antialiasing (optional). Defaults to 0.
- **fsaaHint** **The level of multisampling to use if this is a render target.** Ignored if usage does not include TU_RENDERTARGET or if the device does not support it. (optional).
- **explicitResolve** Whether FSAA resolves are done implicitly when used as texture, or must be done explicitly. (optional).
- **shareableDepthBuffer** Only valid for depth texture formats. When true, the depth buffer is a "view" of an existing depth texture (e.g. useful for reading the depth buffer contents of a GBuffer pass in deferred rendering). When false, the texture gets its own depth buffer created for itself (e.g. useful for shadow mapping, which is a depth-only pass).

Parameters

- **handle**: texture handle
- **params**: Texture params, see above

TexturePtr **getTexture** (RenderSystem::TextureHandle *handle*)

Gets existing texture

Parameters

- **handle**: texture handle

bool **deleteTexture** (RenderSystem::TextureHandle *handle*)

Delete texture by handle

Return true if succeed

Parameters

- `handle`: texture id

void **setDefaultPixelFormat** (*Ogre::PixelFormat* *format*)

Set default pixelFormat

Parameters

- `format`: *Ogre::PixelFormat*

void **updateDirtyTexures** ()

Update dirty textures

Private Types

typedef `std::unordered_map<Ogre::String, int>` **RenderSystemCapabilities**

Private Members

`std::map<RenderSystem::TextureHandle, TexturePtr>` **mTextures**

Ogre::PixelFormat **mPixelFormat**

OgreRenderSystem ***mRenderSystem**

RenderSystemCapabilities **mRenderSystemCapabilities**

class **OgreTexture** : **public** *Texture*, **public** *Listener*

#include <ManualTextureManager.h> Implements abstract texture class *Texture*

Public Types

enum **Flags**

Values:

BlitDirty = 1

Public Functions

OgreTexture (**const** `std::string` &*name*, **const** *DataProxy* &*params*, *Ogre::PixelFormat* *pixelFormat*, *int* *flags* = 0)

virtual **~OgreTexture** ()

void **update** (**const** `void *`*buffer*, `size_t` *size*, *int* *width*, *int* *height*)

Update texture data

Parameters

- `buffer`: buffer to use
- `size`: provided buffer size
- `width`: buffer width
- `height`: buffer height

virtual void **update** (**const** `void *`*buffer*, `size_t` *size*, *int* *width*, *int* *height*, **const** *Rect<int>* &*area*)

Update texture data using changed rectangle

Parameters

- `buffer`: buffer to use
- `size`: provided buffer size
- `width`: buffer width
- `height`: buffer height
- `area`: changed rect

bool **hasData** () **const**

Check if the texture has actual data

void **setSize** (int *width*, int *height*)

Set texture size

Parameters

- `width`: texture width
- `height`: texture height

void **unloadingComplete** (*Ogre::Resource *res*)

Ogre::Texture::Listener implementation

void **create** (int *width* = -1, int *height* = -1)

Create the underlying texture

Parameters

- `width`: override width
- `height`: override height

bool **isDirty** () **const**

Texture buffer was changed

void **render** ()

Update texture using supplied buffer

Ogre::TexturePtr **getOgreTexture** ()

Get underlying *OgreTexture* object

void **destroy** ()

Destroy underlying *Ogre::TexturePtr*

Private Functions

std::unique_ptr<*OgreTexture::ScalingPolicy*> **createScalingPolicy** (**const** DataProxy
&*params*)

bool **blitDirty** ()

bool **blitAll** ()

Private Members

Ogre::TexturePtr **mTexture**

std::string **mName**

std::unique_ptr<*OgreTexture::ScalingPolicy*> **mScalingPolicy**

bool **mHasData**

bool **mDirty**

```
bool mCreate
int mFlags
std::vector<Rect<int>> mDirtyRegions
```

Friends

```
friend Gsage::ScalingPolicy
class AllocateScalingPolicy: public Gsage::OgreTexture::ScalingPolicy
```

Public Functions

```
AllocateScalingPolicy(OgreTexture &texture, float scalingFactor)
```

Protected Functions

```
bool resize()
    Actual resize
```

Private Members

```
float mScalingFactor
class DefaultScalingPolicy: public Gsage::OgreTexture::ScalingPolicy
```

Public Functions

```
DefaultScalingPolicy(OgreTexture &texture)
```

Protected Functions

```
bool resize()
    Actual resize
```

```
class ScalingPolicy
    Subclassed by Gsage::OgreTexture::AllocateScalingPolicy, Gsage::OgreTexture::DefaultScalingPolicy
```

Public Functions

```
ScalingPolicy(OgreTexture &texture)
virtual ~ScalingPolicy()
void invalidate()
    Forces rescaling without width/height change
void update(int width, int height)
    Update scaling policy
```

bool **render** ()
 Should be called in the render loop
Return true if the texture was recreated

Protected Functions

virtual bool **resize** () = 0
 Actual resize

Protected Attributes

OgreTexture &**mTexture**
 int **mWidth**
 int **mHeight**
 bool **mDirty**

1.24.35 File MaterialBuilder.h

namespace Gsage

class MaterialBuilder

Public Functions

MaterialBuilder()
 virtual ~MaterialBuilder()

Public Static Functions

static *Ogre::MaterialPtr* **parse** (const std::string &*name*, const DataProxy &*data*)
 Create material from json data
Return true if succeed
Parameters
 • *data*: to create from

1.24.36 File MaterialLoader.h

namespace Gsage

```
class MaterialLoader : public EventDispatcher, public EventSubscriber<MaterialLoader>
#include <MaterialLoader.h> TODO: INDEXER
{ "filename": { "checksum": "...", "changed": "...", "materials": [] } }
for(auto& file : filesystem->ls(folder)) { if(filesystem->extension(file) != "material") { continue; } } Custom
OGRE material loader
```

Public Functions

MaterialLoader (*OgreRenderSystem* *render, GsageFacade *facade)

bool **load** (const std::string &material, const std::string &group, bool background = true)
Loads material from any of resource folders

Parameters

- material: Material name
- group: Material group
- background: Loads material in background

void **scan** (const std::string &folder)
Indexer scans materials files to detect materials sets defined there

Parameters

- folder: Folder to scan

Private Types

typedef std::map<std::string, *FileInfo*> **MaterialIndex**

Private Functions

void **scanFolder** (const std::string &folder, std::vector<std::string> &files, const std::string
extension)

void **readIndexFile** ()

void **writeIndexFile** ()

bool **onEnvUpdated** (EventDispatcher *sender, const Event &event)

void **reloadIndex** ()

Private Members

OgreRenderSystem ***mRender**

GsageFacade ***mFacade**

DataProxy **mIndex**

std::string **mWorkdir**

MaterialIndex **mMaterialIndex**

struct FileInfo

Public Members

std::string **path**

signed long **modified**

std::string **folder**

1.24.37 File MeshTools.h

namespace Ogre

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

class MeshInformation

#include <MeshTools.h> A single mesh information

Public Types

enum Flags

Values:

Vertices = 0x01

Indices = 1 << 1

Normals = 1 << 2

Default = *Vertices* | *Indices*

All = *Vertices* | *Indices* | *Normals*

Public Functions

MeshInformation()

~MeshInformation()

void **allocate** (size_t *vertexCount*, size_t *indexCount*, *MeshInformation::Flags* *flags*)
Allocates *MeshInformation* data arrays

Public Members

Ogre::Vector3 ***vertices**

Ogre::Vector3 ***normals**

Ogre::uint32 ***indices**

size_t **vertexCount**

size_t **indexCount**

class MeshTools : public *Ogre::Singleton<MeshTools>*

#include <MeshTools.h> Provides cached access to raw mesh data

Public Functions

MeshTools ()

virtual ~MeshTools ()

bool **getMeshInformation** (*Ogre::MovableObject* *object, *MeshInformation* &dest, *Ogre::Matrix4* transform, *MeshInformation::Flags* flags = *MeshInformation::Default*)

Gets movable object mesh information, if eligible

Return false if not eligible to get this information

bool **getMeshInformation** (*Ogre::MovableObject* *object, *MeshInformation* &dest, const *Ogre::Vector3* &position, const *Ogre::Quaternion* &orient, const *Ogre::Vector3* &scale, *MeshInformation::Flags* flags = *MeshInformation::Default*)

Gets movable object mesh information, if eligible

Return false if not eligible to get this information

void **getMeshInformation** (*OgreV1::MeshPtr* mesh, *MeshInformation* &dest, const *Ogre::Matrix4* &transform, *MeshInformation::Flags* flags = *MeshInformation::Default*)

Gets v1 mesh information

Public Static Functions

static *MeshTools* &getSingleton (void)

static *MeshTools* *getSingletonPtr (void)

1.24.38 File MovableObjectWrapper.h

namespace Gsage

class IMovableObjectWrapper : public *Gsage::OgreObject*

#include <MovableObjectWrapper.h> Provides templateless base class for *MovableObjectWrapper*

Subclassed by *Gsage::MovableObjectWrapper*< *T* >, *Gsage::MovableObjectWrapper*< *Ogre::Camera* >, *Gsage::MovableObjectWrapper*< *Ogre::Item* >, *Gsage::MovableObjectWrapper*< *Ogre::Light* >, *Gsage::MovableObjectWrapper*< *Ogre::ManualObject* >, *Gsage::MovableObjectWrapper*< *OgreV1::BillboardSet* >, *Gsage::MovableObjectWrapper*< *OgreV1::Entity* >

Public Functions

virtual void setRenderQueueGroup (const unsigned char &queueId) = 0

virtual unsigned char getRenderQueueGroup () = 0

virtual void setVisibilityFlags (unsigned int mask) = 0

virtual void resetVisibilityFlags () = 0

template<typename T>

class MovableObjectWrapper : public *Gsage::IMovableObjectWrapper*

Public Functions

```
MovableObjectWrapper ()  
  
virtual ~MovableObjectWrapper ()  
  
void defineUserBindings ()  
  
void setRenderQueueGroup (const unsigned char &queueId)  
  
unsigned char getRenderQueueGroup ()  
  
void setVisibilityFlags (unsigned int mask)  
  
void resetVisibilityFlags ()
```

Protected Attributes

```
T *mObject
```

1.24.39 File OSXUtils.h

```
namespace Gsage
```

Functions

```
unsigned long WindowContentViewHandle (SDL_SysWMInfo &info)
```

1.24.40 File ObjectMutation.h

```
namespace Gsage
```

```
class ObjectMutation
```

Public Types

```
typedef std::function<void ()> Callback
```

Public Functions

```
ObjectMutation ()  
  
ObjectMutation (Callback cb)  
  
virtual ~ObjectMutation ()  
  
void execute ()  
    Execute obejct mutation
```

Private Members

Callback `mCallback`

1.24.41 File OgreConverters.h

namespace Gsage

Functions

`static const Ogre::Vector3 GsageVector3ToOgreVector3 (const Gsage::Vector3 &vector)`

`static const Gsage::Vector3 OgreVector3ToGsageVector3 (const Ogre::Vector3 &vector)`

`static const Ogre::Quaternion GsageQuaternionToOgreQuaternion (const Gsage::Quaternion &quaternion)`

`static const Gsage::Quaternion OgreQuaternionToGsageQuaternion (const Ogre::Quaternion &quaternion)`

`static const Ogre::AxisAlignedBox BoundingBoxToAxisAlignedBox (const BoundingBox &bbox)`

`TYPE_CASTER (OgreDegreeCaster, Ogre::Degree, std::string)`

`TYPE_CASTER (OgreColourValueCaster, Ogre::ColourValue, std::string)`

`TYPE_CASTER (OgreVector3Caster, Ogre::Vector3, std::string)`

`TYPE_CASTER (OgreQuaternionCaster, Ogre::Quaternion, std::string)`

`TYPE_CASTER (OgreFloatRectCaster, Ogre::FloatRect, std::string)`

`TYPE_CASTER (OgrePixelFormatCaster, Ogre::PixelFormat, std::string)`

`TYPE_CASTER (RenderOperationTypeCaster, Ogre::RenderOperation::OperationType, std::string)`

`TYPE_CASTER (RenderTargetTypeCaster, RenderTargetType::Type, std::string)`

1.24.42 File OgreGeom.h

namespace Ogre

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

```
namespace Gsage
```

```
class OgreGeom : public Geom
```

Public Types

```
typedef std::vector<Ogre::MovableObject*> OgreEntities
```

Public Functions

```
OgreGeom(OgreEntities src, Ogre::SceneNode *referenceNode)
```

```
virtual ~OgreGeom()
```

Private Members

```
OgreEntities mSrcEntities
```

1.24.43 File OgreObjectManager.h

```
namespace Ogre
```

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

```
namespace Gsage
```

```
class OgreObjectManager : public EventDispatcher
```

Public Functions

```
OgreObjectManager(OgreRenderSystem *rs)
```

```
virtual ~OgreObjectManager()
```

```
OgreObject *create(const DataProxy &dict, const std::string &owner, Ogre::SceneManager  
*sceneManager, Ogre::SceneNode *parent = 0)
```

Create object from the DataProxy

Parameters

- dict: DataProxy with all values (dict should contain type field)
- owner: Owner entity of the created object
- sceneManager: Ogre::SceneManager to create object in
- parent: Parent object to attach to

OgreObject *create (const DataProxy &dict, const std::string &owner, *Ogre::SceneManager* *sceneManager, const std::string &type, *Ogre::SceneNode* *parent = 0)

Create object from the DataProxy

Parameters

- dict: DataProxy with all values
- owner: Owner entity of the created object
- sceneManager: *Ogre::SceneManager* to create object in
- type: Object type string, defined explicitly
- parent: Parent object to attach to

OgreObject *create (const DataProxy &dict, const std::string &owner, *Ogre::SceneManager* *sceneManager, const std::string &type, const DataProxy ¶ms, *OgreObject* *parent = 0)

Create object from the DataProxy

Parameters

- dict: DataProxy with all values
- owner: Owner entity of the created object
- sceneManager: *Ogre::SceneManager* to create object in
- type: Object type string, defined explicitly
- parent: Parent object to attach to

template<typename C>

C *create (const DataProxy &dict, const std::string &owner, *Ogre::SceneManager* *sceneManager, *Ogre::SceneNode* *parent = 0)

See *OgreObjectManager::create*

void destroy (*OgreObject* *object)

Destroy object

Parameters

- object: Object to destroy

template<typename C>

void registerElement (const std::string &type)

Register new element type, so factory will be able to create it

Parameters

- type: String type representation

template<typename C>

void registerElement ()

Register new element type, so factory will be able to create id. For this method, visual element should have static std::string TYPE defined

bool unregisterElement (const std::string &type)

Unregister existing element type It will destroy all existing objects of that type

Parameters

- type: String type representation

template<typename C>

bool unregisterElement ()

Unregister existing element type It will destroy all existing objects of that type

OgreRenderSystem *getRenderSystem ()

Get targeted render system instance

Private Types

```
typedef std::map<std::string, OgreObjectPool *> OgreObjectsCollections
```

Private Members

```
OgreObjectsCollections mObjects
```

```
OgreRenderSystem *mRenderSystem
```

```
template<typename C>
```

```
class ConcreteOgreObjectPool : public Gsage::OgreObjectManager::OgreObjectPool
```

Public Functions

```
virtual ~ConcreteOgreObjectPool ()
```

```
OgreObject *allocate ()
```

```
void remove (OgreObject *object)
```

```
void clear ()
```

Private Members

```
ObjectPool<C> mObjects
```

```
class OgreObjectPool
```

Public Functions

```
virtual ~OgreObjectPool ()
```

```
virtual OgreObject *allocate () = 0
```

```
virtual void remove (OgreObject *object) = 0
```

```
class OgreObjectManagerEvent : public Event
```

```
#include <OgreObjectManager.h> Event related to factory lifecycle
```

Public Functions

```
OgreObjectManagerEvent (Event::ConstType type, const std::string &id, OgreObject *object = 0)
```

```
virtual ~OgreObjectManagerEvent ()
```

```
const std::string &getId () const
```

```
Get factory id that was unregistered or object id related to event
```

```
const OgreObject *getObject () const
```

```
Related object
```

Public Static Attributes

const Event::Type **FACTORY_UNREGISTERED**
Factory was unregistered

const Event::Type **OBJECT_DESTROYED**
Object was destroyed

Private Members

std::string **mId**

OgreObject ***mObject**

1.24.44 File OgreObject.h

namespace Ogre

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

namespace Gsage

class **OgreObject** : **public** Serializable<*OgreObject*>

#include <*OgreObject.h*> Abstract ogre object

Subclassed by *Gsage::IMovableObjectWrapper*, *Gsage::ParticleSystemWrapper*,
Gsage::SceneNodeWrapper

Public Functions

OgreObject ()

virtual ~**OgreObject** ()

virtual **bool** **initialize** (*OgreObjectManager* **objectManager*, **const** DataProxy
&*dict*, **const** std::string &*ownerId*, **const** std::string &*type*,
Ogre::SceneManager **sceneManager*, *Ogre::SceneNode* **parent*)

Initialize element from the node with values

Parameters

- *factory*: *OgreObjectManager* to enable child class creation
- *dict*: DataProxy with values
- *type*: String type of the object
- *parent*: Parent SceneNode
- *sceneManager*: SceneManager to use
- *ownerId*: Id of entity that owns the element

```
virtual bool initialize (OgreObjectManager *objectManager, const DataProxy
                        &dict, const std::string &ownerId, const std::string &type,
                        Ogre::SceneManager *sceneManager, const DataProxy &attach-
                        Params, OgreObject *parent)
```

Initialize element from the node with values

Parameters

- *factory*: *OgreObjectManager* to enable child class creation
- *dict*: *DataProxy* with values
- *type*: String type of the object
- *parent*: Parent *OgreObject*
- *sceneManager*: *SceneManager* to use
- *attachParams*: Passed down to the parent object when attaching
- *ownerId*: Id of entity that owns the element

```
virtual void destroy ()
```

Destroy object

```
const std::string &getType () const
```

Get object type

```
const std::string &getObjectId () const
```

Get object id

```
void attachObject (Ogre::MovableObject *object)
```

Attach object to the parent node

Parameters

- *object*: Object to attach

```
virtual bool attach (Ogre::MovableObject *object, const DataProxy &params)
```

Attach movable object to this object

Parameters

- *object*: Object to attach

```
std::string generateName () const
```

Generate unique name for the object.

```
void sync ()
```

Sync all pending property updates

Protected Attributes

```
std::string mObjectId
```

```
std::string mOwnerId
```

```
std::string mType
```

```
OgreObjectManager *mObjectManager
```

```
Ogre::SceneManager *mSceneManager
```

```
Ogre::SceneNode *mParentNode
```

```
OgreObject *mParentObject
```

```
DataProxy mAttachParams
```

```
struct PendingPropertyUpdate
```

Public Members

std::string **propertyName**

1.24.45 File OgreRenderComponent.h

namespace Ogre

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

namespace Gsage

```
class OgreRenderComponent : public EventDispatcher, public RenderComponent
    #include <OgreRenderComponent.h> Ogre render system component
```

Public Types

```
enum RotationAxis
```

Values:

X_AXIS

Y_AXIS

Z_AXIS

NONE

Public Functions

```
OgreRenderComponent ( )
```

```
virtual ~OgreRenderComponent ( )
```

```
void prepare (Ogre::SceneManager *sceneManager, ResourceManager *resourceManager, Ogre-  
              ObjectManager *objectManager)  
    Set scene manager instance
```

Parameters

- sceneManager: *Ogre::SceneManager* instance to allow creating objects on scene
- resourceManager: *ResourceManager** instance to allow component loading additional resource
- objectManager: *OgreObjectManager* instance to allow creating root tree

```
void setPosition (const Ogre::Vector3 &position)
```

Set render component position

Parameters

- `position`: New position

void **setOrientation** (**const** *Ogre::Quaternion* &*orientation*)

Set render component orientation (equal to *Ogre::SceneNode::setOrientation*)

Parameters

- `orientation`: Orientation quaternion (absolute)

void **rotate** (**const** *Ogre::Quaternion* &*rotation*)

Rotates render component

Parameters

- `rotation`: Rotation quaternion (relative)

void **lookAt** (**const** *Ogre::Vector3* &*position*, **const** *Geometry::RotationAxis* *rotationAxis*, *Geometry::TransformSpace* *transformSpace* = *Geometry::TS_WORLD*)

Equalient to *Ogre::Node* lookAt

Parameters

- `position`: Position to look at
- `rotationAxis`: rotate only in one axis
- `transformSpace`: ogre transform space

void **lookAt** (**const** *Ogre::Vector3* &*position*)

Equalient to *Ogre::Node* lookAt

Parameters

- `position`: Position to look at

const *Ogre::Vector3* **getOgrePosition** ()

Get current position

const *Ogre::Vector3* **getOgreScale** ()

Get current scale

const *Ogre::Vector3* **getOgreDirection** ()

Get current direction vector, uses *orientationVector* from config to detect front of the object

const *Ogre::Quaternion* **getOgreOrientation** ()

Get object orientation

const *Ogre::Quaternion* **getOgreFaceOrientation** ()

Get object orientation with respect to the direction vector

void **rotate** (**const** *Gsage::Quaternion* &*rotation*)

Rotates render component using *Gsage::Quaternion*

Parameters

- `rotation`: Rotation quaternion (relative)

void **lookAt** (**const** *Gsage::Vector3* &*position*, **const** *Geometry::RotationAxis* *rotationAxis*, *Geometry::TransformSpace* *transformSpace* = *Geometry::TS_WORLD*)

Equalient to *Ogre::Node* lookAt

Parameters

- `position`: Position to look at
- `rotationAxis`: rotate only in one axis
- `transformSpace`: ogre transform space

void **lookAt** (**const** *Gsage::Vector3* &*position*)

Parameters

- `position`: Position to look at

void **setPosition** (const *Gsage*::Vector3 &position)
Set position by using Gsage::Vector
Parameters

- position: New position

void **setOrientation** (const *Gsage*::Quaternion &orientation)
Set orientation using Gsage::Quaternion
Parameters

- orientation: Orientation quaternion (absolute)

const *Gsage*::Vector3 **getPosition** ()
Get current position

const *Gsage*::Vector3 **getScale** ()
Get current scale

const *Gsage*::Vector3 **getDirection** ()
Get current direction vector, uses orientationVector from config to detect front of the object

const *Gsage*::Quaternion **getOrientation** ()
Get object orientation

const *Gsage*::Quaternion **getFaceOrientation** ()
Get object orientation with respect to the direction vector

bool **adjustAnimationStateSpeed** (const std::string &name, double speed)
Adjusts animation speed for state
Parameters

- name: *Animation* state name
- speed: animation speed

bool **setAnimationState** (const std::string &name)
Sets animation state
Return true if state was found
Parameters

- name: *Animation* name

bool **playAnimation** (const std::string &name, int times = -1, double speed = 1, double offset = 0, bool reset = false)
Plays animation
Return true if animation group exists
Parameters

- name: *Animation* group name
- times: Repeat times, -1 means loop
- speed: *Animation* speed, 1 is normal speed
- offset: *Animation* start offset
- reset: Starts animation immediately

void **resetAnimationState** ()
Resets animation state to default

DataProxy **getRootNode** ()
Reads component root node

void **setRootNode** (const DataProxy &value)
Builds node tree from DataProxy. Example:

```
{
  "position": "0,0,0",
  "scale": "0,0,0",
  "children": [
    {
      "type": "model",
      "mesh": "mesh.mesh"
    }
  ]
}
```

DataProxy **getAnimations** ()

Reads component animations

void **setAnimations** (const DataProxy &value)

Sets animations DataProxy should have the following format:

```
{
  "states": {
    "state1": {"base": {"<model-name>": "<anim-name>"}}
  }
}
```

SceneNodeWrapper ***getRoot** ()

Get root *SceneNodeWrapper*

void **setResources** (const DataProxy &resources)

Read additional resources paths from the DataProxy

Parameters

- resources: DataProxy with all resources settings

const DataProxy &**getResources** () const

Get resources

Public Static Attributes

const std::string **SYSTEM**

const Event::Type **POSITION_CHANGE**

Private Members

bool **mAddedToScene**

AnimationScheduler **mAnimationScheduler**

SceneNodeWrapper ***mRootNode**

DataProxy **mResources**

Ogre::SceneManager ***mSceneManager**

ResourceManager ***mResourceManager**

OgreObjectManager ***mObjectManager**

Friends

```
friend Gsage::OgreRenderSystem
```

1.24.46 File OgreRenderSystem.h

Variables

```
const std::string OGRE_SECTION = "OgreRenderSystem"
```

```
const std::string OGRE_PLUGINS_PATH = "PluginsPath"
```

```
const std::string OGRE_CONFIG_PATH = "ConfigPath"
```

```
const std::string OGRE_RESOURCES = "Resources"
```

```
namespace Ogre
```

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

```
namespace Gsage
```

```
class OgreLogRedirect : public LogListener
```

#include <OgreRenderSystem.h> Class, used to redirect all ogre output to the easylogging++

Private Functions

```
void messageLogged (const std::string &message, Ogre::LogMessageLevel lml, bool maskDebug, const std::string &system, bool &skipMessage)
```

```
class OgreRenderSystem : public ComponentStorage<OgreRenderComponent>, public RenderQueueListener, pub
```

Public Types

```
typedef std::vector<Entity *> Entities
```

```
typedef std::vector<Ogre::MovableObject *> OgreEntities  
    Enumerable of Ogre::MovableObject
```

Public Functions

```
OgreRenderSystem()
```

```
virtual ~OgreRenderSystem()
```

virtual bool initialize (**const** DataProxy &settings)
 Initializes ogre render system
Parameters

- settings: DataProxy with initial settings for the render system

virtual void shutdown ()
 Shutdown OGRE

virtual bool prepareComponent (*OgreRenderComponent* *c)
 Initializes *OgreRenderComponent*, injects objectManager, resourceManager and sceneManager
Parameters

- c: *OgreRenderComponent* component to initialize

virtual bool fillComponentData (*OgreRenderComponent* *component, **const** DataProxy &data)
 Finalizes setup
Return false if failed to initialize component for some reason
Parameters

- component: *OgreRenderComponent* component to initialize
- data: DataProxy with node settings

virtual void update (**const** double &time)
 Update render system
Parameters

- time: Elapsed time

virtual void updateComponent (*OgreRenderComponent* *component, Entity *entity, **const** double &time)
 Update *OgreRenderComponent*
Parameters

- component: *OgreRenderComponent* to update
- entity: Entity that owns that component
- time: Elapsed time

virtual bool removeComponent (*OgreRenderComponent* *component)
 Remove render component from the system. Removes all related visual nodes
Parameters

- component: *OgreRenderComponent* component to remove

virtual bool configure (**const** DataProxy &config)
 Reconfigure render system
Parameters

- config: DataProxy with configs

DataProxy &**getConfig** ()
 Get current configs of the system

GeomPtr **getGeometry** (**const** BoundingBox &bounds, int flags = 0xFF)
 Get implementation independent Geometry information
Return GeomPtr
Parameters

- bounds: get entities in bounds
- flags: filter entities by flags (default is all)

GeomPtr **getGeometry** (std::vector<std::string> *entities*)

Get implementation independent Geometry information

Return GeomPtr

Parameters

- *entities*: filter entities by names

virtual TexturePtr **createTexture** (RenderSystem::TextureHandle *handle*, **const** DataProxy &*parameters*)

Create texture manually

Parameters

- *handle*: texture id
- *parameters*: variable parameters that can be used for texture creation

virtual TexturePtr **getTexture** (RenderSystem::TextureHandle *handle*)

Get texture by name

Parameters

- *handle*: texture id

virtual bool **deleteTexture** (RenderSystem::TextureHandle *handle*)

Delete texture by handle

Return true if succeed

Parameters

- *handle*: texture id

virtual void **renderQueueStarted** (Ogre::uint8 *queueGroupId*, **const** Ogre::String &*invocation*, bool &*skipThisInvocation*)

Callback for ui updating 1.x, 2.0

virtual void **renderQueueEnded** (Ogre::uint8 *queueGroupId*, **const** Ogre::String &*invocation*, bool &*skipThisInvocation*)

Render queue event pass through

template<typename T>

void **registerElement** ()

Register new type of factory in the render system

template<typename T>

bool **unregisterElement** ()

Unregister factory type

Ogre::RenderWindow ***getRenderWindow** ()

Get ogre render window

Ogre::SceneManager ***getSceneManager** ()

Get ogre scene manager

Ogre::RenderSystem ***getRenderSystem** ()

Get ogre render system

OgreEntities **getEntities** (**const** unsigned int &*query*, **const** BoundingBox &*bounds*)

Gets all scene entities

Parameters

- *query*: Filter entities by some query, default is all
- *bounds*: filter entities which are intersected by bounds

OgreEntities **getEntities** (**const** unsigned int &query = 0xFF)

Gets all scene entities

Parameters

- query: Filter entities by some query, default is all

Entities **getObjectsInRadius** (**const** *Ogre::Vector3* ¢er, float distance, **const** unsigned int flags = 0xFF, **const** std::string &id = "")

Get objects in radius

Return list of entities

Parameters

- position: Point to search around
- distance: Radius of the sphere query

virtual void **setWidth** (unsigned int width, **const** std::string &target = "")

Set render target width

Parameters

- target: Render target name

virtual unsigned int **getWidth** (**const** std::string &target = "") **const**

Get render window width

virtual void **setHeight** (unsigned int height, **const** std::string &target = "")

Set render target height

Parameters

- target: Render target name

virtual unsigned int **getHeight** (**const** std::string &target = "") **const**

Get render window height

virtual void **setSize** (unsigned int width, unsigned int height, **const** std::string &target = "")

Set size of render target

Parameters

- target: Render target name

void **renderCameraToTarget** (**const** std::string &cameraName, **const** std::string &target)

Render camera to texture with name

Parameters

- cameraName: Name of the camera to use
- target: Name of RTT to render to

void **renderCameraToTarget** (*Ogre::Camera* *cam, **const** std::string &target)

Render camera to texture with name

Parameters

- cam: Camera pointer
- target: Name of RTT to render to

RenderTargetPtr **createRenderTarget** (**const** std::string &name, *RenderTargetType::Type* type, DataProxy parameters)

Create new render target

Parameters

- name: Render target name
- type: Render target type
- parameters: Additional parameters

RenderTargetPtr **getRenderTarget** (const std::string &name)

Get render target by name

Parameters

- name: Render target name

RenderTargetPtr **getRenderTarget** (Ogre::RenderTarget *target)

Get render target by wrapped Ogre::RenderTarget*

Parameters

- target: Ogre::RenderTarget

RenderTargetPtr **getMainRenderTarget** ()

Get main render target (window)

MaterialLoader ***getMaterialLoader** ()

Get material loader

bool **allowMultithreading** ()

Allow multithreaded mode for *OgreRenderSystem*

void **queueMutation** (ObjectMutation::Callback callback)

Queue object mutation is called from *Ogre* wrappers

Parameters

- callback: Mutation callback

OgreObjectManager ***getObjectManager** ()

Gets object manager

Public Static Attributes

const std::string ID

Protected Types

typedef std::queue<DataProxy> **ComponentLoadQueue**

typedef std::map<std::string, *RenderTargetPtr*> **RenderTargets**

typedef std::map<*Ogre::RenderTarget* *, std::string> **RenderTargetReverseIndex**

typedef std::map<std::string, *RenderTargetPtr*> **RenderWindowsByHandle**

Protected Functions

bool **handleWindowResized** (EventDispatcher *sender, const Event &event)

Handle window resizing

Parameters

- sender: Engine
- event: WindowEvent

bool **installPlugin** (const std::string &name)

GeomPtr **getGeometry** (*OgreEntities* entities)

void **removeAllRenderTargets** ()

Protected Attributes

```

Ogre::Root *mRoot
Ogre::SceneManager *mSceneManager
Ogre::RenderSystem *mRenderSystem
Ogre::LogManager *mLogManager
Ogre::FontManager *mFontManager
Ogre::Viewport *mViewport
Ogre::ManualMovableTextRendererFactory *mManualMovableTextParticleFactory
OgreLogRedirect mLogRedirect
OgreObjectManager mObjectManager
WindowEventListener *mWindowEventListener
ResourceManager *mResourceManager
MaterialLoader *mMaterialLoader
ComponentLoadQueue mLoadQueue
RenderTarget mRenderTargets
RenderTargetReverseIndex mRenderTargetsReverseIndex
RenderTargetFactory mRenderTargetFactory
RenderWindowsByHandle mRenderWindowsByHandle
RenderTargetPtr mWindow
ThreadSafeQueue<ObjectMutation> mMutationQueue
ManualTextureManager mManualTextureManager

```

1.24.47 File OgreSelectEvent.h

```
namespace Gsage
```

```
class OgreSelectEvent : public SelectEvent
```

Public Functions

```

OgreSelectEvent (Event::ConstType type, const unsigned int &flags, const std::string &entityId, const Ogre::Vector3 &intersection)

virtual ~OgreSelectEvent ()

const Ogre::Vector3 &getIntersection () const
    Get selection hit point

```

Private Members

```
Ogre::Vector3 mIntersection
```

1.24.48 File OgreView.h

namespace Gsage

```
class OgreView: public EventSubscriber<OgreView>
    #include <OgreView.h> Allows displaying ogre camera into imgui window.
```

Lua usage example:

```
-- create ogre viewport
viewport = imgui.createOgreView("#000000FF")

local textureID = "myOgreView"

-- render camera to texture
local cam = camera:create("free")
cam:renderToTexture(textureID, {
    autoUpdated = false
})

-- set ogre view texture
viewport:setTextureID(textureID)
-- update on each imgui render call
viewport:render(320, 240)
```

Public Functions

OgreView (*OgreRenderSystem* *render, ImVec4 bgColour)

virtual ~OgreView ()

void **render** (unsigned int *width*, unsigned int *height*)
Render ogre view

void **setTexture** (TexturePtr *texture*)
Sets rendered manual texture object
Parameters
• texture: Texture object

bool **setTexture** (const std::string &*id*)
Sets texture by texture id
Return true if succeed
Parameters
• id: texture id

Private Functions

void **setTextureID** (const std::string &*textureID*)
Sets rendered texture id
Parameters
• textureID: RTT texture

bool **onTextureEvent** (EventDispatcher *sender, const Event &event)

Private Members

```

std::string mTextureID
TexturePtr mTexture
OgreRenderSystem *mRender
unsigned int mWidth
unsigned int mHeight
ViewportRenderData *mViewport
ImVec2 mPosition
ImVec4 mBgColour

```

1.24.49 File OverlayPass.h

```
namespace Gsage
```

```
class OverlayPass : public Gsage::CustomPass
```

Public Functions

```
OverlayPass (Engine *engine, const Ogre::CompositorPassDef *def, const
             Ogre::CompositorChannel &target, Ogre::CompositorNode *parentNode)
```

```
virtual ~OverlayPass ()
```

```
void execute (const Ogre::Camera *lodCamera)
    Do the UI rendering
```

Parameters

- lodCamera: Unused

```
class OverlayPassDef : public Gsage::CustomPassDef
```

Public Functions

```
OverlayPassDef (Engine *engine, Ogre::CompositorTargetDef *target,
                Ogre::CompositorNodeDef *node)
```

```
virtual ~OverlayPassDef ()
```

1.24.50 File ParticleSystemWrapper.h

Defines

```
__NODE_ID_TYPE
```

```
namespace Gsage
```

```
class ParticleSystemWrapper : public Gsage::OgreObject
```

Public Functions

ParticleSystemWrapper()

virtual ~ParticleSystemWrapper()

bool **read** (const DataProxy &dict)
Override default behavior for read

void **setTemplate** (const std::string &templateName)
Set particle system template. Note that it requires particle system recreation
Parameters

- templateName: Template name

const std::string &**getTemplate** () const
Get particle system template name

void **createParticle** (unsigned short index, __NODE_ID_TYPE nodeId)
Manually create particle in the system
Parameters

- index: Index of the emitter
- nodeId: Emit particle from the specified node

void **createParticle** (unsigned short index, __NODE_ID_TYPE nodeId, const std::string &value)
Manually create text particle in the system Particle system type should be “manual_text” to use this method. Otherwise it will just generate a particle
Parameters

- index: Index of the emitter
- nodeId: Emit particle from the specified node
- value: Text value

void **createParticle** (unsigned short index, __NODE_ID_TYPE nodeId, const std::string &value, const Ogre::Quaternion &rotation)
Manually create text particle in the system Particle system type should be “manual_text” to use this method. Otherwise it will just generate a particle
Parameters

- index: Index of the emitter
- nodeId: Emit particle from the specified node
- value: Text value
- rotate: the emitter in the following direction

Public Static Attributes

const std::string **TYPE**

Private Functions

void **createParticle** (unsigned short index, Ogre::ParticleVisualData *data, __NODE_ID_TYPE nodeId, const Ogre::Quaternion &rotation)

Private Members

std::string **mTemplate**

Ogre::ParticleSystem ***mParticleSystem**

1.24.51 File RenderEvent.h

namespace Gsage

class **RenderEvent** : public Event

#include <RenderEvent.h> Event that is used to update any UI overlay system

Public Functions

RenderEvent (Event::ConstType type, *OgreRenderSystem* *renderSystem, *Ogre::uint8* queueID = 0, *RenderTargetPtr* renderTarget = nullptr)

virtual ~**RenderEvent** ()

OgreRenderSystem ***getRenderSystem** ()

Gets a reference to the current render system

Public Members

Ogre::uint8 **queueID**

Ogre queue id

RenderTargetPtr **renderTarget**

RenderTarget that triggered that render event

Public Static Attributes

const Event::Type **RENDER_QUEUE_STARTED**

const Event::Type **UPDATE**

Main update

const Event::Type **RENDER_QUEUE_ENDED**

Same to ogre render queue ended

Private Members

OgreRenderSystem ***mRenderSystem**

1.24.52 File RenderInterfaceOgre3D.h

class **RenderInterfaceOgre3D** : public RenderInterface

#include <RenderInterfaceOgre3D.h> A sample render interface for Rocket into Ogre3D.

Modified by Brett Didemus to work with programable pipeline

Author Peter Curry

Public Functions

RenderInterfaceOgre3D (unsigned int *window_width*, unsigned int *window_height*)

virtual ~RenderInterfaceOgre3D ()

virtual void RenderGeometry (Rocket::Core::Vertex **vertices*, int *num_vertices*, int **indices*, int *num_indices*, Rocket::Core::TextureHandle *texture*, **const** Rocket::Core::Vector2f &*translation*)

Called by Rocket when it wants to render geometry that it does not wish to optimise.

virtual Rocket::Core::CompiledGeometryHandle CompileGeometry (Rocket::Core::Vertex **vertices*, int *num_vertices*, int **indices*, int *num_indices*, Rocket::Core::TextureHandle *texture*)

Called by Rocket when it wants to compile geometry it believes will be static for the foreseeable future.

virtual void RenderCompiledGeometry (Rocket::Core::CompiledGeometryHandle *geometry*, **const** Rocket::Core::Vector2f &*translation*)

Called by Rocket when it wants to render application-compiled geometry.

virtual void ReleaseCompiledGeometry (Rocket::Core::CompiledGeometryHandle *geometry*)

Called by Rocket when it wants to release application-compiled geometry.

virtual void EnableScissorRegion (bool *enable*)

Called by Rocket when it wants to enable or disable scissoring to clip content.

virtual void SetScissorRegion (int *x*, int *y*, int *width*, int *height*)

Called by Rocket when it wants to change the scissor region.

virtual bool LoadTexture (Rocket::Core::TextureHandle &*texture_handle*, Rocket::Core::Vector2i &*texture_dimensions*, **const** Rocket::Core::String &*source*)

Called by Rocket when a texture is required by the library.

virtual bool GenerateTexture (Rocket::Core::TextureHandle &*texture_handle*, **const** Rocket::Core::byte **source*, **const** Rocket::Core::Vector2i &*source_dimensions*)

Called by Rocket when a texture is required to be built from an internally-generated sequence of pixels.

virtual void ReleaseTexture (Rocket::Core::TextureHandle *texture*)

Called by Rocket when a loaded texture is no longer required.

float **GetHorizontalTexelOffset** ()

Returns the native horizontal texel offset for the renderer.

float **GetVerticalTexelOffset** ()

Returns the native vertical texel offset for the renderer.

void **setCustomProjectionMatrix** (*Ogre::Matrix4 projMatrix*)

Private Members

Ogre::RenderSystem ***render_system**

Ogre::LayerBlendModeEx **colour_blend_mode**

Ogre::LayerBlendModeEx **alpha_blend_mode**

```

Ogre::Matrix4 customProjectionMatrix
Ogre::TexturePtr m_blankTexture
bool scissor_enable
int scissor_left
int scissor_top
int scissor_right
int scissor_bottom

```

1.24.53 File RenderTargetTypes.h

```
namespace Gsage
```

```
class RenderTargetType
```

Public Types

```
enum Type
```

Represents different kinds of underlying wrapped target

Values:

```
Generic = 0
```

```
Rtt = 1
```

```
Window = 2
```

1.24.54 File RenderTarget.h

```
namespace Ogre
```

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

```
namespace Gsage
```

Typedefs

```
typedef RenderTarget *RenderTargetPtr
```

```
class RenderTarget : public EventSubscriber<RenderTarget>
```

Subclassed by *Gsage::RttRenderTarget*, *Gsage::WindowRenderTarget*

Public Functions

RenderTarget (**const** std::string &name, *Ogre::RenderTarget *wrapped*, **const** DataProxy ¶meters, Engine *engine)

RenderTarget (**const** std::string &name, *RenderTargetType::Type* type, **const** DataProxy ¶meters, Engine *engine)

virtual ~RenderTarget ()

const std::string &**getName** () **const**
Get *RenderTarget* name

RenderTargetType::Type **getType** () **const**
Get *RenderTarget* type

void initialize (*Ogre::SceneManager *sceneManager*)
Initialize with scene manager

Parameters

- sceneManager: *Ogre::SceneManager*

virtual int getWidth () **const**
Get render target width

virtual int getHeight () **const**
Get render target height

virtual void setWidth (int width)
Set render target width

Parameters

- width:

virtual void setHeight (int height)
Set render target height

Parameters

- height:

virtual void setDimensions (int width, int height)
Set render target dimensions

Parameters

- width:
- height:

virtual void setPosition (int x, int y)
Set render target position

Parameters

- x:
- y:

virtual void setCamera (*Ogre::Camera *camera*)
Set render target camera

Parameters

- camera:

virtual *Ogre::Camera* *getCamera ()
Get camera

```

virtual Ogre::Viewport *getViewPort ()
    Get viewport

virtual void update ()
    Update render target

bool isAutoUpdated () const
    Is render target auto updated

Ogre::RenderTarget *getOgreRenderTarget ()
    Get underlying Ogre::RenderTarget

bool isMouseOver () const
    Check if mouse over

void switchToDefaultCamera ()
    Switches back to default camera

virtual bool onTextureEvent (EventDispatcher *sender, const Event &event)

std::tuple<Ogre::Vector3, Entity *> raycast (Ogre::Real defaultDistance, Ogre::Real closestDis-
                                         tance, unsigned int flags) const
    Raycast and get object and 3d point under the pointer

Return tuple: point and object hit by ray
Parameters
    • defaultDistance: If nothing is hit, use ray and point on it
    • closestDistance: Closest raycast distance
    • flags: Objects flags filter

```

Protected Functions

```

virtual void updateCameraAspectRatio ()

virtual void configureViewport (Ogre::Viewport *viewport)

```

Protected Attributes

```

Ogre::RenderTarget *mWrappedTarget
std::string renderQueueSequenceName
int mX
int mY
int mWidth
int mHeight
DataProxy mParameters
const std::string mName
Ogre::Camera *mCurrentCamera
Ogre::Camera *mDefaultCamera
bool mAutoUpdate
RenderTargetType::Type mType

```

```
bool mHasQueueSequence
bool mDestroying
int mSamples
Engine *mEngine
Ogre::SceneManager *mSceneManager
std::shared_ptr<MOC::CollisionTools> mCollisionTools
bool mContinuousRaycast
bool mMouseOver
Ogre::Vector2 mMousePosition
std::string mRolledOverObject
bool mRenderQueueSequenceCreated
```

Private Functions

```
void subscribe ()

bool handleRaycast (EventDispatcher *sender, const Event &event)

bool handleMouseEvent (EventDispatcher *sender, const Event &event)

std::tuple<Ogre::Ray, bool> getRay () const

void doRaycasting (float x, float y, unsigned int flags = 0xFFFF, bool select = false)
```

class RenderTargetFactory

#include <RenderTarget.h> Factory creates Ogre::RenderTarget wrapped into Gsage wrapper

Public Functions

RenderTargetPtr **create** (**const** std::string &name, *RenderTargetType::Type* type, **const** DataProxy ¶meters, Engine *engine)
Create render target of specified type

Parameters

- name: render target name
- type: *RenderTargetType::Rtt* or *RenderTargetType::Window*
- parameters: additional parameters
- engine: Engine instance

RenderTargetPtr **wrap** (*Ogre::RenderTarget *renderTarget*, **const** std::string &name, **const** DataProxy ¶meters, Engine *engine)
Wrap render target

Parameters

- renderTarget: *Ogre::RenderTarget*
- name: wrapped rt name
- parameters: additional parameters
- engine: Engine instance

class RttRenderTarget : public Gsage::RenderTarget

#include <RenderTarget.h> Wraps *Ogre* RT target

Public Functions

RttRenderTarget (**const** std::string &*name*, **const** DataProxy &*parameters*, Engine **engine*)

virtual ~RttRenderTarget ()

void **setDimensions** (int *width*, int *height*)
Set render target dimensions

Parameters

- *width*:
- *height*:

Protected Functions

void **createTexture** (**const** std::string &*name*, unsigned int *width*, unsigned int *height*, unsigned int *samples*, *Ogre::PixelFormat pixelFormat*)

bool **onTextureEvent** (EventDispatcher **sender*, **const** Event &*event*)

Protected Attributes

TexturePtr **mTexture**

Private Functions

void **wrap** ()

class WindowRenderTarget : **public** *Gsage::RenderTarget*
#include <RenderTarget.h> Wrapped Ogre::RenderWindow

Public Functions

WindowRenderTarget (**const** std::string &*name*, **const** DataProxy &*parameters*, Engine **engine*)

virtual ~WindowRenderTarget ()

void **setDimensions** (int *width*, int *height*)
Set render target dimensions

Parameters

- *width*:
- *height*:

1.24.55 File Renderer.hpp

namespace Gsage

Typedefs

```
typedef std::shared_ptr<Renderer> RendererPtr
```

```
class Renderer
```

Subclassed by *Gsage::ImageRenderer*

Public Functions

```
Renderer (WindowPtr window, SDLCore *core)
```

```
virtual void render () = 0
```

Protected Attributes

SDLCore *mCore

WindowPtr mWindow

Friends

```
friend Gsage::RendererFactory
```

1.24.56 File ResourceManager.h

```
namespace Gsage
```

```
class ResourceManager
```

Public Functions

```
ResourceManager (GsageFacade *facade, MaterialLoader *materialLoader)
```

```
virtual ~ResourceManager (void)
```

```
bool load (const DataProxy &resources, bool initialize = true)
```

Loads all resource groups

Parameters

- resources: DataProxy with all required resources
- initialize: Initializes resource group

```
void unload (const std::string &group)
```

Unload resources

Parameters

- group: Resource group to unload

```
void unload ()
```

Unload all resources

```
void unload (const DataProxy &resources)
```

Unload resources

Parameters

- `resources`: Resource groups DataProxy

```
std::tuple<std::string, std::string> processPath (const std::string &line, const std::string
&workdir = "")
```

Private Members

```
bool mHlmsLoaded
```

```
GsageFacade *mFacade
```

```
MaterialLoader *mMaterialLoader
```

1.24.57 File RocketEvent.h

```
namespace Gsage
```

```
class RocketContextEvent : public Event
```

Public Functions

```
RocketContextEvent (Event::ConstType type, const std::string &name)
```

```
virtual ~RocketContextEvent ()
```

Public Members

```
std::string name
```

Public Static Attributes

```
const Event::Type CREATE
```

1.24.58 File RocketOgreWrapper.h

```
namespace Gsage
```

```
class RocketOgreWrapper : public EventSubscriber<RocketOgreWrapper>, public Gsage::RenderSystemWrapper
```

Public Functions

```
RocketOgreWrapper (Engine *engine)
```

```
virtual ~RocketOgreWrapper ()
```

```
void destroy ()
```

```
Destroy RocketOgreWrapper
```

Protected Functions

void **setUpInterfaces** (unsigned int *width*, unsigned int *height*)
Set up interfaces

Private Functions

bool **render** (EventDispatcher **sender*, const Event &*event*)
void **configureRenderSystem** (*RenderEvent* &*event*)

Private Members

Rocket::Core::RenderInterface ***mRenderInterface**
Rocket::Core::SystemInterface ***mSystemInterface**

1.24.59 File RocketUIManager.h

namespace Gsage

class **RenderSystemWrapper**
Subclassed by *Gsage::RocketOgreWrapper*

Public Types

typedef std::map<std::string, Rocket::Core::Context *> **Contexts**

Public Functions

RenderSystemWrapper (Engine **engine*)
virtual ~**RenderSystemWrapper** ()
virtual Rocket::Core::Context ***getContext** (const std::string &*name*)
Get context
Return rocket context
Parameters

- *name*: Context name

virtual *Contexts* **getContexts** ()
Get all contexts
virtual void **setLuaState** (lua_State **L*)
Set lua state
virtual void **destroy** ()
Destroy render system wrapper

Protected Functions

virtual Rocket::Core::Context ***createContext** (**const** std::string &*name*, unsigned int *width*, unsigned int *height*)

Create context

virtual void **setUpInterfaces** (unsigned int *width*, unsigned int *height*) = 0

Set up interfaces

Protected Attributes

Contexts **mContexts**

Engine ***mEngine**

lua_State ***mLuaState**

bool **mInitialized**

class RocketUIManager : **public** UIManager, **public** EventSubscriber<*RocketUIManager*>

Public Functions

RocketUIManager ()

virtual ~**RocketUIManager** ()

virtual void **initialize** (GsageFacade **facade*, lua_State **L* = 0)

Initialize ui manager

Parameters

- *called*: by gsage facade on setup
- *L*: init with lua state

lua_State ***getLuaState** ()

Gets Rocket lua state

void **setLuaState** (lua_State **L*)

Update load state

Parameters

- *L*: lua_State

void **setUp** ()

Configures rendering

bool **handleSystemChange** (EventDispatcher **sender*, **const** Event &*event*)

SystemChangeEvent::SYSTEM_ADDED and SystemChangeEvent::SYSTEM_REMOVED handler

const std::string &**getType** ()

Public Static Attributes

const std::string **TYPE**

Private Types

```
typedef std::map<KeyboardEvent::Key, Rocket::Core::Input::KeyIdentifier> KeyIdentifierMap
```

Private Functions

```
bool handleMouseEvent (EventDispatcher *sender, const Event &event)  
    Handle mouse event from engine
```

Parameters

- event: Event

```
bool handleKeyboardEvent (EventDispatcher *sender, const Event &event)  
    Handle keyboard event from engine
```

```
bool handleInputEvent (EventDispatcher *sender, const Event &event)  
    Handle keyboard event from engine
```

```
int getKeyModifierState ()  
    Get key modifier state
```

```
void buildKeyMap ()  
    Build Engine <-> Rocket key map
```

```
bool doCapture (Rocket::Core::Context *ctx)  
    Check if mouse event can be captured by any rocket element
```

Private Members

```
RenderSystemWrapper *mRenderSystemWrapper
```

```
KeyIdentifierMap mKeyMap
```

```
unsigned int mModifiersState
```

```
bool mIsSetUp
```

1.24.60 File RocketUIPlugin.h

```
namespace Gsage
```

```
class RocketUIPlugin : public IPlugin
```

Public Functions

```
RocketUIPlugin ()
```

```
virtual ~RocketUIPlugin ()
```

```
virtual const std::string &getName () const  
    Get rocket UI plugin name
```

```
Return RocketUI string
```

```
virtual bool installImpl ()
```

Install rocker ui manager

Return true if succesful

```
virtual void uninstallImpl ()
```

Uninstall rocket ui manager

```
void setupLuaBindings ()
```

Set up lua bindings

Private Members

```
int mUIManagerHandle
```

RocketUIManager **mUIManager**

1.24.61 File SDLAudioSystem.h

```
namespace Gsage
```

```
class SDLAudioSystem
```

Public Functions

```
SDLAudioSystem ()
```

```
virtual ~SDLAudioSystem ()
```

1.24.62 File SDLCore.h

```
namespace Gsage
```

```
class SDLCore : public UpdateListener
```

Public Functions

```
SDLCore ()
```

```
virtual ~SDLCore ()
```

```
bool initialize (const DataProxy &params, GsageFacade *facade)
```

Initialize SDL core

Return true if succeed

```
void tearDown ()
```

Tear down SDL core

```
void update (double time)
```

Update is called on each engine loop

```
void addEventListener (SDLEventListener *listener)
```

Add SDL event listener

Parameters

- listener: *SDLEventListener*

```
void removeEventListener (SDLEventListener *listener)
```

Remove SDL event listener

Parameters

- listener: *SDLEventListener*

```
void setWindowManager (SDLWindowManager *value)
```

Sets active window manager. *SDLCore* will call update for each window

Parameters

- value: *SDLWindowManager*

```
const std::string &getResourcePath () const
```

Get engine resources path

Private Types

```
typedef std::vector<SDLEventListener *> EventListeners
```

Private Members

```
bool mInitialized
```

```
GsageFacade *mFacade
```

```
EventListeners mEventListeners
```

```
SDLWindowManager *mWindowManager
```

1.24.63 File SDLEventListener.h

```
namespace Gsage
```

```
class SDLEventListener
```

Subclassed by *Gsage::SDLInputListener*

Public Functions

```
SDLEventListener ()
```

```
virtual ~SDLEventListener ()
```

```
virtual void handleEvent (SDL_Event *event) = 0
```

Abstract event handling interface

```
virtual void setSDLCore (SDLCore *core)
```

Inject sdl core instance

Parameters

- core: *SDLCore* pointer

Protected Attributes

SDLCore *mSDLCore

1.24.64 File SDLInputListener.h

namespace Gsage

```
class SDLInputFactory : public AbstractInputFactory
```

Public Functions

```
SDLInputFactory ()
```

```
virtual ~SDLInputFactory ()
```

```
InputHandlerPtr create (size_t windowHandle, Engine *engine)
    Create SDL input handler
```

Return InputHandler

Parameters

- windowHandle: Window handle to attach to
- engine: Engine instance

```
void setSDLCore (SDLCore *core)
    Inject SDL core instance
```

Parameters

- core: SDL core

Private Members

SDLCore *mCore

InputHandlerPtr mListener

```
class SDLInputListener : public InputHandler, public EventDispatcher, public Gsage::SDLEventListener
```

Public Functions

```
SDLInputListener (size_t handle, Engine *eventRedirect = 0)
```

```
virtual ~SDLInputListener ()
```

```
void handleEvent (SDL_Event *event)
    Handle sdl event
```

Parameters

- event: SDL event

```
virtual void update (double time)
```

```
virtual void handleResize (unsigned int width, unsigned int height)
```

```
virtual void handleClose ()
```

Private Functions

const MouseEvent::ButtonType **mapButtonType** (Uint8 *sdlButtonID*)

void **handleMouseEvent** (SDL_Event **event*)

Handle mouse event

Parameters

- *event*: SDL_Event

void **handleKeyboardEvent** (SDL_Event **event*)

Handle keyboard event

Parameters

- *event*: SDL_Event

Private Members

std::map<SDL_Keycode, KeyboardEvent::Key> **mKeyMap**

1.24.65 File SDLPlugin.h

namespace Gsage

class SDLPlugin : public IPlugin

Public Functions

SDLPlugin ()

virtual ~SDLPlugin ()

virtual const std::string &**getName** () **const**

Get rocket UI plugin name

Return "SDL" string

virtual bool **installImpl** ()

Install rocker ui manager

Return true if succesful

virtual void **uninstallImpl** ()

Uninstall rocket ui manager

virtual void **setupLuaBindings** ()

Set up lua bindings for imgui

Private Members

SDLCore **mSDLCore**

1.24.66 File SDLRenderer.h

namespace Gsage

class **SDLRenderer**

#include <SDLRenderer.h> Wraps SDL renderer and call updates on each engine update

Public Functions

SDLRenderer (*WindowPtr window*, **const** *DataProxy ¶ms*, *SDL_Renderer *renderer*, *SDL-Core *core*)

virtual **~SDLRenderer** ()

void **render** ()
Update underlying views

Private Types

typedef *std::vector<RendererPtr>* **Renderers**

Private Members

SDL_Renderer ***mRenderer**

Renderers **mRenderers**

1.24.67 File SDLWindowManager.h

namespace Gsage

class **SDLWindow** : **public** *Window*

Public Functions

SDLWindow (**const** *std::string &name*, *SDL_Window *window*)

virtual **~SDLWindow** ()

unsigned long long **getWindowHandle** ()
Get window handle

void ***getGLContext** ()
Get GL context

std::tuple<int, int> **getPosition** () **const**
Get window position

virtual void **setPosition** (int *x*, int *y*)
Sets window position

Parameters

- x:
- y:

virtual std::tuple<int, int> **getSize** ()

Get window size

void **setSize** (int *width*, int *height*)

Set window size

Parameters

- width:
- height:

virtual std::tuple<int, int, int, int> **getDisplayBounds** ()

Get display size information Selects the display which contains current window position

virtual float **getScaleFactor** () **const**

Get screen DPI that is used to display the window

virtual int **getDisplay** () **const**

Get display index that contains the window

virtual void **show** ()

Show window

virtual void **hide** ()

Hide window

Private Members

SDL_Window ***mWindow**

SDL_GLContext **mGLContext**

Friends

friend Gsage::SDLWindowManager

class SDLWindowManager : **public** WindowManager

#include <SDLWindowManager.h> SDL Window Manager implementation

Public Functions

SDLWindowManager (**const** std::string &*type*, *SDLCore* **core*)

virtual ~SDLWindowManager ()

bool **initialize** (**const** DataProxy &*config*)

Initialize SDL system

Parameters

- config: *SDLWindowManager*

WindowPtr **createWindow** (**const** std::string &*name*, unsigned int *width*, unsigned int *height*,
bool *fullscreen*, **const** DataProxy &*params*)

Create a window

Return newly created window pointer

Parameters

- `create`: a window
- `width`: window width
- `height`: window height
- `fullscreen`: create a fullscreen window
- `params`: additional parameters

virtual bool **destroyWindow** (WindowPtr *window*)

Close a window

Return true if succeed

Parameters

- `window`: window pointer to destroy

virtual void **update** (double *time*)

Update is used to render windows, whose rendering is handled by SDL2 itself

Parameters

- `time`: Delta time

Private Types

typedef std::unique_ptr<*SDLRenderer*> **SDLRenderrerPtr**

Private Members

std::mutex **mWindowsMutex**

SDLCore ***mCore**

std::map<std::string, *SDLRenderrerPtr*> **mRenderers**

1.24.68 File SceneNodeWrapper.h

namespace **Gsage**

class **SceneNodeWrapper** : **public** *Gsage::OgreObject*, **public** EventSubscriber<*SceneNodeWrapper*>

Public Functions

SceneNodeWrapper ()

virtual ~**SceneNodeWrapper** ()

bool **initialize** (*OgreObjectManager* **objectManager*, **const** DataProxy &*dict*, **const** std::string &*ownerId*, **const** std::string &*type*, *Ogre::SceneManager* **sceneManager*, *Ogre::SceneNode* **parent*)

Initialize element from the node with values

Parameters

- `factory`: *OgreObjectManager* to enable child class creation
- `dict`: DataProxy with values
- `type`: String type of the object

- `parent`: Parent SceneNode
- `sceneManager`: SceneManager to use
- `ownerId`: Id of entity that owns the element

bool **hasNode** ()

Check that node has *Ogre* node created

void **createNode** (const std::string &*id*)

Create Ogre::SceneNode with the specified *id*

Parameters

- *id*: Node *id*

const std::string &**getId** () const

Get node *id*

void **setPosition** (const Ogre::Vector3 &*position*)

Set node position with defined offset

Parameters

- *position*: Ogre::Vector3 position

void **setPositionWithoutOffset** (const Ogre::Vector3 &*position*)

Set node position without defined offset

Parameters

- *position*: Ogre::Vector3 position

Ogre::Vector3 **getPosition** ()

Get node position

Ogre::Vector3 **getPositionWithoutOffset** ()

Get node position without correction that is defined in offset field

void **setScale** (const Ogre::Vector3 &*scale*)

Set SceneNode scale

Parameters

- *scale*: Ogre::Vector3 scale in all dimensions

Ogre::Vector3 **getScale** ()

Get node scale

void **setOrientation** (const Ogre::Quaternion &*rotation*)

Set node orientation

Parameters

- *rotation*: Orientation quaternion

Ogre::Quaternion **getOrientation** ()

Get node orientation

void **readChildren** (const DataProxy &*dict*)

DataProxy **writeChildren** ()

void **setOrientationVector** (const Ogre::Vector3 &*value*)

Set vector that defines “face” of the node

Parameters

- *value*: Ogre::Vector3 normalized/not normalized vector (it is normalized anyway)

const Ogre::Vector3 &**getOrientationVector** () const

Get vector that defines “face” of the node

```

void rotate (const Ogre::Quaternion &rotation, const Ogre::Node::TransformSpace ts)
    Rotate node
Parameters
    • rotation: Ogre::Quaternion that defines rotation

void rotate (const Ogre::Vector3 &axis, const Ogre::Degree &degree, const
    Ogre::Node::TransformSpace ts)
    Rotate node
Parameters
    • axis: Axis to rotate around
    • degree: Rotation degree
    • ts: Transform space

void lookAt (const Ogre::Vector3 &position, Ogre::Node::TransformSpace relativeTo)
    Rotate node to look at position
Parameters
    • position: Position to look at
    • relativeTo: Coordinate space to use

void destroy ()
    Remove all children and destroy node

bool attach (Ogre::MovableObject *object, const DataProxy &params)
    Attach movable object to this object
Parameters
    • object: Object to attach

void destroyAllAttachedMovableObjects (Ogre::SceneNode *node)
    Remove all attached movable objects from the node
Parameters
    • node: Node to process

OgreObject *getChild (const std::string &type, const std::string &name, bool traverse =
    false)
    Get child
Parameters
    • type: Type of the child
    • name: Name, that was defined in the “name” field
    • traverse: Traverse children splitting name by .

template<class T>
T *getChildOfType (const std::string &name)
    Get child of specific type
Parameters
    • type: Type of the child
    • name: Name, that was defined in the “name” field

IMovableObjectWrapper *getMovableObject (const std::string &type, const std::string
    &name)

void pitch (const Ogre::Radian &angle, Ogre::Node::TransformSpace relativeTo =
    Ogre::Node::TS_LOCAL)
    Rotate the node around X axis
Parameters
    • angle: Rotation angle
    • relativeTo: Transformation space

```

```
void yaw (const  Ogre::Radian  &angle,  Ogre::Node::TransformSpace  relativeTo  =
         Ogre::Node::TS_LOCAL)
    Rotate the node around Y axis
```

Parameters

- angle: Rotation angle
- relativeTo: Transformation space

```
void roll (const  Ogre::Radian  &angle,  Ogre::Node::TransformSpace  relativeTo  =
         Ogre::Node::TS_LOCAL)
    Rotate the node around Z axis
```

Parameters

- angle: Rotation angle
- relativeTo: Transformation space

```
void translate (const  Ogre::Vector3 &d)
    Moves the node along the Cartesian axes
```

Parameters

- d: Vector with x,y,z values representing the translation

```
void translate (const  Ogre::Vector3  &d,  Ogre::Node::TransformSpace  relativeTo  =
         Ogre::Node::TS_LOCAL)
    Moves the node along the Cartesian axes
```

Parameters

- d: Vector with x,y,z values representing the translation
- relativeTo: Transformation space

```
Ogre::SceneNode *getNode ()
    Get Ogre::SceneNode
```

Public Static Attributes

```
const std::string TYPE
```

Private Types

```
typedef std::map<const std::string,  OgreObject *> ObjectCollection
```

```
typedef std::map<const std::string,  ObjectCollection> Objects
```

Private Functions

```
bool onFactoryUnregister (EventDispatcher *sender, const Event &event)
    Handle factory removal event
```

Parameters

- event: *OgreObjectManager* event

Private Members

```
std::string mId
```

```
Ogre::Vector3 mOrientationVector
```

```
Ogre::SceneNode *mNode
Ogre::Vector3 mOffset
Objects mChildren
```

1.24.69 File SystemInterfaceOgre3D.h

```
class SystemInterfaceOgre3D : public SystemInterface
    #include <SystemInterfaceOgre3D.h> A sample system interface for Rocket into Ogre3D.
```

Author Peter Curry

Public Functions

```
SystemInterfaceOgre3D ()

virtual ~SystemInterfaceOgre3D ()

virtual float GetElapsedTime ()
    Gets the number of seconds elapsed since the start of the application.

virtual bool LogMessage (Rocket::Core::Log::Type type, const Rocket::Core::String &message)
    Logs the specified message.
```

Private Members

```
Ogre::Timer timer
```

1.24.70 File ViewportRenderable.h

```
namespace Gsage
```

```
class ViewportRenderData
```

Public Functions

```
ViewportRenderData ()

virtual ~ViewportRenderData ()

void updatePos (ImVec2 pos)
    Update position

void updateSize (ImVec2 size)
    Update size

void updateUVs (const Texture::UVs &uvs)
    Update UV

void updateVertexBuffer ()
    Update vertex buffer
```

```
void resetDatablock ()  
    Removes datablock  
  
void setDatablock (const Ogre::String &name)  
    Set viewport datablock  
  
Ogre::TexturePtr getRenderTexture ()  
    Get RTT to render
```

Private Members

```
ImVec2 pos  
ImVec2 size  
ImDrawCmd mDrawCmd  
ImDrawVert mVertexBuffer[4]  
ImDrawIdx mIndexBuffer[6]  
Ogre::String mTextureName  
Ogre::TextureUnitState *mTexUnitState  
ImVec2 mPos  
ImVec2 mSize  
bool mDirty
```

Friends

```
friend Gsage::ImGuiRendererV1
```

1.24.71 File WindowEventListener.h

```
namespace Gsage
```

```
class WindowEventListener : public WindowEventListener  
    #include <WindowEventListener.h> Proxy ogre window resize events to the engine
```

Public Functions

```
WindowEventListener (Ogre::RenderWindow *window, Engine *engine)  
  
virtual ~WindowEventListener ()  
  
virtual void windowResized (Ogre::RenderWindow *window)  
    windowResized handler  
  
    Parameters  
        • window: Ogre::RenderWindow  
  
virtual void windowClosed (Ogre::RenderWindow *window)  
    windowClosed handler
```

Parameters

- window: Ogre::RenderWindow

Private Functions

void **fireWindowEvent** (Event::ConstType *type*, *Ogre::RenderWindow *window*)

Private Members

Engine ***mEngine**

1.24.72 File WorkspaceEvent.h

namespace Ogre

File: MovableText.h

Description: This creates a billboard object that display a text. Note: This object must have a dedicated scene node since it will rotate it to face the camera (OGRE 2.1)

Author 2003 by cTh see gavocanov@rambler.ru 2006 by barraq see nospam@barraquand.com 2012 to work with newer versions of OGRE by MindCalamity mindcalamity@gmail.com 2015 to work on OGRE 2.1 (but not on older versions anymore) by Jayray jeremy.richert1@gmail.com

- See “Notes” on: <http://www.ogre3d.org/tikiwiki/tiki-editpage.php?page=MovableText>

namespace Gsage

class WorkspaceEvent : public Event

#include <WorkspaceEvent.h> Fired when a new Ogre::CompositorWorkspace is created

Public Functions

WorkspaceEvent (Event::ConstType *type*, *Ogre::CompositorWorkspace *workspace*)

virtual ~WorkspaceEvent ()

Public Members

Ogre::CompositorWorkspace ***mWorkspace**

Public Static Attributes

const Event::Type CREATE

Create event

1.24.73 File `imgui_extensions.h`

Defines

`IMGUI_DEFINE_MATH_OPERATORS`

`namespace ImGui`

Functions

`IMGUI_API bool ImGui::Spinner(const char * label, float radius, int thickness, const ImVec2 &pos, const ImVec4 &col)`

`static void AddConvexPolyFilled(ImDrawList *drawList, const ImVec2 *points, const int points_count, const Gradient &gradient)`

`static void PathFillConvex(ImDrawList *drawList, const Gradient &gradient)`

`struct Gradient`

Subclassed by *ImGui::VerticalGradient*

Public Functions

`virtual ImU32 Calc(const ImVec2 &pos) const = 0`

`struct VerticalGradient : public ImGui::Gradient`

Public Functions

`VerticalGradient(const ImVec2 &start, const ImVec2 &end, const ImVec4 &col0, const ImVec4 &col1)`

`VerticalGradient(const ImVec2 &start, const ImVec2 &end, ImU32 col0, ImU32 col1)`

`void evalStep()`

`virtual ImU32 Calc(const ImVec2 &pos) const`

Public Members

`ImVec4 Col0`

`ImVec4 Col1`

`ImVec2 Start`

`ImVec2 End`

`float Len`

1.24.74 File ImGuizmo.h

Warning: doxygenfile: Found multiple matches for file “ImGuizmo.h

1.24.75 File ImGuiRenderable.h

Warning: doxygenfile: Found multiple matches for file “ImGuiRenderable.h

1.24.76 File ManualObjectWrapper.h

Warning: doxygenfile: Found multiple matches for file “ManualObjectWrapper.h

1.24.77 File MovableText.h

Warning: doxygenfile: Found multiple matches for file “MovableText.h

1.24.78 File ImGuizmo.h

Warning: doxygenfile: Found multiple matches for file “ImGuizmo.h

1.24.79 File ImGuiRenderable.h

Warning: doxygenfile: Found multiple matches for file “ImGuiRenderable.h

1.24.80 File ManualObjectWrapper.h

Warning: doxygenfile: Found multiple matches for file “ManualObjectWrapper.h

1.24.81 File MovableText.h

Warning: doxygenfile: Found multiple matches for file “MovableText.h

CHAPTER
TWO

LINKS

Project Tracker

Symbols

__NODE_ID_TYPE (*C macro*), 131

B

BBT_ORIENTED_COMMON_ID (*C macro*), 61

BBT_ORIENTED_SELF_ID (*C macro*), 61

BBT_PERPENDICULAR_COMMON_ID (*C macro*), 61

BBT_PERPENDICULAR_SELF_ID (*C macro*), 61

BBT_POINT_ID (*C macro*), 61

D

DEFAULT_FADE_SPEED (*C macro*), 56

G

GET_IV_DATA (*C macro*), 69

Gsage (*C++ type*), 48, 56, 61, 64, 67–69, 71, 72, 74–76, 85–88, 93–99, 105, 109, 112–115, 118, 120, 124, 129–131, 133, 135, 139–142, 144–149, 151, 155–157

Gsage::Animation (*C++ class*), 36, 50, 56

Gsage::Animation::~~Animation (*C++ function*), 56

Gsage::Animation::Animation (*C++ function*), 56

Gsage::Animation::disable (*C++ function*), 56

Gsage::Animation::enable (*C++ function*), 56

Gsage::Animation::getEnabled (*C++ function*), 57

Gsage::Animation::getLength (*C++ function*), 56

Gsage::Animation::getLoop (*C++ function*), 57

Gsage::Animation::getSpeed (*C++ function*), 57

Gsage::Animation::getTimePosition (*C++ function*), 56

Gsage::Animation::hasEnded (*C++ function*), 57

Gsage::Animation::initialize (*C++ function*), 56

Gsage::Animation::isEnding (*C++ function*), 57

Gsage::Animation::isFadingOut (*C++ function*), 57

Gsage::Animation::isInitialized (*C++ function*), 57

Gsage::Animation::mAnimationState (*C++ member*), 57

Gsage::Animation::mFadeIn (*C++ member*), 57

Gsage::Animation::mFadeOut (*C++ member*), 57

Gsage::Animation::mFadeTime (*C++ member*), 57

Gsage::Animation::mSpeed (*C++ member*), 57

Gsage::Animation::rewind (*C++ function*), 57

Gsage::Animation::setEnabled (*C++ function*), 57

Gsage::Animation::setLoop (*C++ function*), 57

Gsage::Animation::setSpeed (*C++ function*), 57

Gsage::Animation::setPosition (*C++ function*), 56

Gsage::Animation::update (*C++ function*), 56

Gsage::AnimationController (*C++ class*), 36, 50, 58

Gsage::AnimationController::~~AnimationController (*C++ function*), 58

Gsage::AnimationController::AnimationController (*C++ function*), 58

Gsage::AnimationController::finish (*C++ function*), 58

Gsage::AnimationController::hasEnded (*C++ function*), 58

Gsage::AnimationController::mAnimation (*C++ member*), 58

Gsage::AnimationController::mOffset (*C++ member*), 58

Gsage::AnimationController::mSpeed (*C++ member*), 58

Gsage::AnimationController::operator!= (*C++ function*), 58

Gsage::AnimationController::operator= (*C++ function*), 58

Gsage::AnimationController::operator==

(C++ function), 58	Gsage::AnimationScheduler::mCurrentAnimation (C++ member), 61
Gsage::AnimationController::start (C++ function), 58	Gsage::AnimationScheduler::mDefaultAnimation (C++ member), 61
Gsage::AnimationGroup (C++ class), 36, 50, 58	Gsage::AnimationScheduler::mDefaultAnimationSpeed (C++ member), 61
Gsage::AnimationGroup::~~AnimationGroup (C++ function), 58	Gsage::AnimationScheduler::mInitialized (C++ member), 61
Gsage::AnimationGroup::AnimationGroup (C++ function), 58	Gsage::AnimationScheduler::mRenderComponent (C++ member), 61
Gsage::AnimationGroup::Animations (C++ type), 59	Gsage::AnimationScheduler::mSceneManager (C++ member), 61
Gsage::AnimationGroup::getSpeed (C++ function), 58	Gsage::AnimationScheduler::play (C++ function), 59
Gsage::AnimationGroup::hasEnded (C++ function), 59	Gsage::AnimationScheduler::playDefaultAnimation (C++ function), 60
Gsage::AnimationGroup::initialize (C++ function), 58	Gsage::AnimationScheduler::queueAnimation (C++ function), 60
Gsage::AnimationGroup::mAnimations (C++ member), 59	Gsage::AnimationScheduler::resetState (C++ function), 60
Gsage::AnimationGroup::mRenderComponent (C++ member), 59	Gsage::AnimationScheduler::setRenderComponent (C++ function), 59
Gsage::AnimationGroup::mSpeed (C++ member), 59	Gsage::AnimationScheduler::setStates (C++ function), 60
Gsage::AnimationGroup::setSpeed (C++ function), 58	Gsage::AnimationScheduler::update (C++ function), 60
Gsage::AnimationScheduler (C++ class), 36, 50, 59	Gsage::BillboardSetWrapper (C++ class), 36, 61
Gsage::AnimationScheduler::~~AnimationScheduler (C++ function), 59	Gsage::BillboardSetWrapper::~~BillboardSetWrapper (C++ function), 61
Gsage::AnimationScheduler::adjustSpeed (C++ function), 59	Gsage::BillboardSetWrapper::Billboards (C++ type), 62
Gsage::AnimationScheduler::AnimationGroup (C++ type), 60	Gsage::BillboardSetWrapper::BillboardSetWrapper (C++ function), 61
Gsage::AnimationScheduler::AnimationQueue (C++ type), 60	Gsage::BillboardSetWrapper::getBillboards (C++ function), 62
Gsage::AnimationScheduler::AnimationQueue (C++ type), 60	Gsage::BillboardSetWrapper::getBillboardType (C++ function), 62
Gsage::AnimationScheduler::Animations (C++ type), 60	Gsage::BillboardSetWrapper::getCommonDirection (C++ function), 62
Gsage::AnimationScheduler::AnimationScheduler (C++ function), 59	Gsage::BillboardSetWrapper::getCommonUpVector (C++ function), 61
Gsage::AnimationScheduler::getStates (C++ function), 60	Gsage::BillboardSetWrapper::getMaterialName (C++ function), 62
Gsage::AnimationScheduler::initialize (C++ function), 59	Gsage::BillboardSetWrapper::mapBillboardType (C++ function), 62
Gsage::AnimationScheduler::isQueued (C++ function), 60	Gsage::BillboardSetWrapper::mBillboards (C++ member), 62
Gsage::AnimationScheduler::mAnimationGroup (C++ member), 61	Gsage::BillboardSetWrapper::mBillboardSet (C++ member), 62
Gsage::AnimationScheduler::mAnimationQueue (C++ member), 61	Gsage::BillboardSetWrapper::mMaterialName (C++ member), 62
Gsage::AnimationScheduler::mAnimations (C++ member), 61	Gsage::BillboardSetWrapper::read (C++ function), 61
Gsage::AnimationScheduler::mAnimationState (C++ member), 61	

Gsage::BillboardSetWrapper::setBillboards *function*), 63
 (C++ *function*), 62 Gsage::BlitDirty (C++ *enumerator*), 106
 Gsage::BillboardSetWrapper::setBillboardType (C++ *enumerator*), 77
 (C++ *function*), 62 Gsage::BoundingBoxToAxisAlignedBox (C++
 Gsage::BillboardSetWrapper::setCommonDirection *function*), 49, 114
 (C++ *function*), 61 Gsage::CameraWrapper (C++ *class*), 36, 64
 Gsage::BillboardSetWrapper::setCommonUpVector (C++ *function*), 61
 (C++ *function*), 61 Gsage::CameraWrapper::~~CameraWrapper
 Gsage::BillboardSetWrapper::setMaterialName (C++ *function*), 62
 (C++ *function*), 62 Gsage::CameraWrapper::attach (C++ *func-*
 Gsage::BillboardSetWrapper::TYPE (C++ *member*), 62 Gsage::CameraWrapper::CameraWrapper
 (C++ *function*), 64
 Gsage::BillboardWrapper (C++ *class*), 36, 63 Gsage::CameraWrapper::createCamera (C++
 Gsage::BillboardWrapper::~~BillboardWrapper *function*), 65
 (C++ *function*), 63 Gsage::CameraWrapper::destroy (C++ *func-*
 Gsage::BillboardWrapper::BillboardWrapper *tion*), 65
 (C++ *function*), 63 Gsage::CameraWrapper::detach (C++ *func-*
 Gsage::BillboardWrapper::getColour (C++ *tion*), 65
function), 63 Gsage::CameraWrapper::getBgColour (C++
 Gsage::BillboardWrapper::getHeight (C++ *function*), 65
function), 63 Gsage::CameraWrapper::getCamera (C++
 Gsage::BillboardWrapper::getPosition (C++ *function*), 65
function), 63 Gsage::CameraWrapper::getClipDistance
 Gsage::BillboardWrapper::getRotation (C++ *function*), 65
function), 63 Gsage::CameraWrapper::getName (C++ *func-*
 Gsage::BillboardWrapper::getTexcoordIndex *tion*), 65
 (C++ *function*), 64 Gsage::CameraWrapper::getOrientation
 Gsage::BillboardWrapper::getTexcoordRect (C++ *function*), 64
 (C++ *function*), 64 Gsage::CameraWrapper::getProjectionMatrix
 Gsage::BillboardWrapper::getWidth (C++ *function*), 65
function), 63 Gsage::CameraWrapper::getRenderTarget
 Gsage::BillboardWrapper::initialize (C++ *function*), 65
 (C++ *function*), 63 Gsage::CameraWrapper::getViewMatrix
 Gsage::BillboardWrapper::mBillboard (C++ *member*), 64
 (C++ *member*), 64 Gsage::CameraWrapper::isActive (C++ *func-*
 Gsage::BillboardWrapper::mBillboardSet *tion*), 65
 (C++ *member*), 64 Gsage::CameraWrapper::mBgColour (C++
 Gsage::BillboardWrapper::mHeight (C++ *member*), 66
member), 64 Gsage::CameraWrapper::mIsActive (C++
 Gsage::BillboardWrapper::mWidth (C++ *member*), 66
member), 64 Gsage::CameraWrapper::mRenderTarget
 Gsage::BillboardWrapper::setColour (C++ *function*), 63
 (C++ *function*), 63 Gsage::CameraWrapper::mTarget (C++ *mem-*
 Gsage::BillboardWrapper::setHeight (C++ *ber*), 66
function), 63 Gsage::CameraWrapper::mViewport (C++
 Gsage::BillboardWrapper::setPosition (C++ *member*), 66
function), 63 Gsage::CameraWrapper::mWindow (C++ *mem-*
 Gsage::BillboardWrapper::setRotation (C++ *ber*), 66
function), 63 Gsage::CameraWrapper::objectDestroyed
 Gsage::BillboardWrapper::setTexcoordIndex (C++ *function*), 65
 (C++ *function*), 64 Gsage::CameraWrapper::setBgColour (C++
 Gsage::BillboardWrapper::setTexcoordRect (C++ *function*), 65
 (C++ *function*), 64 Gsage::CameraWrapper::setClipDistance
 Gsage::BillboardWrapper::setWidth (C++ *function*), 65
 (C++ *function*), 65

Gsage::CameraWrapper::setOrientation (C++ function), 64
 Gsage::CameraWrapper::TYPE (C++ member), 66
 Gsage::CustomPass (C++ class), 37, 50, 68
 Gsage::CustomPass::~~CustomPass (C++ function), 68
 Gsage::CustomPass::CustomPass (C++ function), 68
 Gsage::CustomPass::mEngine (C++ member), 68
 Gsage::CustomPassDef (C++ class), 37, 50, 68
 Gsage::CustomPassDef::~~CustomPassDef (C++ function), 68
 Gsage::CustomPassDef::CustomPassDef (C++ function), 68
 Gsage::CustomPassDef::getID (C++ function), 68
 Gsage::CustomPassDef::mEngine (C++ member), 69
 Gsage::CustomPassDef::mID (C++ member), 69
 Gsage::CustomPassProvider (C++ class), 37, 67
 Gsage::CustomPassProvider::~~CustomPassProvider (C++ function), 67
 Gsage::CustomPassProvider::addPass (C++ function), 67
 Gsage::CustomPassProvider::addPassDef (C++ function), 67
 Gsage::CustomPassProvider::CustomPassProvider (C++ function), 67
 Gsage::CustomPassProvider::DefFactoryFunction (C++ type), 68
 Gsage::CustomPassProvider::DefFactoryFunction (C++ type), 68
 Gsage::CustomPassProvider::initialize (C++ function), 67
 Gsage::CustomPassProvider::mDefFactories (C++ member), 68
 Gsage::CustomPassProvider::mEngine (C++ member), 68
 Gsage::CustomPassProvider::mPassFactories (C++ member), 68
 Gsage::CustomPassProvider::PassFactoryFunction (C++ type), 68
 Gsage::CustomPassProvider::PassFactoryFunction (C++ type), 68
 Gsage::CustomPassProvider::registerPassDef (C++ function), 67
 Gsage::Dock (C++ class), 37, 76
 Gsage::Dock::~~Dock (C++ function), 77
 Gsage::Dock::activateOther (C++ function), 77
 Gsage::Dock::addChild (C++ function), 78
 Gsage::Dock::addChildAt (C++ function), 78
 Gsage::Dock::anyTabOpen (C++ function), 78
 Gsage::Dock::bothChildrenVisible (C++ function), 77
 Gsage::Dock::calculateRatio (C++ function), 79
 Gsage::Dock::Dock (C++ function), 77
 Gsage::Dock::docked (C++ function), 78
 Gsage::Dock::getActive (C++ function), 78
 Gsage::Dock::getBoundingRect (C++ function), 79
 Gsage::Dock::getChildAt (C++ function), 78
 Gsage::Dock::getLabel (C++ function), 79
 Gsage::Dock::getLayout (C++ function), 78
 Gsage::Dock::getLocation (C++ function), 78
 Gsage::Dock::getNextTab (C++ function), 79
 Gsage::Dock::getOpened (C++ function), 78
 Gsage::Dock::getParent (C++ function), 79
 Gsage::Dock::getPosition (C++ function), 78
 Gsage::Dock::getPreviousTab (C++ function), 79
 Gsage::Dock::getRatio (C++ function), 77
 Gsage::Dock::getSize (C++ function), 78
 Gsage::Dock::getTitle (C++ function), 79
 Gsage::Dock::hasBothChildren (C++ function), 77
 Gsage::Dock::isContainer (C++ function), 77
 Gsage::Dock::locationToIndex (C++ function), 79
 Gsage::Dock::mActive (C++ member), 79
 Gsage::Dock::mBounds (C++ member), 79
 Gsage::Dock::mChildren (C++ member), 79
 Gsage::Dock::mDirty (C++ member), 79
 Gsage::Dock::mFlags (C++ member), 80
 Gsage::Dock::mLabel (C++ member), 79
 Gsage::Dock::mLayout (C++ member), 79
 Gsage::Dock::mLocation (C++ member), 79
 Gsage::Dock::mNextTab (C++ member), 79
 Gsage::Dock::mOpened (C++ member), 79
 Gsage::Dock::mParent (C++ member), 79
 Gsage::Dock::mPos (C++ member), 79
 Gsage::Dock::mPreviousTab (C++ member), 79
 Gsage::Dock::mRatio (C++ member), 80
 Gsage::Dock::mRendered (C++ member), 79
 Gsage::Dock::mResized (C++ member), 79
 Gsage::Dock::mSize (C++ member), 79
 Gsage::Dock::mStyle (C++ member), 80
 Gsage::Dock::mTitle (C++ member), 79
 Gsage::Dock::removeChildAt (C++ function), 78
 Gsage::Dock::reset (C++ function), 77
 Gsage::Dock::setActive (C++ function), 78
 Gsage::Dock::setDimensions (C++ function), 77
 Gsage::Dock::setOpened (C++ function), 78

Gsage::Dock::setRatio (C++ *function*), 77
 Gsage::Dock::updateChildren (C++ *function*), 77
 Gsage::Dock::visible (C++ *function*), 78
 Gsage::DockPtr (C++ *type*), 49, 76
 Gsage::DockSlotPreviewFill (C++ *enumerator*), 49, 76
 Gsage::DockSlotPreviewOutline (C++ *enumerator*), 49, 76
 Gsage::DockSlotPreviewStyle (C++ *enum*), 49, 76
 Gsage::dumpState (C++ *function*), 49, 76
 Gsage::DYNAMIC (C++ *enumerator*), 69, 98
 Gsage::End_Child (C++ *enumerator*), 83
 Gsage::End_None (C++ *enumerator*), 83
 Gsage::End_Window (C++ *enumerator*), 83
 Gsage::EndCommand (C++ *enum*), 83
 Gsage::EntityWrapper (C++ *class*), 37, 69
 Gsage::EntityWrapper::~~EntityWrapper (C++ *function*), 69
 Gsage::EntityWrapper::attach (C++ *function*), 70
 Gsage::EntityWrapper::AttachedEntities (C++ *type*), 71
 Gsage::EntityWrapper::attachToBone (C++ *function*), 70
 Gsage::EntityWrapper::createCloneWithMaterial (C++ *function*), 70
 Gsage::EntityWrapper::EntityWrapper (C++ *function*), 69
 Gsage::EntityWrapper::getAabb (C++ *function*), 71
 Gsage::EntityWrapper::getCastShadows (C++ *function*), 70
 Gsage::EntityWrapper::getEntity (C++ *function*), 70
 Gsage::EntityWrapper::getMesh (C++ *function*), 70
 Gsage::EntityWrapper::getQueryFlags (C++ *function*), 70
 Gsage::EntityWrapper::getRenderQueue (C++ *function*), 70
 Gsage::EntityWrapper::getResourceGroup (C++ *function*), 70
 Gsage::EntityWrapper::mAnimBlendMode (C++ *member*), 71
 Gsage::EntityWrapper::mAttachedEntities (C++ *member*), 71
 Gsage::EntityWrapper::mClone (C++ *member*), 71
 Gsage::EntityWrapper::mMeshName (C++ *member*), 71
 Gsage::EntityWrapper::mQuery (C++ *member*), 71
 Gsage::EntityWrapper::mQueryString (C++ *member*), 71
 Gsage::EntityWrapper::mResourceGroup (C++ *member*), 71
 Gsage::EntityWrapper::removeClone (C++ *function*), 70
 Gsage::EntityWrapper::setCastShadows (C++ *function*), 70
 Gsage::EntityWrapper::setMesh (C++ *function*), 70
 Gsage::EntityWrapper::setQueryFlags (C++ *function*), 69
 Gsage::EntityWrapper::setRenderQueue (C++ *function*), 70
 Gsage::EntityWrapper::setResourceGroup (C++ *function*), 70
 Gsage::EntityWrapper::TYPE (C++ *member*), 71
 Gsage::Flags (C++ *enum*), 106
 Gsage::Generic (C++ *enumerator*), 135
 Gsage::Gizmo (C++ *class*), 34, 37, 50, 72
 Gsage::Gizmo::~~Gizmo (C++ *function*), 72
 Gsage::Gizmo::addTarget (C++ *function*), 72
 Gsage::Gizmo::drawCoordinatesEditor (C++ *function*), 73
 Gsage::Gizmo::enable (C++ *function*), 72
 Gsage::Gizmo::extractMatrix (C++ *function*), 73
 Gsage::Gizmo::getMode (C++ *function*), 73
 Gsage::Gizmo::getOperation (C++ *function*), 73
 Gsage::Gizmo::getOrientation (C++ *function*), 73
 Gsage::Gizmo::getPosition (C++ *function*), 73
 Gsage::Gizmo::getScale (C++ *function*), 73
 Gsage::Gizmo::Gizmo (C++ *function*), 72
 Gsage::Gizmo::mEnabled (C++ *member*), 74
 Gsage::Gizmo::mInitialPositions (C++ *member*), 73
 Gsage::Gizmo::mInitialScales (C++ *member*), 73
 Gsage::Gizmo::mMode (C++ *member*), 74
 Gsage::Gizmo::mModelMatrix (C++ *member*), 73
 Gsage::Gizmo::mOperation (C++ *member*), 74
 Gsage::Gizmo::mOrientation (C++ *member*), 74
 Gsage::Gizmo::mPosition (C++ *member*), 74
 Gsage::Gizmo::mPositions (C++ *member*), 73
 Gsage::Gizmo::mRenderSystem (C++ *member*), 73
 Gsage::Gizmo::mScale (C++ *member*), 74
 Gsage::Gizmo::mTargets (C++ *member*), 73
 Gsage::Gizmo::mUsing (C++ *member*), 74

Gsage::Gizmo::onTargetDestroyed (C++ [function](#)), [73](#)
 Gsage::Gizmo::Positions (C++ [type](#)), [73](#)
 Gsage::Gizmo::removeTarget (C++ [function](#)), [72](#)
 Gsage::Gizmo::render (C++ [function](#)), [72](#)
 Gsage::Gizmo::resetTargets (C++ [function](#)), [72](#)
 Gsage::Gizmo::rotate (C++ [function](#)), [73](#)
 Gsage::Gizmo::setMode (C++ [function](#)), [73](#)
 Gsage::Gizmo::setOperation (C++ [function](#)), [73](#)
 Gsage::Gizmo::setScale (C++ [function](#)), [73](#)
 Gsage::Gizmo::Targets (C++ [type](#)), [73](#)
 Gsage::Gizmo::updateTargetNode (C++ [function](#)), [73](#)
 Gsage::GsageOgrePlugin (C++ [class](#)), [38](#), [74](#)
 Gsage::GsageOgrePlugin::~~GsageOgrePlugin (C++ [function](#)), [74](#)
 Gsage::GsageOgrePlugin::getName (C++ [function](#)), [74](#)
 Gsage::GsageOgrePlugin::GsageOgrePlugin (C++ [function](#)), [74](#)
 Gsage::GsageOgrePlugin::installImpl (C++ [function](#)), [74](#)
 Gsage::GsageOgrePlugin::setupLuaBindings (C++ [function](#)), [74](#)
 Gsage::GsageOgrePlugin::uninstallImpl (C++ [function](#)), [74](#)
 Gsage::GsageQuaternionToOgreQuaternion (C++ [function](#)), [49](#), [114](#)
 Gsage::GsageVector3ToOgreVector3 (C++ [function](#)), [49](#), [114](#)
 Gsage::HlmsUnlit (C++ [class](#)), [38](#), [50](#), [75](#)
 Gsage::HlmsUnlit::~~HlmsUnlit (C++ [function](#)), [75](#)
 Gsage::HlmsUnlit::createDatablockImpl (C++ [function](#)), [75](#)
 Gsage::HlmsUnlit::CustomProjectionMatrix (C++ [type](#)), [75](#)
 Gsage::HlmsUnlit::fillBuffersForV2 (C++ [function](#)), [75](#)
 Gsage::HlmsUnlit::getDefaultPaths (C++ [function](#)), [75](#)
 Gsage::HlmsUnlit::HlmsUnlit (C++ [function](#)), [75](#)
 Gsage::HlmsUnlit::mCustomProjectionMatrix (C++ [member](#)), [76](#)
 Gsage::HlmsUnlit::setUseCustomProjectionMatrix (C++ [function](#)), [75](#)
 Gsage::HlmsUnlitDatablock (C++ [class](#)), [38](#), [74](#)
 Gsage::HlmsUnlitDatablock::HlmsUnlitDatablock (C++ [function](#)), [74](#)
 Gsage::Horizontal (C++ [enumerator](#)), [77](#)
 Gsage::ImageRenderer (C++ [class](#)), [39](#), [50](#), [85](#)
 Gsage::ImageRenderer::~~ImageRenderer (C++ [function](#)), [85](#)
 Gsage::ImageRenderer::ImageRenderer (C++ [function](#)), [85](#)
 Gsage::ImageRenderer::mCustomRect (C++ [member](#)), [86](#)
 Gsage::ImageRenderer::mDestRect (C++ [member](#)), [86](#)
 Gsage::ImageRenderer::mImage (C++ [member](#)), [86](#)
 Gsage::ImageRenderer::mRenderer (C++ [member](#)), [86](#)
 Gsage::ImageRenderer::mTexture (C++ [member](#)), [86](#)
 Gsage::ImageRenderer::render (C++ [function](#)), [85](#)
 Gsage::ImGuiDock_NoResize (C++ [enumerator](#)), [49](#), [76](#)
 Gsage::ImGuiDock_NoTitleBar (C++ [enumerator](#)), [49](#), [76](#)
 Gsage::ImGuiDockFlags (C++ [enum](#)), [49](#), [76](#)
 Gsage::ImGuiDockspace (C++ [class](#)), [38](#), [51](#), [80](#)
 Gsage::ImGuiDockspace::~~ImGuiDockspace (C++ [function](#)), [80](#)
 Gsage::ImGuiDockspace::addChild (C++ [function](#)), [81](#)
 Gsage::ImGuiDockspace::createDock (C++ [function](#)), [81](#)
 Gsage::ImGuiDockspace::Docks (C++ [type](#)), [81](#)
 Gsage::ImGuiDockspace::dockTo (C++ [function](#)), [80](#)
 Gsage::ImGuiDockspace::getDock (C++ [function](#)), [81](#)
 Gsage::ImGuiDockspace::getDockAt (C++ [function](#)), [81](#)
 Gsage::ImGuiDockspace::getRootDock (C++ [function](#)), [81](#)
 Gsage::ImGuiDockspace::getState (C++ [function](#)), [80](#)
 Gsage::ImGuiDockspace::ImGuiDockspace (C++ [function](#)), [80](#)
 Gsage::ImGuiDockspace::LocationToLayout (C++ [type](#)), [81](#)
 Gsage::ImGuiDockspace::mDocks (C++ [member](#)), [81](#)
 Gsage::ImGuiDockspace::mLocationToLayout (C++ [member](#)), [81](#)
 Gsage::ImGuiDockspace::mRootDock (C++ [member](#)), [81](#)
 Gsage::ImGuiDockspace::mSize (C++ [member](#)), [81](#)
 Gsage::ImGuiDockspace::mStyle (C++ [member](#)), [81](#)

Gsage::ImGuiDockspace::reset (C++ function), 81
 Gsage::ImGuiDockspace::setDimensions (C++ function), 80
 Gsage::ImGuiDockspace::setState (C++ function), 80
 Gsage::ImGuiDockspace::undock (C++ function), 80
 Gsage::ImGuiDockspace::updateLayout (C++ function), 80
 Gsage::ImGuiDockspaceRenderer (C++ class), 35, 38, 51, 82
 Gsage::ImGuiDockspaceRenderer::~~ImGuiDockspaceRenderer (C++ function), 82
 Gsage::ImGuiDockspaceRenderer::activateDockspace (C++ function), 82
 Gsage::ImGuiDockspaceRenderer::begin (C++ function), 82, 83
 Gsage::ImGuiDockspaceRenderer::beginWorkspace (C++ function), 82
 Gsage::ImGuiDockspaceRenderer::dockSlots (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::dockWindows (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::end (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::endWorkspace (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::getState (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::handleDragging (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::ImGuiDockspaceRenderer (C++ function), 82
 Gsage::ImGuiDockspaceRenderer::mDockableWindows (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::mDockspace (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::mDraggedDockspace (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::mEndCommand (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::mPopClipRect (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::mPos (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::mSize (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::mStateWasUpdated (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::mStyle (C++ member), 84
 Gsage::ImGuiDockspaceRenderer::setState (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::splitters (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::tabbar (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::title (C++ function), 83
 Gsage::ImGuiDockspaceRenderer::Windows (C++ type), 83
 Gsage::ImGuiDockspaceState (C++ class), 48, 84
 Gsage::ImGuiDockspaceState::docks (C++ member), 84
 Gsage::ImGuiDockspaceState::Docks (C++ type), 84
 Gsage::ImGuiDockspaceState::Dockstate (C++ class), 48, 84
 Gsage::ImGuiDockspaceState::Dockstate::~~Dockstate (C++ function), 84
 Gsage::ImGuiDockspaceState::Dockstate::active (C++ member), 85
 Gsage::ImGuiDockspaceState::Dockstate::children (C++ member), 84
 Gsage::ImGuiDockspaceState::Dockstate::Dockstate (C++ function), 84
 Gsage::ImGuiDockspaceState::Dockstate::layout (C++ member), 84
 Gsage::ImGuiDockspaceState::Dockstate::location (C++ member), 84
 Gsage::ImGuiDockspaceState::Dockstate::next (C++ member), 84
 Gsage::ImGuiDockspaceState::Dockstate::opened (C++ member), 85
 Gsage::ImGuiDockspaceState::Dockstate::parent (C++ member), 84
 Gsage::ImGuiDockspaceState::Dockstate::prev (C++ member), 84
 Gsage::ImGuiDockspaceState::Dockstate::ratio (C++ member), 84
 Gsage::ImGuiDockspaceState::size (C++ member), 84
 Gsage::ImGuiDockspaceStyle (C++ class), 48, 85
 Gsage::ImGuiDockspaceStyle::dockSlotHoveredColor (C++ member), 85
 Gsage::ImGuiDockspaceStyle::dockSlotNormalColor (C++ member), 85
 Gsage::ImGuiDockspaceStyle::dockSlotPreview (C++ member), 85
 Gsage::ImGuiDockspaceStyle::splitterHoveredColor (C++ member), 85
 Gsage::ImGuiDockspaceStyle::splitterNormalColor (C++ member), 85
 Gsage::ImGuiDockspaceStyle::splitterThickness (C++ member), 85

Gsage::ImGuiDockspaceStyle::tabActiveCol6 (C++ member), 85	Gsage::ImGuiImage::getTexture (C++ func- tion), 87
Gsage::ImGuiDockspaceStyle::tabbarHeight (C++ member), 85	Gsage::ImGuiImage::ImGuiImage (C++ func- tion), 87
Gsage::ImGuiDockspaceStyle::tabbarPadding (C++ member), 85	Gsage::ImGuiImage::mName (C++ member), 87
Gsage::ImGuiDockspaceStyle::tabbarTextMargin (C++ member), 85	Gsage::ImGuiImage::mRender (C++ member), 87
Gsage::ImGuiDockspaceStyle::tabHoveredColor (C++ member), 85	Gsage::ImGuiImage::mTexture (C++ member), 87
Gsage::ImGuiDockspaceStyle::tabInactiveC6 (C++ member), 85	Gsage::ImGuiImage::render (C++ function), 87
Gsage::ImGuiDockspaceStyle::textColor (C++ member), 85	Gsage::ImGuiLuaInterface (C++ class), 39, 87
Gsage::ImGuiDockspaceStyle::windowBGColor (C++ member), 85	Gsage::ImGuiLuaInterface::addLuaBindings (C++ function), 87
Gsage::ImGuiDockspaceStyle::windowRoundi6 (C++ member), 85	Gsage::ImGuiManager (C++ class), 39, 89
Gsage::ImGuiDockspaceView (C++ class), 39, 51, 88	Gsage::ImGuiManager::~~ImGuiManager (C++ function), 89
Gsage::ImGuiDockspaceView::activateDock (C++ function), 89	Gsage::ImGuiManager::addRendererFactory (C++ function), 90
Gsage::ImGuiDockspaceView::addView (C++ function), 89	Gsage::ImGuiManager::doCapture (C++ func- tion), 90
Gsage::ImGuiDockspaceView::getState (C++ function), 88	Gsage::ImGuiManager::getImGuiContext (C++ function), 90
Gsage::ImGuiDockspaceView::ImGuiDockspace (C++ function), 88	Gsage::ImGuiManager::getLuaState (C++ function), 89
Gsage::ImGuiDockspaceView::mDockspace (C++ member), 89	Gsage::ImGuiManager::getType (C++ func- tion), 90
Gsage::ImGuiDockspaceView::mManager (C++ member), 89	Gsage::ImGuiManager::GlyphRanges (C++ type), 90
Gsage::ImGuiDockspaceView::mName (C++ member), 89	Gsage::ImGuiManager::handleInputEvent (C++ function), 90
Gsage::ImGuiDockspaceView::mState (C++ member), 89	Gsage::ImGuiManager::handleKeyboardEvent (C++ function), 90
Gsage::ImGuiDockspaceView::removeView (C++ function), 89	Gsage::ImGuiManager::handleMouseEvent (C++ function), 90
Gsage::ImGuiDockspaceView::render (C++ function), 88	Gsage::ImGuiManager::handleSystemChange (C++ function), 90
Gsage::ImGuiDockspaceView::setState (C++ function), 88	Gsage::ImGuiManager::ImGuiManager (C++ function), 89
Gsage::ImGuiEvent (C++ class), 39, 86	Gsage::ImGuiManager::initialize (C++ function), 89
Gsage::ImGuiEvent::~~ImGuiEvent (C++ func- tion), 86	Gsage::ImGuiManager::mContexts (C++ mem- ber), 91
Gsage::ImGuiEvent::CONTEXT_CREATED (C++ member), 86	Gsage::ImGuiManager::mCurrentDockspace (C++ member), 91
Gsage::ImGuiEvent::ImGuiEvent (C++ func- tion), 86	Gsage::ImGuiManager::mFontAtlas (C++ member), 91
Gsage::ImGuiEvent::mContextName (C++ member), 86	Gsage::ImGuiManager::mFonts (C++ member), 91
Gsage::ImGuiImage (C++ class), 39, 87	Gsage::ImGuiManager::mGlyphRanges (C++ member), 91
Gsage::ImGuiImage::~~ImGuiImage (C++ func- tion), 87	Gsage::ImGuiManager::mIsSetUp (C++ mem- ber), 91
	Gsage::ImGuiManager::mPendingSystemType (C++ member), 91

Gsage::ImGuiManager::mRenderer (C++ member), 91
 Gsage::ImGuiManager::mRendererFactories (C++ member), 91
 Gsage::ImGuiManager::mUsedRendererType (C++ member), 91
 Gsage::ImGuiManager::removeRendererFactory (C++ function), 90
 Gsage::ImGuiManager::render (C++ function), 91
 Gsage::ImGuiManager::RendererFactories (C++ type), 90
 Gsage::ImGuiManager::RendererFactory (C++ type), 89
 Gsage::ImGuiManager::renderViews (C++ function), 90
 Gsage::ImGuiManager::setLuaState (C++ function), 89
 Gsage::ImGuiManager::setUp (C++ function), 90
 Gsage::ImGuiManager::tearDown (C++ function), 90
 Gsage::ImGuiManager::TYPE (C++ member), 90
 Gsage::ImGuiMovableObject (C++ class), 39, 93
 Gsage::ImGuiMovableObject::~~ImGuiMovableObject (C++ function), 93
 Gsage::ImGuiMovableObject::getMovableType (C++ function), 93
 Gsage::ImGuiMovableObject::ImGuiMovableObject (C++ function), 93
 Gsage::ImGuiMovableObject::mDatablockName (C++ member), 93
 Gsage::ImGuiMovableObject::setDatablock (C++ function), 93
 Gsage::ImGuiMovableObject::updateVertexData (C++ function), 93
 Gsage::ImGuiMovableObjectFactory (C++ class), 40, 51, 93
 Gsage::ImGuiMovableObjectFactory::~~ImGuiMovableObjectFactory (C++ function), 94
 Gsage::ImGuiMovableObjectFactory::createInstance (C++ function), 94
 Gsage::ImGuiMovableObjectFactory::destroyInstance (C++ function), 94
 Gsage::ImGuiMovableObjectFactory::FACTORY_TYPE_NAME (C++ member), 94
 Gsage::ImGuiMovableObjectFactory::getType (C++ function), 94
 Gsage::ImGuiMovableObjectFactory::ImGuiMovableObjectFactory (C++ function), 94
 Gsage::ImGuiOgrePlugin (C++ class), 40, 94
 Gsage::ImGuiOgrePlugin::~~ImGuiOgrePlugin (C++ function), 94
 Gsage::ImGuiOgrePlugin::getName (C++ function), 96
 Gsage::ImGuiOgrePlugin::mRenderer (C++ member), 94
 Gsage::ImGuiOgrePlugin::mRendererFactories (C++ member), 94
 Gsage::ImGuiOgrePlugin::installImpl (C++ function), 94
 Gsage::ImGuiOgrePlugin::setupLuaBindings (C++ function), 94
 Gsage::ImGuiOgrePlugin::uninstallImpl (C++ function), 94
 Gsage::ImGuiOgreRenderer (C++ class), 40, 51, 95
 Gsage::ImGuiOgreRenderer::~~ImGuiOgreRenderer (C++ function), 95
 Gsage::ImGuiOgreRenderer::createFontTexture (C++ function), 95
 Gsage::ImGuiOgreRenderer::createMaterial (C++ function), 95
 Gsage::ImGuiOgreRenderer::ImGuiOgreRenderer (C++ function), 95
 Gsage::ImGuiOgreRenderer::initialize (C++ function), 95
 Gsage::ImGuiOgreRenderer::mFontPixels (C++ member), 95
 Gsage::ImGuiOgreRenderer::mFontTex (C++ member), 95
 Gsage::ImGuiOgreRenderer::mFontTexHeight (C++ member), 95
 Gsage::ImGuiOgreRenderer::mFontTexLock (C++ member), 96
 Gsage::ImGuiOgreRenderer::mFontTexWidth (C++ member), 95
 Gsage::ImGuiOgreRenderer::mSceneMgr (C++ member), 95
 Gsage::ImGuiOgreRenderer::mUpdateFontTex (C++ member), 95
 Gsage::ImGuiOgreRenderer::renderQueueEnded (C++ function), 95
 Gsage::ImGuiOgreRenderer::setImGuiContext (C++ function), 95
 Gsage::ImGuiOgreRenderer::updateFontTexture (C++ function), 95
 Gsage::ImGuiOgreRenderer::updateVertexData (C++ function), 95
 Gsage::ImGuiPlugin (C++ class), 40, 96
 Gsage::ImGuiPlugin::~~ImGuiPlugin (C++ function), 96
 Gsage::ImGuiPlugin::getName (C++ function), 96
 Gsage::ImGuiPlugin::installImpl (C++ function), 96
 Gsage::ImGuiPlugin::mUIManager (C++ member), 96

Gsage::ImGuiPlugin::mUIManagerHandle (C++ member), 96
 Gsage::ImGuiPlugin::setupLuaBindings (C++ function), 96
 Gsage::ImGuiPlugin::uninstallImpl (C++ function), 96
 Gsage::ImGUIRenderable (C++ class), 38
 Gsage::ImGuiRenderable (C++ class), 40
 Gsage::ImGuiRenderer (C++ class), 40, 51, 91
 Gsage::ImGuiRenderer::~~ImGuiRenderer (C++ function), 91
 Gsage::ImGuiRenderer::Context (C++ class), 48, 92
 Gsage::ImGuiRenderer::Context::context (C++ member), 92
 Gsage::ImGuiRenderer::Context::size (C++ member), 92
 Gsage::ImGuiRenderer::createFontTexture (C++ function), 91
 Gsage::ImGuiRenderer::getContext (C++ function), 92
 Gsage::ImGuiRenderer::ImGuiRenderer (C++ function), 91
 Gsage::ImGuiRenderer::initialize (C++ function), 91
 Gsage::ImGuiRenderer::initializeContext (C++ function), 92
 Gsage::ImGuiRenderer::mContextLock (C++ member), 92
 Gsage::ImGuiRenderer::mContextNames (C++ member), 92
 Gsage::ImGuiRenderer::mContexts (C++ member), 92
 Gsage::ImGuiRenderer::mEngine (C++ member), 92
 Gsage::ImGuiRenderer::mManager (C++ member), 92
 Gsage::ImGuiRenderer::mMousePositions (C++ member), 92
 Gsage::ImGuiRenderer::mRenderTargetWhitelist (C++ member), 92
 Gsage::ImGuiRenderer::render (C++ function), 92
 Gsage::ImGuiRenderer::setImGuiContext (C++ function), 92
 Gsage::ImGuiRenderer::setMousePosition (C++ function), 91
 Gsage::ImGuiRendererV1 (C++ class), 40, 96
 Gsage::ImGuiRendererV1::~~ImGuiRendererV1 (C++ function), 96
 Gsage::ImGuiRendererV1::createMaterial (C++ function), 97
 Gsage::ImGuiRendererV1::ImGuiRendererV1 (C++ function), 96
 Gsage::ImGuiRendererV1::initialize (C++ function), 96
 Gsage::ImGuiRendererV1::mPass (C++ member), 97
 Gsage::ImGuiRendererV1::mRenderables (C++ member), 97
 Gsage::ImGuiRendererV1::mTexUnit (C++ member), 97
 Gsage::ImGuiRendererV1::setFiltering (C++ function), 97
 Gsage::ImGuiRendererV1::updateFontTexture (C++ function), 97
 Gsage::ImGuiRendererV1::updateVertexData (C++ function), 97
 Gsage::ImGuiRendererV2 (C++ class), 40, 97
 Gsage::ImGuiRendererV2::~~ImGuiRendererV2 (C++ function), 97
 Gsage::ImGuiRendererV2::createImGuiMovableObject (C++ function), 97
 Gsage::ImGuiRendererV2::createMaterial (C++ function), 97
 Gsage::ImGuiRendererV2::ImGuiRendererV2 (C++ function), 97
 Gsage::ImGuiRendererV2::initialize (C++ function), 97
 Gsage::ImGuiRendererV2::mHlms (C++ member), 98
 Gsage::ImGuiRendererV2::mImGuiMovableObjects (C++ member), 98
 Gsage::ImGuiRendererV2::mMovableObjectFactory (C++ member), 98
 Gsage::ImGuiRendererV2::mRenderQueueGroup (C++ member), 98
 Gsage::ImGuiRendererV2::updateVertexData (C++ function), 97
 Gsage::ImGuiTextBuffer (C++ class), 40, 51, 87
 Gsage::ImGuiTextBuffer::~~ImGuiTextBuffer (C++ function), 88
 Gsage::ImGuiTextBuffer::ImGuiTextBuffer (C++ function), 88
 Gsage::ImGuiTextBuffer::mBuffer (C++ member), 88
 Gsage::ImGuiTextBuffer::mSize (C++ member), 88
 Gsage::ImGuiTextBuffer::read (C++ function), 88
 Gsage::ImGuiTextBuffer::size (C++ function), 88
 Gsage::ImGuiTextBuffer::write (C++ function), 88
 Gsage::ImGuiViewCollection (C++ class), 40, 51, 92
 Gsage::ImGuiViewCollection::addView (C++ function), 92

Gsage::ImGuiViewCollection::mViews (C++ member), 93
 Gsage::ImGuiViewCollection::removeView (C++ function), 93
 Gsage::ImGuiViewCollection::RenderView (C++ type), 93
 Gsage::ImGuiViewCollection::Views (C++ type), 93
 Gsage::IMovableObjectWrapper (C++ class), 38, 51, 112
 Gsage::IMovableObjectWrapper::getRenderQueueGroup (C++ function), 112
 Gsage::IMovableObjectWrapper::resetVisibilityFlags (C++ function), 112
 Gsage::IMovableObjectWrapper::setRenderQueueGroup (C++ function), 112
 Gsage::IMovableObjectWrapper::setVisibilityFlags (C++ function), 112
 Gsage::ItemWrapper (C++ class), 41, 98
 Gsage::ItemWrapper::~~ItemWrapper (C++ function), 98
 Gsage::ItemWrapper::getDatablock (C++ function), 99
 Gsage::ItemWrapper::getItem (C++ function), 99
 Gsage::ItemWrapper::getMesh (C++ function), 98
 Gsage::ItemWrapper::getQueryFlags (C++ function), 99
 Gsage::ItemWrapper::ItemWrapper (C++ function), 98
 Gsage::ItemWrapper::mDatablock (C++ member), 99
 Gsage::ItemWrapper::mMeshName (C++ member), 99
 Gsage::ItemWrapper::mQuery (C++ member), 99
 Gsage::ItemWrapper::mQueryString (C++ member), 99
 Gsage::ItemWrapper::setDatablock (C++ function), 99
 Gsage::ItemWrapper::setMesh (C++ function), 98
 Gsage::ItemWrapper::setQueryFlags (C++ function), 98
 Gsage::ItemWrapper::TYPE (C++ member), 99
 Gsage::Layout (C++ enum), 76
 Gsage::Left (C++ enumerator), 77
 Gsage::LightWrapper (C++ class), 41, 99
 Gsage::LightWrapper::~~LightWrapper (C++ function), 99
 Gsage::LightWrapper::create (C++ function), 99
 Gsage::LightWrapper::getCastShadows (C++ function), 100
 Gsage::LightWrapper::getDiffuseColour (C++ function), 100
 Gsage::LightWrapper::getDirection (C++ function), 100
 Gsage::LightWrapper::getName (C++ function), 99
 Gsage::LightWrapper::getPosition (C++ function), 100
 Gsage::LightWrapper::getRenderQueueGroup (C++ function), 100
 Gsage::LightWrapper::getSpecularColour (C++ function), 100
 Gsage::LightWrapper::getType (C++ function), 100
 Gsage::LightWrapper::LightWrapper (C++ function), 99
 Gsage::LightWrapper::mapType (C++ function), 101
 Gsage::LightWrapper::setCastShadows (C++ function), 100
 Gsage::LightWrapper::setDiffuseColour (C++ function), 100
 Gsage::LightWrapper::setDirection (C++ function), 100
 Gsage::LightWrapper::setPosition (C++ function), 100
 Gsage::LightWrapper::setRenderQueueGroup (C++ function), 100
 Gsage::LightWrapper::setSpecularColour (C++ function), 100
 Gsage::LightWrapper::setType (C++ function), 99
 Gsage::LightWrapper::TYPE (C++ member), 100
 Gsage::loadState (C++ function), 49, 76
 Gsage::Location (C++ enum), 77
 Gsage::ManualObjectWrapper (C++ class), 41
 Gsage::ManualTextureManager (C++ class), 41, 51, 105
 Gsage::ManualTextureManager::~~ManualTextureManager (C++ function), 105
 Gsage::ManualTextureManager::createTexture (C++ function), 105
 Gsage::ManualTextureManager::deleteTexture (C++ function), 105
 Gsage::ManualTextureManager::getTexture (C++ function), 105
 Gsage::ManualTextureManager::ManualTextureManager (C++ function), 105
 Gsage::ManualTextureManager::mPixelFormat (C++ member), 106
 Gsage::ManualTextureManager::mRenderSystem (C++ member), 106

Gsage::ManualTextureManager::mRenderSystem (C++ member), 106
 Gsage::ManualTextureManager::mTextures (C++ member), 106
 Gsage::ManualTextureManager::RenderSystem (C++ type), 106
 Gsage::ManualTextureManager::reset (C++ function), 105
 Gsage::ManualTextureManager::setDefaultPixelFormat (C++ function), 106
 Gsage::ManualTextureManager::updateDirtyTextures (C++ function), 106
 Gsage::MaterialBuilder (C++ class), 41, 109
 Gsage::MaterialBuilder::~~MaterialBuilder (C++ function), 109
 Gsage::MaterialBuilder::MaterialBuilder (C++ function), 109
 Gsage::MaterialBuilder::parse (C++ function), 109
 Gsage::MaterialLoader (C++ class), 41, 51, 109
 Gsage::MaterialLoader::FileInfo (C++ class), 48, 110
 Gsage::MaterialLoader::FileInfo::folder (C++ member), 110
 Gsage::MaterialLoader::FileInfo::modified (C++ member), 110
 Gsage::MaterialLoader::FileInfo::path (C++ member), 110
 Gsage::MaterialLoader::load (C++ function), 110
 Gsage::MaterialLoader::MaterialIndex (C++ type), 110
 Gsage::MaterialLoader::MaterialLoader (C++ function), 110
 Gsage::MaterialLoader::mFacade (C++ member), 110
 Gsage::MaterialLoader::mIndex (C++ member), 110
 Gsage::MaterialLoader::mMaterialIndex (C++ member), 110
 Gsage::MaterialLoader::mRender (C++ member), 110
 Gsage::MaterialLoader::mWorkdir (C++ member), 110
 Gsage::MaterialLoader::onEnvUpdated (C++ function), 110
 Gsage::MaterialLoader::readIndexFile (C++ function), 110
 Gsage::MaterialLoader::reloadIndex (C++ function), 110
 Gsage::MaterialLoader::scan (C++ function), 110
 Gsage::MaterialLoader::scanFolder (C++ function), 110
 Gsage::MaterialLoader::writeIndexFile (C++ function), 110
 Gsage::MovableObjectWrapper (C++ class), 41, 112
 Gsage::MovableObjectWrapper::~~MovableObjectWrapper (C++ function), 113
 Gsage::MovableObjectWrapper::defineUserBindings (C++ function), 113
 Gsage::MovableObjectWrapper::getRenderQueueGroup (C++ function), 113
 Gsage::MovableObjectWrapper::mObject (C++ member), 113
 Gsage::MovableObjectWrapper::MovableObjectWrapper (C++ function), 113
 Gsage::MovableObjectWrapper::resetVisibilityFlags (C++ function), 113
 Gsage::MovableObjectWrapper::setRenderQueueGroup (C++ function), 113
 Gsage::MovableObjectWrapper::setVisibilityFlags (C++ function), 113
 Gsage::NONE (C++ enumerator), 120
 Gsage::None (C++ enumerator), 76
 Gsage::ObjectMutation (C++ class), 41, 113
 Gsage::ObjectMutation::~~ObjectMutation (C++ function), 113
 Gsage::ObjectMutation::Callback (C++ type), 113
 Gsage::ObjectMutation::execute (C++ function), 113
 Gsage::ObjectMutation::mCallback (C++ member), 114
 Gsage::ObjectMutation::ObjectMutation (C++ function), 113
 Gsage::OgreGeom (C++ class), 41, 115
 Gsage::OgreGeom::~~OgreGeom (C++ function), 115
 Gsage::OgreGeom::mSrcEntities (C++ member), 115
 Gsage::OgreGeom::OgreEntities (C++ type), 115
 Gsage::OgreGeom::OgreGeom (C++ function), 115
 Gsage::OgreLogRedirect (C++ class), 42, 52, 124
 Gsage::OgreLogRedirect::messageLogged (C++ function), 124
 Gsage::OgreObject (C++ class), 42, 52, 118
 Gsage::OgreObject::~~OgreObject (C++ function), 118
 Gsage::OgreObject::attach (C++ function), 119
 Gsage::OgreObject::attachObject (C++ function), 119
 Gsage::OgreObject::destroy (C++ function),

119

Gsage::OgreObject::generateName (C++ function), 119

Gsage::OgreObject::getObjectId (C++ function), 119

Gsage::OgreObject::getType (C++ function), 119

Gsage::OgreObject::initialize (C++ function), 118

Gsage::OgreObject::mAttachParams (C++ member), 119

Gsage::OgreObject::mObjectId (C++ member), 119

Gsage::OgreObject::mObjectManager (C++ member), 119

Gsage::OgreObject::mOwnerId (C++ member), 119

Gsage::OgreObject::mParentNode (C++ member), 119

Gsage::OgreObject::mParentObject (C++ member), 119

Gsage::OgreObject::mSceneManager (C++ member), 119

Gsage::OgreObject::mType (C++ member), 119

Gsage::OgreObject::OgreObject (C++ function), 118

Gsage::OgreObject::PendingPropertyUpdateGsage::OgreObjectManagerEvent (C++ class), 48, 119

Gsage::OgreObject::PendingPropertyUpdateGsage::OgreObjectManagerEvent (C++ member), 120

Gsage::OgreObject::sync (C++ function), 119

Gsage::OgreObjectManager (C++ class), 42, 115

Gsage::OgreObjectManager::~~OgreObjectManager (C++ function), 115

Gsage::OgreObjectManager::ConcreteOgreObjectPool (C++ class), 42, 117

Gsage::OgreObjectManager::ConcreteOgreObjectPool (C++ function), 117

Gsage::OgreObjectManager::ConcreteOgreObjectPool (C++ function), 117

Gsage::OgreObjectManager::ConcreteOgreObjectPool (C++ function), 117

Gsage::OgreObjectManager::create (C++ function), 115, 116

Gsage::OgreObjectManager::destroy (C++ function), 116

Gsage::OgreObjectManager::getRenderSystem (C++ function), 116

Gsage::OgreObjectManager::mObjects (C++ member), 117

Gsage::OgreObjectManager::mRenderSystem (C++ member), 117

Gsage::OgreObjectManager::OgreObjectManager (C++ function), 115

Gsage::OgreObjectManager::OgreObjectPool (C++ class), 42, 117

Gsage::OgreObjectManager::OgreObjectPool::~~OgreObjectPool (C++ function), 117

Gsage::OgreObjectManager::OgreObjectPool::allocate (C++ function), 117

Gsage::OgreObjectManager::OgreObjectPool::remove (C++ function), 117

Gsage::OgreObjectManager::OgreObjectsCollections (C++ type), 117

Gsage::OgreObjectManager::registerElement (C++ function), 116

Gsage::OgreObjectManager::unregisterElement (C++ function), 116

Gsage::OgreObjectManagerEvent (C++ class), 42, 52, 117

Gsage::OgreObjectManagerEvent::~~OgreObjectManagerEvent (C++ function), 117

Gsage::OgreObjectManagerEvent::FACTORY_UNREGISTERED (C++ member), 118

Gsage::OgreObjectManagerEvent::getId (C++ function), 117

Gsage::OgreObjectManagerEvent::getObject (C++ function), 117

Gsage::OgreObjectManagerEvent::mId (C++ member), 118

Gsage::OgreObjectManagerEvent::mObject (C++ member), 118

Gsage::OgreObjectManagerEvent::OBJECT_DESTROYED (C++ member), 118

Gsage::OgreObjectManagerEvent::OgreObjectManagerEvent (C++ function), 117

Gsage::OgreObjectManagerEvent::OgreQuaternion (C++ function), 49, 114

Gsage::OgreObjectManagerEvent::OgreRenderComponent (C++ class), 42, 52, 120

Gsage::OgreObjectManagerEvent::OgreRenderComponent::~~OgreRenderComponent (C++ function), 120

Gsage::OgreObjectManagerEvent::OgreRenderComponent::adjustAnimationStateSpace (C++ function), 122

Gsage::OgreObjectManagerEvent::OgreRenderComponent::getAnimations (C++ function), 123

Gsage::OgreObjectManagerEvent::OgreRenderComponent::getDirection (C++ function), 122

Gsage::OgreObjectManagerEvent::OgreRenderComponent::getFaceOrientation (C++ function), 122

Gsage::OgreObjectManagerEvent::OgreRenderComponent::getOgreDirection (C++ function), 121

Gsage::OgreObjectManagerEvent::OgreRenderComponent::getOgreFaceOrientation (C++ function), 121

Gsage::OgreRenderComponent::getOgreOrientation (C++ function), 121
 Gsage::OgreRenderComponent::getOgrePosition (C++ function), 121
 Gsage::OgreRenderComponent::getOgreScale (C++ function), 121
 Gsage::OgreRenderComponent::getOrientation (C++ function), 122
 Gsage::OgreRenderComponent::getPosition (C++ function), 122
 Gsage::OgreRenderComponent::getResources (C++ function), 123
 Gsage::OgreRenderComponent::getRoot (C++ function), 123
 Gsage::OgreRenderComponent::getRootNode (C++ function), 122
 Gsage::OgreRenderComponent::getScale (C++ function), 122
 Gsage::OgreRenderComponent::lookAt (C++ function), 121
 Gsage::OgreRenderComponent::mAddedToScene (C++ member), 123
 Gsage::OgreRenderComponent::mAnimationScheduler (C++ member), 123
 Gsage::OgreRenderComponent::mObjectManager (C++ member), 123
 Gsage::OgreRenderComponent::mResourceManager (C++ member), 123
 Gsage::OgreRenderComponent::mResources (C++ member), 123
 Gsage::OgreRenderComponent::mRootNode (C++ member), 123
 Gsage::OgreRenderComponent::mSceneManager (C++ member), 123
 Gsage::OgreRenderComponent::OgreRenderComponent (C++ function), 120
 Gsage::OgreRenderComponent::playAnimation (C++ function), 122
 Gsage::OgreRenderComponent::POSITION_CHANGE (C++ member), 123
 Gsage::OgreRenderComponent::prepare (C++ function), 120
 Gsage::OgreRenderComponent::resetAnimationState (C++ function), 122
 Gsage::OgreRenderComponent::rotate (C++ function), 121
 Gsage::OgreRenderComponent::setAnimations (C++ function), 123
 Gsage::OgreRenderComponent::setAnimationState (C++ function), 122
 Gsage::OgreRenderComponent::setOrientation (C++ function), 121, 122
 Gsage::OgreRenderComponent::setPosition (C++ function), 120, 121
 Gsage::OgreRenderComponent::setResources (C++ function), 123
 Gsage::OgreRenderComponent::setRootNode (C++ function), 122
 Gsage::OgreRenderComponent::SYSTEM (C++ member), 123
 Gsage::OgreRenderSystem (C++ class), 42, 124
 Gsage::OgreRenderSystem::~~OgreRenderSystem (C++ function), 124
 Gsage::OgreRenderSystem::allowMultithreading (C++ function), 128
 Gsage::OgreRenderSystem::ComponentLoadQueue (C++ type), 128
 Gsage::OgreRenderSystem::configure (C++ function), 125
 Gsage::OgreRenderSystem::createRenderTarget (C++ function), 127
 Gsage::OgreRenderSystem::createTexture (C++ function), 126
 Gsage::OgreRenderSystem::deleteTexture (C++ function), 126
 Gsage::OgreRenderSystem::Entities (C++ type), 124
 Gsage::OgreRenderSystem::fillComponentData (C++ function), 125
 Gsage::OgreRenderSystem::getConfig (C++ function), 125
 Gsage::OgreRenderSystem::getEntities (C++ function), 126
 Gsage::OgreRenderSystem::getGeometry (C++ function), 125, 128
 Gsage::OgreRenderSystem::getHeight (C++ function), 127
 Gsage::OgreRenderSystem::getMainRenderTarget (C++ function), 128
 Gsage::OgreRenderSystem::getMaterialLoader (C++ function), 128
 Gsage::OgreRenderSystem::getObjectManager (C++ function), 128
 Gsage::OgreRenderSystem::getObjectsInRadius (C++ function), 127
 Gsage::OgreRenderSystem::getRenderSystem (C++ function), 126
 Gsage::OgreRenderSystem::getRenderTarget (C++ function), 127, 128
 Gsage::OgreRenderSystem::getRenderWindow (C++ function), 126
 Gsage::OgreRenderSystem::getSceneManager (C++ function), 126
 Gsage::OgreRenderSystem::getTexture (C++ function), 126
 Gsage::OgreRenderSystem::getWidth (C++ function), 127
 Gsage::OgreRenderSystem::handleWindowResized

(C++ function), 128

Gsage::OgreRenderSystem::ID (C++ member), 128

Gsage::OgreRenderSystem::initialize (C++ function), 124

Gsage::OgreRenderSystem::installPlugin (C++ function), 128

Gsage::OgreRenderSystem::mFontManager (C++ member), 129

Gsage::OgreRenderSystem::mLoadQueue (C++ member), 129

Gsage::OgreRenderSystem::mLogManager (C++ member), 129

Gsage::OgreRenderSystem::mLogRedirect (C++ member), 129

Gsage::OgreRenderSystem::mManualMovableTextureManager (C++ member), 129

Gsage::OgreRenderSystem::mManualTextureManager (C++ member), 129

Gsage::OgreRenderSystem::mMaterialLoader (C++ member), 129

Gsage::OgreRenderSystem::mMutationQueue (C++ member), 129

Gsage::OgreRenderSystem::mObjectManager (C++ member), 129

Gsage::OgreRenderSystem::mRenderSystem (C++ member), 129

Gsage::OgreRenderSystem::mRenderTargetFactory (C++ member), 129

Gsage::OgreRenderSystem::mRenderTargets (C++ member), 129

Gsage::OgreRenderSystem::mRenderTargetsReverseIndex (C++ member), 129

Gsage::OgreRenderSystem::mRenderWindowByHandle (C++ member), 129

Gsage::OgreRenderSystem::mResourceManager (C++ member), 129

Gsage::OgreRenderSystem::mRoot (C++ member), 129

Gsage::OgreRenderSystem::mSceneManager (C++ member), 129

Gsage::OgreRenderSystem::mViewport (C++ member), 129

Gsage::OgreRenderSystem::mWindow (C++ member), 129

Gsage::OgreRenderSystem::mWindowEventListener (C++ member), 129

Gsage::OgreRenderSystem::OgreEntities (C++ type), 124

Gsage::OgreRenderSystem::OgreRenderSystem (C++ function), 124

Gsage::OgreRenderSystem::prepareComponent (C++ function), 125

Gsage::OgreRenderSystem::queueMutation (C++ function), 128

Gsage::OgreRenderSystem::registerElement (C++ function), 126

Gsage::OgreRenderSystem::removeAllRenderTargets (C++ function), 128

Gsage::OgreRenderSystem::removeComponent (C++ function), 125

Gsage::OgreRenderSystem::renderCameraToTarget (C++ function), 127

Gsage::OgreRenderSystem::renderQueueEnded (C++ function), 126

Gsage::OgreRenderSystem::renderQueueStarted (C++ function), 126

Gsage::OgreRenderSystem::RenderTargetReverseIndex (C++ type), 128

Gsage::OgreRenderSystem::RenderTargets (C++ type), 128

Gsage::OgreRenderSystem::RenderWindowsByHandle (C++ type), 128

Gsage::OgreRenderSystem::setHeight (C++ function), 127

Gsage::OgreRenderSystem::setSize (C++ function), 127

Gsage::OgreRenderSystem::setWidth (C++ function), 127

Gsage::OgreRenderSystem::shutdown (C++ function), 125

Gsage::OgreRenderSystem::unregisterElement (C++ function), 126

Gsage::OgreRenderSystem::update (C++ function), 125

Gsage::OgreRenderSystem::updateComponent (C++ function), 125

Gsage::OgreSelectEvent (C++ class), 42, 129

Gsage::OgreSelectEvent::~OgreSelectEvent (C++ function), 129

Gsage::OgreSelectEvent::getIntersection (C++ function), 129

Gsage::OgreSelectEvent::mIntersection (C++ member), 129

Gsage::OgreSelectEvent::OgreSelectEvent (C++ function), 129

Gsage::OgreTexture (C++ class), 42, 52, 106

Gsage::OgreTexture::~OgreTexture (C++ function), 106

Gsage::OgreTexture::AllocateScalingPolicy (C++ class), 43, 108

Gsage::OgreTexture::AllocateScalingPolicy::AllocateScalingPolicy (C++ function), 108

Gsage::OgreTexture::AllocateScalingPolicy::mScalingPolicy (C++ member), 108

Gsage::OgreTexture::AllocateScalingPolicy::resize (C++ function), 108

Gsage::OgreTexture::blitAll (C++ function), 106

107

Gsage::OgreTexture::blitDirty (C++ function), 107

Gsage::OgreTexture::create (C++ function), 107

Gsage::OgreTexture::createScalingPolicy (C++ function), 107

Gsage::OgreTexture::DefaultScalingPolicy (C++ class), 43, 108

Gsage::OgreTexture::DefaultScalingPolicy::DefaultScalingPolicy (C++ function), 108

Gsage::OgreTexture::DefaultScalingPolicy::GetSize (C++ function), 107

Gsage::OgreTexture::DefaultScalingPolicy::GetSize (C++ function), 107

Gsage::OgreTexture::destroy (C++ function), 107

Gsage::OgreTexture::getOgreTexture (C++ function), 107

Gsage::OgreTexture::hasData (C++ function), 107

Gsage::OgreTexture::isDirty (C++ function), 107

Gsage::OgreTexture::mCreate (C++ member), 107

Gsage::OgreTexture::mDirty (C++ member), 107

Gsage::OgreTexture::mDirtyRegions (C++ member), 108

Gsage::OgreTexture::mFlags (C++ member), 108

Gsage::OgreTexture::mHasData (C++ member), 107

Gsage::OgreTexture::mName (C++ member), 107

Gsage::OgreTexture::mScalingPolicy (C++ member), 107

Gsage::OgreTexture::mTexture (C++ member), 107

Gsage::OgreTexture::OgreTexture (C++ function), 106

Gsage::OgreTexture::render (C++ function), 107

Gsage::OgreTexture::ScalingPolicy (C++ class), 43, 108

Gsage::OgreTexture::ScalingPolicy::~ScalingPolicy (C++ function), 108

Gsage::OgreTexture::ScalingPolicy::invalidate (C++ function), 108

Gsage::OgreTexture::ScalingPolicy::mDirty (C++ member), 109

Gsage::OgreTexture::ScalingPolicy::mHeight (C++ member), 109

Gsage::OgreTexture::ScalingPolicy::mTexture (C++ member), 109

Gsage::OgreTexture::ScalingPolicy::mWidth (C++ member), 109

Gsage::OgreTexture::ScalingPolicy::render (C++ member), 109

Gsage::OgreTexture::ScalingPolicy::resize (C++ function), 109

Gsage::OgreTexture::ScalingPolicy::ScalingPolicy (C++ function), 108

Gsage::OgreTexture::ScalingPolicy::update (C++ function), 108

Gsage::OgreTexture::unloadingComplete (C++ function), 107

Gsage::OgreVector3ToGsageVector3 (C++ function), 49, 114

Gsage::OgreView (C++ class), 34, 43, 52, 130

Gsage::OgreView::~OgreView (C++ function), 130

Gsage::OgreView::mBgColour (C++ member), 131

Gsage::OgreView::mHeight (C++ member), 131

Gsage::OgreView::mPosition (C++ member), 131

Gsage::OgreView::mRender (C++ member), 131

Gsage::OgreView::mTexture (C++ member), 131

Gsage::OgreView::mTextureID (C++ member), 131

Gsage::OgreView::mViewport (C++ member), 131

Gsage::OgreView::mWidth (C++ member), 131

Gsage::OgreView::OgreView (C++ function), 130

Gsage::OgreView::onTextureEvent (C++ function), 130

Gsage::OgreView::render (C++ function), 130

Gsage::OgreView::setTexture (C++ function), 130

Gsage::OgreView::setTextureID (C++ function), 130

Gsage::OverlayPass (C++ class), 43, 131

Gsage::OverlayPass::~OverlayPass (C++ function), 131

Gsage::OverlayPass::execute (C++ function), 131

Gsage::OverlayPass::OverlayPass (C++ function), 131

Gsage::OverlayPassDef (C++ class), 43, 131

Gsage::OverlayPassDef::~OverlayPassDef (C++ function), 131

Gsage::OverlayPassDef::OverlayPassDef (C++ function), 131

[Gsage::ParticleSystemWrapper \(C++ class\), 43, 131](#)
[Gsage::ParticleSystemWrapper::~ParticleSystemWrapper \(C++ function\), 132](#)
[Gsage::ParticleSystemWrapper::createParticleSystemWrapper \(C++ function\), 132](#)
[Gsage::ParticleSystemWrapper::getTemplate \(C++ function\), 132](#)
[Gsage::ParticleSystemWrapper::mParticleSystemWrapper \(C++ member\), 133](#)
[Gsage::ParticleSystemWrapper::mTemplate \(C++ member\), 133](#)
[Gsage::ParticleSystemWrapper::ParticleSystemWrapper \(C++ function\), 132](#)
[Gsage::ParticleSystemWrapper::read \(C++ function\), 132](#)
[Gsage::ParticleSystemWrapper::setTemplate \(C++ function\), 132](#)
[Gsage::ParticleSystemWrapper::TYPE \(C++ member\), 132](#)
[Gsage::Query \(C++ enum\), 69, 98](#)
[Gsage::Renderer \(C++ class\), 44, 52, 140](#)
[Gsage::Renderer::mCore \(C++ member\), 140](#)
[Gsage::Renderer::mWindow \(C++ member\), 140](#)
[Gsage::Renderer::render \(C++ function\), 140](#)
[Gsage::Renderer::Renderer \(C++ function\), 140](#)
[Gsage::RendererFactory \(C++ class\), 44, 71](#)
[Gsage::RendererFactory::create \(C++ function\), 71](#)
[Gsage::RendererPtr \(C++ type\), 49, 140](#)
[Gsage::RenderEvent \(C++ class\), 44, 52, 133](#)
[Gsage::RenderEvent::~~RenderEvent \(C++ function\), 133](#)
[Gsage::RenderEvent::getRenderSystem \(C++ function\), 133](#)
[Gsage::RenderEvent::mRenderSystem \(C++ member\), 133](#)
[Gsage::RenderEvent::queueID \(C++ member\), 133](#)
[Gsage::RenderEvent::RENDER_QUEUE_ENDED \(C++ member\), 133](#)
[Gsage::RenderEvent::RENDER_QUEUE_STARTED \(C++ member\), 133](#)
[Gsage::RenderEvent::RenderEvent \(C++ function\), 133](#)
[Gsage::RenderEvent::renderTarget \(C++ member\), 133](#)
[Gsage::RenderEvent::UPDATE \(C++ member\), 133](#)
[Gsage::RenderSystemWrapper \(C++ class\), 44, 52, 142](#)
[Gsage::RenderSystemWrapper::~~RenderSystemWrapper \(C++ function\), 142](#)
[Gsage::RenderSystemWrapper::Contexts \(C++ type\), 142](#)
[Gsage::RenderSystemWrapper::createContext \(C++ function\), 143](#)
[Gsage::RenderSystemWrapper::destroy \(C++ function\), 142](#)
[Gsage::RenderSystemWrapper::getContext \(C++ function\), 142](#)
[Gsage::RenderSystemWrapper::getContexts \(C++ function\), 142](#)
[Gsage::RenderSystemWrapper::mContexts \(C++ member\), 143](#)
[Gsage::RenderSystemWrapper::mEngine \(C++ member\), 143](#)
[Gsage::RenderSystemWrapper::mInitialized \(C++ member\), 143](#)
[Gsage::RenderSystemWrapper::mLuaState \(C++ member\), 143](#)
[Gsage::RenderSystemWrapper::RenderSystemWrapper \(C++ function\), 142](#)
[Gsage::RenderSystemWrapper::setLuaState \(C++ function\), 142](#)
[Gsage::RenderSystemWrapper::setUpInterfaces \(C++ function\), 143](#)
[Gsage::RenderTarget \(C++ class\), 44, 52, 135](#)
[Gsage::RenderTarget::~~RenderTarget \(C++ function\), 136](#)
[Gsage::RenderTarget::configureViewport \(C++ function\), 137](#)
[Gsage::RenderTarget::doRaycasting \(C++ function\), 138](#)
[Gsage::RenderTarget::getCamera \(C++ function\), 136](#)
[Gsage::RenderTarget::getHeight \(C++ function\), 136](#)
[Gsage::RenderTarget::getName \(C++ function\), 136](#)
[Gsage::RenderTarget::getOgreRenderTarget \(C++ function\), 137](#)
[Gsage::RenderTarget::getRay \(C++ function\), 138](#)
[Gsage::RenderTarget::getType \(C++ function\), 136](#)
[Gsage::RenderTarget::getViewport \(C++ function\), 137](#)
[Gsage::RenderTarget::getWidth \(C++ function\), 136](#)
[Gsage::RenderTarget::handleMouseEvent \(C++ function\), 138](#)
[Gsage::RenderTarget::handleRaycast \(C++ function\), 138](#)
[Gsage::RenderTarget::initialize \(C++ function\), 136](#)
[Gsage::RenderTarget::isAutoUpdated \(C++ function\), 136](#)

function), 137

Gsage::RenderTarget::isMouseOver (C++ function), 137

Gsage::RenderTarget::mAutoUpdate (C++ member), 137

Gsage::RenderTarget::mCollisionTools (C++ member), 138

Gsage::RenderTarget::mContinuousRaycast (C++ member), 138

Gsage::RenderTarget::mCurrentCamera (C++ member), 137

Gsage::RenderTarget::mDefaultCamera (C++ member), 137

Gsage::RenderTarget::mDestroying (C++ member), 138

Gsage::RenderTarget::mEngine (C++ member), 138

Gsage::RenderTarget::mHasQueueSequence (C++ member), 137

Gsage::RenderTarget::mHeight (C++ member), 137

Gsage::RenderTarget::mMouseOver (C++ member), 138

Gsage::RenderTarget::mMousePosition (C++ member), 138

Gsage::RenderTarget::mName (C++ member), 137

Gsage::RenderTarget::mParameters (C++ member), 137

Gsage::RenderTarget::mRenderQueueSequenceCreation (C++ member), 138

Gsage::RenderTarget::mRolledOverObject (C++ member), 138

Gsage::RenderTarget::mSamples (C++ member), 138

Gsage::RenderTarget::mSceneManager (C++ member), 138

Gsage::RenderTarget::mType (C++ member), 137

Gsage::RenderTarget::mWidth (C++ member), 137

Gsage::RenderTarget::mWrappedTarget (C++ member), 137

Gsage::RenderTarget::mX (C++ member), 137

Gsage::RenderTarget::mY (C++ member), 137

Gsage::RenderTarget::onTextureEvent (C++ function), 137

Gsage::RenderTarget::raycast (C++ function), 137

Gsage::RenderTarget::renderQueueSequenceName (C++ member), 137

Gsage::RenderTarget::RenderTarget (C++ function), 136

Gsage::RenderTarget::setCamera (C++ function), 136

Gsage::RenderTarget::setDimensions (C++ function), 136

Gsage::RenderTarget::setHeight (C++ function), 136

Gsage::RenderTarget::setPosition (C++ function), 136

Gsage::RenderTarget::setWidth (C++ function), 136

Gsage::RenderTarget::subscribe (C++ function), 138

Gsage::RenderTarget::switchToDefaultCamera (C++ function), 137

Gsage::RenderTarget::update (C++ function), 137

Gsage::RenderTarget::updateCameraAspectRatio (C++ function), 137

Gsage::RenderTargetFactory (C++ class), 44, 52, 138

Gsage::RenderTargetFactory::create (C++ function), 138

Gsage::RenderTargetFactory::wrap (C++ function), 138

Gsage::RenderTargetPtr (C++ type), 49, 135

Gsage::RenderTargetType (C++ class), 44, 135

Gsage::ResourceManager (C++ class), 44, 140

Gsage::ResourceManager::~ResourceManager (C++ function), 140

Gsage::ResourceManager::load (C++ function), 140

Gsage::ResourceManager::mFacade (C++ member), 141

Gsage::ResourceManager::mHlmsLoaded (C++ member), 141

Gsage::ResourceManager::mMaterialLoader (C++ member), 141

Gsage::ResourceManager::processPath (C++ function), 141

Gsage::ResourceManager::ResourceManager (C++ function), 140

Gsage::ResourceManager::unload (C++ function), 140

Gsage::Right (C++ enumerator), 77

Gsage::RocketContextEvent (C++ class), 44, 141

Gsage::RocketContextEvent::~~RocketContextEvent (C++ function), 141

Gsage::RocketContextEvent::CREATE (C++ member), 141

Gsage::RocketContextEvent::name (C++ member), 141

Gsage::RocketContextEvent::RocketContextEvent (C++ function), 141

Gsage::RocketOgreWrapper (C++ class), 44, 141

[Gsage::RocketOgreWrapper::~~RocketOgreWrapper \(C++ function\), 141](#)
[Gsage::RocketOgreWrapper::configureRenderSystem \(C++ function\), 142](#)
[Gsage::RocketOgreWrapper::destroy \(C++ function\), 141](#)
[Gsage::RocketOgreWrapper::mRenderInterface \(C++ member\), 142](#)
[Gsage::RocketOgreWrapper::mSystemInterface \(C++ member\), 142](#)
[Gsage::RocketOgreWrapper::render \(C++ function\), 142](#)
[Gsage::RocketOgreWrapper::RocketOgreWrapper \(C++ function\), 141](#)
[Gsage::RocketOgreWrapper::setUpInterface \(C++ function\), 142](#)
[Gsage::RocketUIManager \(C++ class\), 45, 143](#)
[Gsage::RocketUIManager::~~RocketUIManager \(C++ function\), 143](#)
[Gsage::RocketUIManager::buildKeyMap \(C++ function\), 144](#)
[Gsage::RocketUIManager::doCapture \(C++ function\), 144](#)
[Gsage::RocketUIManager::getKeyModifierState \(C++ function\), 144](#)
[Gsage::RocketUIManager::getLuaState \(C++ function\), 143](#)
[Gsage::RocketUIManager::getType \(C++ function\), 143](#)
[Gsage::RocketUIManager::handleInputEvent \(C++ function\), 144](#)
[Gsage::RocketUIManager::handleKeyboardEvent \(C++ function\), 144](#)
[Gsage::RocketUIManager::handleMouseEvent \(C++ function\), 144](#)
[Gsage::RocketUIManager::handleSystemChange \(C++ function\), 143](#)
[Gsage::RocketUIManager::initialize \(C++ function\), 143](#)
[Gsage::RocketUIManager::KeyIdentifierMap \(C++ type\), 144](#)
[Gsage::RocketUIManager::mIsSetUp \(C++ member\), 144](#)
[Gsage::RocketUIManager::mKeyMap \(C++ member\), 144](#)
[Gsage::RocketUIManager::mModifiersState \(C++ member\), 144](#)
[Gsage::RocketUIManager::mRenderSystemWrapper \(C++ member\), 144](#)
[Gsage::RocketUIManager::RocketUIManager \(C++ function\), 143](#)
[Gsage::RocketUIManager::setLuaState \(C++ function\), 143](#)
[Gsage::RocketUIManager::setUp \(C++ function\), 143](#)
[Gsage::RocketUIManager::TYPE \(C++ member\), 143](#)
[Gsage::RocketUIPlugin \(C++ class\), 45, 144](#)
[Gsage::RocketUIPlugin::~~RocketUIPlugin \(C++ function\), 144](#)
[Gsage::RocketUIPlugin::getName \(C++ function\), 144](#)
[Gsage::RocketUIPlugin::installImpl \(C++ function\), 144](#)
[Gsage::RocketUIPlugin::mUIManager \(C++ member\), 145](#)
[Gsage::RocketUIPlugin::mUIManagerHandle \(C++ member\), 145](#)
[Gsage::RocketUIPlugin::RocketUIPlugin \(C++ function\), 144](#)
[Gsage::RocketUIPlugin::setupLuaBindings \(C++ function\), 145](#)
[Gsage::RocketUIPlugin::uninstallImpl \(C++ function\), 145](#)
[Gsage::Root \(C++ enumerator\), 77](#)
[Gsage::RotationAxis \(C++ enum\), 120](#)
[Gsage::Rtt \(C++ enumerator\), 135](#)
[Gsage::RttRenderTarget \(C++ class\), 45, 53, 138](#)
[Gsage::RttRenderTarget::~~RttRenderTarget \(C++ function\), 139](#)
[Gsage::RttRenderTarget::createTexture \(C++ function\), 139](#)
[Gsage::RttRenderTarget::mTexture \(C++ member\), 139](#)
[Gsage::RttRenderTarget::onTextureEvent \(C++ function\), 139](#)
[Gsage::RttRenderTarget::RttRenderTarget \(C++ function\), 139](#)
[Gsage::RttRenderTarget::setDimensions \(C++ function\), 139](#)
[Gsage::RttRenderTarget::wrap \(C++ function\), 139](#)
[Gsage::SceneNodeWrapper \(C++ class\), 46, 151](#)
[Gsage::SceneNodeWrapper::~~SceneNodeWrapper \(C++ function\), 151](#)
[Gsage::SceneNodeWrapper::attach \(C++ function\), 153](#)
[Gsage::SceneNodeWrapper::createNode \(C++ function\), 152](#)
[Gsage::SceneNodeWrapper::destroy \(C++ function\), 153](#)
[Gsage::SceneNodeWrapper::destroyAllAttachedMovable \(C++ function\), 153](#)
[Gsage::SceneNodeWrapper::getChild \(C++ function\), 153](#)
[Gsage::SceneNodeWrapper::getChildOfType \(C++ function\), 153](#)

Gsage::SceneNodeWrapper::getId (C++ function), 152
 Gsage::SceneNodeWrapper::getMovableObject (C++ function), 153
 Gsage::SceneNodeWrapper::getNode (C++ function), 154
 Gsage::SceneNodeWrapper::getOrientation (C++ function), 152
 Gsage::SceneNodeWrapper::getOrientationVector (C++ function), 152
 Gsage::SceneNodeWrapper::getPosition (C++ function), 152
 Gsage::SceneNodeWrapper::getPositionWithOffset (C++ function), 152
 Gsage::SceneNodeWrapper::getScale (C++ function), 152
 Gsage::SceneNodeWrapper::hasNode (C++ function), 152
 Gsage::SceneNodeWrapper::initialize (C++ function), 151
 Gsage::SceneNodeWrapper::lookAt (C++ function), 153
 Gsage::SceneNodeWrapper::mChildren (C++ member), 155
 Gsage::SceneNodeWrapper::mId (C++ member), 154
 Gsage::SceneNodeWrapper::mNode (C++ member), 154
 Gsage::SceneNodeWrapper::mOffset (C++ member), 155
 Gsage::SceneNodeWrapper::mOrientationVector (C++ member), 154
 Gsage::SceneNodeWrapper::ObjectCollection (C++ type), 154
 Gsage::SceneNodeWrapper::Objects (C++ type), 154
 Gsage::SceneNodeWrapper::onFactoryUnregister (C++ function), 154
 Gsage::SceneNodeWrapper::pitch (C++ function), 153
 Gsage::SceneNodeWrapper::readChildren (C++ function), 152
 Gsage::SceneNodeWrapper::roll (C++ function), 154
 Gsage::SceneNodeWrapper::rotate (C++ function), 152, 153
 Gsage::SceneNodeWrapper::SceneNodeWrapper (C++ function), 151
 Gsage::SceneNodeWrapper::setOrientation (C++ function), 152
 Gsage::SceneNodeWrapper::setOrientationVector (C++ function), 152
 Gsage::SceneNodeWrapper::setPosition (C++ function), 152
 Gsage::SceneNodeWrapper::setPositionWithoutOffset (C++ function), 152
 Gsage::SceneNodeWrapper::setScale (C++ function), 152
 Gsage::SceneNodeWrapper::translate (C++ function), 154
 Gsage::SceneNodeWrapper::TYPE (C++ member), 154
 Gsage::SceneNodeWrapper::writeChildren (C++ function), 152
 Gsage::SceneNodeWrapper::yaw (C++ function), 153
 Gsage::SDLAudioSystem (C++ class), 45, 145
 Gsage::SDLAudioSystem::~SDLAudioSystem (C++ function), 145
 Gsage::SDLAudioSystem::SDLAudioSystem (C++ function), 145
 Gsage::SDLCore (C++ class), 45, 145
 Gsage::SDLCore::~SDLCore (C++ function), 145
 Gsage::SDLCore::addEventListener (C++ function), 145
 Gsage::SDLCore::EventListeners (C++ type), 146
 Gsage::SDLCore::getResourcePath (C++ function), 146
 Gsage::SDLCore::initialize (C++ function), 145
 Gsage::SDLCore::mEventListeners (C++ member), 146
 Gsage::SDLCore::mFacade (C++ member), 146
 Gsage::SDLCore::mInitialized (C++ member), 146
 Gsage::SDLCore::mWindowManager (C++ member), 146
 Gsage::SDLCore::removeEventListener (C++ function), 146
 Gsage::SDLCore::SDLCore (C++ function), 145
 Gsage::SDLCore::setWindowManager (C++ function), 146
 Gsage::SDLCore::tearDown (C++ function), 145
 Gsage::SDLCore::update (C++ function), 145
 Gsage::SDLEventListener (C++ class), 45, 53, 146
 Gsage::SDLEventListener::~SDLEventListener (C++ function), 146
 Gsage::SDLEventListener::handleEvent (C++ function), 146
 Gsage::SDLEventListener::mSDLCore (C++ member), 147
 Gsage::SDLEventListener::SDLEventListener (C++ function), 146
 Gsage::SDLEventListener::setSDLCore (C++ function), 146
 Gsage::SDLInputFactory (C++ class), 45, 147

Gsage::SDLInputFactory::~~SDLInputFactory (C++ function), 147
 Gsage::SDLInputFactory::create (C++ function), 147
 Gsage::SDLInputFactory::mCore (C++ member), 147
 Gsage::SDLInputFactory::mListener (C++ member), 147
 Gsage::SDLInputFactory::SDLInputFactory (C++ function), 147
 Gsage::SDLInputFactory::setSDLCore (C++ function), 147
 Gsage::SDLInputListener (C++ class), 45, 147
 Gsage::SDLInputListener::~~SDLInputListener (C++ function), 147
 Gsage::SDLInputListener::handleClose (C++ function), 147
 Gsage::SDLInputListener::handleEvent (C++ function), 147
 Gsage::SDLInputListener::handleKeyboardEvent (C++ function), 148
 Gsage::SDLInputListener::handleMouseEvent (C++ function), 148
 Gsage::SDLInputListener::handleResize (C++ function), 147
 Gsage::SDLInputListener::mapButtonType (C++ function), 148
 Gsage::SDLInputListener::mKeyMap (C++ member), 148
 Gsage::SDLInputListener::SDLInputListener (C++ function), 147
 Gsage::SDLInputListener::update (C++ function), 147
 Gsage::SDLPlugin (C++ class), 45, 148
 Gsage::SDLPlugin::~~SDLPlugin (C++ function), 148
 Gsage::SDLPlugin::getName (C++ function), 148
 Gsage::SDLPlugin::installImpl (C++ function), 148
 Gsage::SDLPlugin::mSDLCore (C++ member), 148
 Gsage::SDLPlugin::SDLPlugin (C++ function), 148
 Gsage::SDLPlugin::setupLuaBindings (C++ function), 148
 Gsage::SDLPlugin::uninstallImpl (C++ function), 148
 Gsage::SDLRenderer (C++ class), 45, 53, 149
 Gsage::SDLRenderer::~~SDLRenderer (C++ function), 149
 Gsage::SDLRenderer::mRenderer (C++ member), 149
 Gsage::SDLRenderer::mRenderers (C++ member), 149
 Gsage::SDLRenderer::render (C++ function), 149
 Gsage::SDLRenderer::Renderers (C++ type), 149
 Gsage::SDLRenderer::SDLRenderer (C++ function), 149
 Gsage::SDLWindow (C++ class), 46, 149
 Gsage::SDLWindow::~~SDLWindow (C++ function), 149
 Gsage::SDLWindow::getDisplay (C++ function), 150
 Gsage::SDLWindow::getDisplayBounds (C++ function), 150
 Gsage::SDLWindow::getGLContext (C++ function), 149
 Gsage::SDLWindow::getPosition (C++ function), 149
 Gsage::SDLWindow::getScaleFactor (C++ function), 150
 Gsage::SDLWindow::getSize (C++ function), 150
 Gsage::SDLWindow::getWindowHandle (C++ function), 149
 Gsage::SDLWindow::hide (C++ function), 150
 Gsage::SDLWindow::mGLContext (C++ member), 150
 Gsage::SDLWindow::mWindow (C++ member), 150
 Gsage::SDLWindow::SDLWindow (C++ function), 149
 Gsage::SDLWindow::setPosition (C++ function), 149
 Gsage::SDLWindow::setSize (C++ function), 150
 Gsage::SDLWindow::show (C++ function), 150
 Gsage::SDLWindowManager (C++ class), 46, 53, 150
 Gsage::SDLWindowManager::~~SDLWindowManager (C++ function), 150
 Gsage::SDLWindowManager::createWindow (C++ function), 150
 Gsage::SDLWindowManager::destroyWindow (C++ function), 151
 Gsage::SDLWindowManager::initialize (C++ function), 150
 Gsage::SDLWindowManager::mCore (C++ member), 151
 Gsage::SDLWindowManager::mRenderers (C++ member), 151
 Gsage::SDLWindowManager::mWindowsMutex (C++ member), 151
 Gsage::SDLWindowManager::SDLRendererPtr (C++ type), 151

Gsage::SDLWindowManager::SDLWindowManager
 (C++ function), 150
 Gsage::SDLWindowManager::update (C++
 function), 151
 Gsage::STATIC (C++ enumerator), 69, 98
 Gsage::Tab (C++ enumerator), 77
 Gsage::Top (C++ enumerator), 77
 Gsage::Type (C++ enum), 135
 Gsage::TYPE_CASTER (C++ function), 49, 114
 Gsage::UNKNOWN (C++ enumerator), 69, 98
 Gsage::Unspecified (C++ enumerator), 77
 Gsage::Vertical (C++ enumerator), 76
 Gsage::ViewportRenderData (C++ class), 46,
 155
 Gsage::ViewportRenderData::~~ViewportRenderData (C++ function), 156
 (C++ function), 155
 Gsage::ViewportRenderData::getRenderTexture
 (C++ function), 156
 Gsage::ViewportRenderData::mDirty (C++
 member), 156
 Gsage::ViewportRenderData::mDrawCmd
 (C++ member), 156
 Gsage::ViewportRenderData::mIndexBuffer
 (C++ member), 156
 Gsage::ViewportRenderData::mPos (C++
 member), 156
 Gsage::ViewportRenderData::mSize (C++
 member), 156
 Gsage::ViewportRenderData::mTextureName
 (C++ member), 156
 Gsage::ViewportRenderData::mTexUnitState
 (C++ member), 156
 Gsage::ViewportRenderData::mVertexBuffer
 (C++ member), 156
 Gsage::ViewportRenderData::pos (C++ mem-
 ber), 156
 Gsage::ViewportRenderData::resetDataBlock
 (C++ function), 155
 Gsage::ViewportRenderData::setDataBlock
 (C++ function), 156
 Gsage::ViewportRenderData::size (C++
 member), 156
 Gsage::ViewportRenderData::updatePos
 (C++ function), 155
 Gsage::ViewportRenderData::updateSize
 (C++ function), 155
 Gsage::ViewportRenderData::updateUVs
 (C++ function), 155
 Gsage::ViewportRenderData::updateVertexBuffer
 (C++ function), 155
 Gsage::ViewportRenderData::ViewportRenderData
 (C++ function), 155
 Gsage::Window (C++ enumerator), 135
 Gsage::WindowContentViewHandle (C++ func-
 tion), 49, 113
 Gsage::WindowEventListener (C++ class), 46,
 53, 156
 Gsage::WindowEventListener::~~WindowEventListener
 (C++ function), 156
 Gsage::WindowEventListener::fireWindowEvent
 (C++ function), 157
 Gsage::WindowEventListener::mEngine
 (C++ member), 157
 Gsage::WindowEventListener::windowClosed
 (C++ function), 156
 Gsage::WindowEventListener::WindowEventListener
 (C++ function), 156
 Gsage::WindowEventListener::windowResized
 (C++ function), 156
 Gsage::WindowRenderTarget (C++ class), 46,
 53, 139
 Gsage::WindowRenderTarget::~~WindowRenderTarget
 (C++ function), 139
 Gsage::WindowRenderTarget::setDimensions
 (C++ function), 139
 Gsage::WindowRenderTarget::WindowRenderTarget
 (C++ function), 139
 Gsage::WorkspaceEvent (C++ class), 46, 53, 157
 Gsage::WorkspaceEvent::~~WorkspaceEvent
 (C++ function), 157
 Gsage::WorkspaceEvent::CREATE (C++ mem-
 ber), 157
 Gsage::WorkspaceEvent::mWorkspace (C++
 member), 157
 Gsage::WorkspaceEvent::WorkspaceEvent
 (C++ function), 157
 Gsage::X_AXIS (C++ enumerator), 120
 Gsage::Y_AXIS (C++ enumerator), 120
 Gsage::Z_AXIS (C++ enumerator), 120
 GSAGE_OGRE_PLUGIN_API (C macro), 69
 ImGui (C++ type), 53, 158
 ImGui::AddConvexPolyFilled (C++ function),
 53, 158
 ImGui::Gradient (C++ class), 48, 53, 158
 ImGui::Gradient::Calc (C++ function), 158
 ImGui::PathFillConvex (C++ function), 53, 158
 ImGui::VerticalGradient (C++ class), 48, 158
 ImGui::VerticalGradient::Calc (C++ func-
 tion), 158
 ImGui::VerticalGradient::Col0 (C++ mem-
 ber), 158
 ImGui::VerticalGradient::Col1 (C++ mem-
 ber), 158
 ImGui::VerticalGradient::End (C++ mem-
 ber), 158

ImGui::VerticalGradient::evalStep (C++ function), 158
 ImGui::VerticalGradient::Len (C++ member), 158
 ImGui::VerticalGradient::Start (C++ member), 158
 ImGui::VerticalGradient::VerticalGradient (C++ function), 158
 IMGUI_DEFINE_MATH_OPERATORS (C macro), 76, 158
 IMGUI_MATERIAL_NAME (C macro), 86
 IMGUI_PLUGIN_API (C macro), 86
 ImGuizmo (C++ type), 53
 ImGuizmo::BOUNDS (C++ enumerator), 54
 ImGuizmo::LOCAL (C++ enumerator), 54
 ImGuizmo::MODE (C++ enum), 54
 ImGuizmo::OPERATION (C++ enum), 53, 54
 ImGuizmo::ROTATE (C++ enumerator), 53, 54
 ImGuizmo::SCALE (C++ enumerator), 53, 54
 ImGuizmo::TRANSLATE (C++ enumerator), 53, 54
 ImGuizmo::WORLD (C++ enumerator), 54

L

LOOP (C macro), 56

M

MOC (C++ type), 55, 66
 MOC::CollisionTools (C++ class), 46, 66
 MOC::CollisionTools::_heightAdjust (C++ member), 67
 MOC::CollisionTools::~CollisionTools (C++ function), 66
 MOC::CollisionTools::calculateY (C++ function), 66
 MOC::CollisionTools::collidesWithEntity (C++ function), 66
 MOC::CollisionTools::CollisionTools (C++ function), 66
 MOC::CollisionTools::getHeightAdjust (C++ function), 67
 MOC::CollisionTools::getTSMHeightAt (C++ function), 66
 MOC::CollisionTools::mRaySceneQuery (C++ member), 67
 MOC::CollisionTools::mSceneMgr (C++ member), 67
 MOC::CollisionTools::mTSMRaySceneQuery (C++ member), 67
 MOC::CollisionTools::raycast (C++ function), 67
 MOC::CollisionTools::raycastFromCamera (C++ function), 66
 MOC::CollisionTools::raycastFromPoint (C++ function), 66

MOC::CollisionTools::setHeightAdjust (C++ function), 67

O

Ogre (C++ type), 55, 56, 64, 69, 71, 98, 101, 111, 114, 115, 118, 120, 124, 135, 157
 Ogre::All (C++ enumerator), 111
 Ogre::Default (C++ enumerator), 111
 Ogre::Flags (C++ enum), 111
 Ogre::Indices (C++ enumerator), 111
 Ogre::ManualMovableTextRenderer (C++ class), 46, 55, 101
 Ogre::ManualMovableTextRenderer::_destroyVisualData (C++ function), 102
 Ogre::ManualMovableTextRenderer::_getSortMode (C++ function), 102
 Ogre::ManualMovableTextRenderer::_notifyAttached (C++ function), 102
 Ogre::ManualMovableTextRenderer::_notifyCurrentCamera (C++ function), 102
 Ogre::ManualMovableTextRenderer::_notifyDefaultDimensions (C++ function), 102
 Ogre::ManualMovableTextRenderer::_notifyParticleQuota (C++ function), 101
 Ogre::ManualMovableTextRenderer::_setDataBlock (C++ function), 102
 Ogre::ManualMovableTextRenderer::_setMaterialName (C++ function), 102
 Ogre::ManualMovableTextRenderer::_updateRenderQueue (C++ function), 101
 Ogre::ManualMovableTextRenderer::~ManualMovableTextRenderer (C++ function), 101
 Ogre::ManualMovableTextRenderer::adjustNodeCount (C++ function), 102
 Ogre::ManualMovableTextRenderer::CmdFontName (C++ class), 47, 103
 Ogre::ManualMovableTextRenderer::CmdFontName::doGet (C++ function), 103
 Ogre::ManualMovableTextRenderer::CmdFontName::doSet (C++ function), 103
 Ogre::ManualMovableTextRenderer::getFontName (C++ function), 101
 Ogre::ManualMovableTextRenderer::getType (C++ function), 101
 Ogre::ManualMovableTextRenderer::ManualMovableTextRenderer (C++ function), 101
 Ogre::ManualMovableTextRenderer::mBusyNodes (C++ member), 103
 Ogre::ManualMovableTextRenderer::mFontName (C++ member), 103
 Ogre::ManualMovableTextRenderer::mFreeNodes (C++ member), 103
 Ogre::ManualMovableTextRenderer::mName (C++ member), 103

Ogre::ManualMovableTextRenderer::mNodePool *member*), 111
 (C++ *member*), 103
 Ogre::ManualMovableTextRenderer::mPreviousParticleMeshUnits:~MeshTools (C++ *function*),
 (C++ *member*), 103 112
 Ogre::ManualMovableTextRenderer::mQuota Ogre::MeshTools::getMeshInformation
 (C++ *member*), 103 (C++ *function*), 112
 Ogre::ManualMovableTextRenderer::msFontName Ogre::MeshTools::getSingleton (C++ *func-*
 (C++ *member*), 102 *tion*), 112
 Ogre::ManualMovableTextRenderer::setFontName Ogre::MeshTools::getSingletonPtr (C++
 (C++ *function*), 101 *function*), 112
 Ogre::ManualMovableTextRenderer::setKeepParticleMeshUnits Ogre::MeshTools (C++ *function*),
 (C++ *function*), 102 112
 Ogre::ManualMovableTextRenderer::setRenderQueueGroup Ogre::MovableText (C++ *class*), 47
 (C++ *function*), 102 Ogre::MovableTextFactory (C++ *class*), 47, 55
 Ogre::ManualMovableTextRenderer::setRenderQueueGroupAndPtrValue (C++ *class*), 47, 55,
 (C++ *function*), 102 103
 Ogre::ManualMovableTextRenderer::setRenderQueueSubGroup Ogre::MovableTextValue::getNode (C++
 (C++ *function*), 102 *function*), 103
 Ogre::ManualMovableTextRenderer::TextNode Ogre::MovableTextValue::getNodeToAttachTo
 (C++ *type*), 102 (C++ *function*), 104
 Ogre::ManualMovableTextRenderer::TextNodeMap Ogre::MovableTextValue::getValue (C++
 (C++ *type*), 102 *function*), 103
 Ogre::ManualMovableTextRenderer::TextNodePers Ogre::MovableTextValue::mNode (C++ *mem-*
 (C++ *type*), 102 *ber*), 104
 Ogre::ManualMovableTextRendererFactory Ogre::MovableTextValue::MovableTextValue
 (C++ *class*), 46, 103 (C++ *function*), 103
 Ogre::ManualMovableTextRendererFactory::Ogre::ManualMovableTextRenderer::mSceneNode (C++
 (C++ *function*), 103 *member*), 104
 Ogre::ManualMovableTextRendererFactory::Ogre::MovableTextValue::mValue (C++ *mem-*
 (C++ *function*), 103 *ber*), 104
 Ogre::ManualMovableTextRendererFactory::Ogre::MovableTextValue::setNode (C++
 (C++ *function*), 103 *function*), 103
 Ogre::ManualMovableTextRendererFactory::GetTypeNormals (C++ *enumerator*), 111
 (C++ *function*), 103 Ogre::TextNode (C++ *class*), 47, 55, 104
 Ogre::ManualMovableTextRendererFactory::Ogre::ManualMovableTextRendererFactory (C++ *function*),
 (C++ *function*), 103 104
 Ogre::ManualMovableTextRendererFactory::Ogre::ManualMovableTextRendererFactory::createRenderersActive (C++ *function*), 104
 (C++ *member*), 103 Ogre::TextNode::deactivate (C++ *function*),
 104
 Ogre::MeshInformation (C++ *class*), 47, 55, 111
 Ogre::MeshInformation::~MeshInformation Ogre::TextNode::mId (C++ *member*), 104
 (C++ *function*), 111 Ogre::TextNode::mObjectManager (C++ *mem-*
 104
 Ogre::MeshInformation::allocate (C++ *function*), 111
 104
 Ogre::MeshInformation::indexCount (C++ *member*), 111
 104
 Ogre::MeshInformation::indices (C++ *mem-*
ber), 111
 104
 Ogre::MeshInformation::MeshInformation Ogre::TextNode::mValue (C++ *member*), 104
 (C++ *function*), 111 Ogre::TextNode::mView (C++ *member*), 104
 104
 Ogre::MeshInformation::normals (C++ *mem-*
ber), 111
 104
 Ogre::MeshInformation::vertexCount (C++ *member*), 111
 104
 104
 Ogre::MeshInformation::vertices (C++ *member*), 111
 104
 104

Ogre::TextNode::TextNode (C++ *function*), 104
 Ogre::Vertices (C++ *enumerator*), 111
 OGRE_CONFIG_PATH (C++ *member*), 124
 OGRE_PLUGINS_PATH (C++ *member*), 124
 OGRE_RESOURCES (C++ *member*), 124
 OGRE_SECTION (C++ *member*), 124
 OgreV1 (C *macro*), 69

R

RENDER_QUEUE_IMGUI (C *macro*), 86
 RenderInterfaceOgre3D (C++ *class*), 47, 133
 RenderInterfaceOgre3D::~RenderInterfaceOgre3D (C++ *function*), 134
 RenderInterfaceOgre3D::alpha_blend_mode (C++ *member*), 134
 RenderInterfaceOgre3D::colour_blend_mode (C++ *member*), 134
 RenderInterfaceOgre3D::CompileGeometry (C++ *function*), 134
 RenderInterfaceOgre3D::customProjectionMatrix (C++ *member*), 134
 RenderInterfaceOgre3D::EnableScissorRegion (C++ *function*), 134
 RenderInterfaceOgre3D::GenerateTexture (C++ *function*), 134
 RenderInterfaceOgre3D::GetHorizontalTexelOffset (C++ *function*), 134
 RenderInterfaceOgre3D::GetVerticalTexelOffset (C++ *function*), 134
 RenderInterfaceOgre3D::LoadTexture (C++ *function*), 134
 RenderInterfaceOgre3D::m_blankTexture (C++ *member*), 135
 RenderInterfaceOgre3D::ReleaseCompiledGeometry (C++ *function*), 134
 RenderInterfaceOgre3D::ReleaseTexture (C++ *function*), 134
 RenderInterfaceOgre3D::render_system (C++ *member*), 134
 RenderInterfaceOgre3D::RenderCompiledGeometry (C++ *function*), 134
 RenderInterfaceOgre3D::RenderGeometry (C++ *function*), 134
 RenderInterfaceOgre3D::RenderInterfaceOgre3D (C++ *function*), 134
 RenderInterfaceOgre3D::scissor_bottom (C++ *member*), 135
 RenderInterfaceOgre3D::scissor_enable (C++ *member*), 135
 RenderInterfaceOgre3D::scissor_left (C++ *member*), 135
 RenderInterfaceOgre3D::scissor_right (C++ *member*), 135

RenderInterfaceOgre3D::scissor_top (C++ *member*), 135
 RenderInterfaceOgre3D::setCustomProjectionMatrix (C++ *function*), 134
 RenderInterfaceOgre3D::SetScissorRegion (C++ *function*), 134

S

sol (C++ *type*), 55
 SystemInterfaceOgre3D (C++ *class*), 47, 155
 SystemInterfaceOgre3D::~SystemInterfaceOgre3D (C++ *function*), 155
 SystemInterfaceOgre3D::GetElapsedTime (C++ *function*), 155
 SystemInterfaceOgre3D::LogMessage (C++ *function*), 155
 SystemInterfaceOgre3D::SystemInterfaceOgre3D (C++ *function*), 155
 SystemInterfaceOgre3D::timer (C++ *member*), 155