# Ariyana Documentation

***Release 0.1***

**Ali Akbar Mohammadi**

**Jan 29, 2019**

# Contents:

Ariyana is an open source and free WIP game engine.

CHAPTER 1

# Building

## 1.1 Getting Source

```
git clone https://github.com/kochol/ariyana.git
cd ariyana
git submodule update --init
```

## 1.2 Quick Start

These are step for users who use Windows with Visual Studio.

Enter ariyana directory:

```
cd ariyana
```

Generate Visual Studio 2017 project files:

```
deps\bx\tools\bin\windows\genie vs2017
```

Open ariyana solution in Visual Studio 2017:

```
start .build\projects\vs2017\ariyana.sln
```

Generate Visual Studio 2017 project files with test projects:

```
deps\bx\tools\bin\windows\genie --with-tests vs2017
```

CHAPTER 2

# Design goals

The goal of this engine design is to make a fast engine.

## 2.1 Jobify the engine with fibers

The first design goal is using fibers to jobify the engine and use all of the CPU cores. For more info see GDC talk by Christian Gyrling in his 2015 GDC Talk 'Parallelizing the Naughty Dog Engine Using Fibers'

## 2.2 Frame data

Use frame data to enable the parallel work of different parts of the engine without waiting for each other to finish their job. For example gameplay systems update entities and send them to scene system then scene system store them to perform culling and sorting next frame after that sends it to render system for next frame rendering so rendering is always two frame behind gameplay systems this design makes every thing works in parallel. For more info see GDC talk by Christian Gyrling in his 2015 GDC Talk 'Parallelizing the Naughty Dog Engine Using Fibers'

For this system to work I made different update stages that be called for every system that needs it.

## 2.3 Entity, Component and Node

This part of the design I got inspired by Godot engine where Components can attach to each other and creates a tree. The base class here is Node class that Entity and Component class are inherited from them so Components can attach to entities and vice versa. This design works well for example when you work on a project with your colleges and everyone can work on a different entity without conflict.

Also, entities are useful when later we add networking to the engine.

CHAPTER 3

# API Reference

## 3.1 Class Hierarchy

## 3.2 File Hierarchy

## 3.3 Full API

### 3.3.1 Namespaces

#### Namespace ari

**Contents**



#### Namespaces

### Classes

- *Class EventQueue*
- *Template Class EventSubscriber*
- *Class FrameData*
- *Class Gui*
- *Class GuiSystem*
- *Class Image*
- *Class IProgram*
- *Class Label*
- *Class Node*
- *Class Node3D*
- *Class PlatformWindow*
- *Class Plugin*
- *Class PluginManager*
- *Class Popup*
- *Class RenderSystem*
- *Class Resource*
- *Class ResourceLoader*
- *Template Class ResourceManager*
- *Class SceneSystem*
- *Class System*
- *Class TextBox*
- *Class Texture*
- *Class TextureManager*
- *Class Viewport*
- *Class Window*
- *Class World*

### Functions

- *Function ari::BX_ALIGN_DECL_16*
- *Function ari::getDefaultAllocator*
- *Function ari::getName*
- *Template Function ari::getTypeIndex*
- *Function ari::inputAddBindings*
- *Function ari::inputChar*
- *Function ari::inputCharFlush*
- *Function ari::inputGetChar*

- *Function ari::inputGetGamepadAxis*
- *Function ari::inputGetKeyState*
- *Function ari::inputGetModifiersState*
- *Function ari::inputGetMouse*
- *Function ari::inputInit*
- *Function ari::inputIsMouseLocked*
- *Function ari::inputProcess*
- *Function ari::inputRemoveBindings*
- *Function ari::inputSetGamepadAxis*
- *Function ari::inputSetKeyState*
- *Function ari::inputSetMouseButtonState*
- *Function ari::inputSetMouseLock*
- *Function ari::inputSetMousePos*
- *Function ari::inputSetMouseResolution*
- *Function ari::inputShutdown*
- *Function ari::isValid(WindowHandle)*
- *Function ari::isValid(GamepadHandle)*
- *Function ari::poll()*
- *Function ari::poll(WindowHandle)*
- *Function ari::release*

### Typedefs

- *Typedef ari::InputBindingFn*
- *Typedef ari::RectF*
- *Typedef ari::RectI*
- *Typedef ari::RectU16*
- *Typedef ari::TypeIndex*

### Variables

- *Variable ari::fDegToRad*
- *Variable ari::fEpsilon*
- *Variable ari::fRadToDeg*
- *Variable ari::g_pEngine*
- *Variable ari::PI*
- *Variable ari::PiOver2*
- *Variable ari::TwoPI*

### Namespace ari::events

**Contents**



### Classes



### Namespace ari::Internal

**Contents**



### Classes



### Namespace bgfx

### Namespace bx

### Namespace ftl

### Namespace ImWindow

### Namespace meta

**Contents**

### Functions



### Namespace shiva

**Contents**



### Classes

#### Variables

- *Variable shiva::g_pEditor*

#### Namespace spdlog

### 3.3.2 Classes and Structs

#### Struct AxisEvent

- Defined in *File IoEvents.hpp*

#### Inheritance Relationships

#### Base Type

- `public ari::Event` (*Struct Event*)

#### Struct Documentation

**`struct AxisEvent`** : **`public`** ari::*Event*

##### Public Functions

**`AxisEvent`**()

##### Public Members

*GamepadAxis*::*Enum* **`m_axis`**

int32_t **`m_value`**

*GamepadHandle* **`m_gamepad`**

#### Struct CharEvent

- Defined in *File IoEvents.hpp*

#### Inheritance Relationships

#### Base Type

- `public ari::Event` (*Struct Event*)

### Struct Documentation

**struct CharEvent** : **public** ari::*Event*

#### Public Functions

**CharEvent**()

#### Public Members

uint8_t **m_len**

uint8_t **m_char**[4]

### Struct ColorVertex

- Defined in *File Vertices.hpp*

### Struct Documentation

**struct ColorVertex**

#### Public Members

uint32_t **argb**

### Struct DropFileEvent

- Defined in *File IoEvents.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Event` (*Struct Event*)

### Struct Documentation

**struct DropFileEvent** : **public** ari::*Event*

#### Public Functions

**DropFileEvent**()

#### Public Members

bx::FilePath **m_filePath**

### Struct Event

- Defined in *File IoEvents.hpp*

### Inheritance Relationships

### Derived Types

- `public ari::AxisEvent` (*Struct AxisEvent*)
- `public ari::CharEvent` (*Struct CharEvent*)
- `public ari::DropFileEvent` (*Struct DropFileEvent*)
- `public ari::GamepadEvent` (*Struct GamepadEvent*)
- `public ari::KeyEvent` (*Struct KeyEvent*)
- `public ari::MouseEvent` (*Struct MouseEvent*)
- `public ari::SizeEvent` (*Struct SizeEvent*)
- `public ari::SuspendEvent` (*Struct SuspendEvent*)
- `public ari::WindowEvent` (*Struct WindowEvent*)

### Struct Documentation

**struct Event**

Subclassed by *ari::AxisEvent*, *ari::CharEvent*, *ari::DropFileEvent*, *ari::GamepadEvent*, *ari::KeyEvent*, *ari::MouseEvent*, *ari::SizeEvent*, *ari::SuspendEvent*, *ari::WindowEvent*

#### Public Types

**enum Enum**

*Values:*

**Axis**

**Char**

**Exit**

**Gamepad**

**Key**

**Mouse**

**Size**

**Window**

**Suspend**

**DropFile**

#### Public Functions

**Event** (*Enum* *_type*)

#### Public Members

*Event*::*Enum* **m_type**

### Template Struct OnComponentAssigned

- Defined in *File EventSubscriber.hpp*

### Struct Documentation

**template** <class *T*>
**struct OnComponentAssigned**

#### Public Members

*Entity* ***entity**

T ***component**

### Template Struct OnComponentRemoved

- Defined in *File EventSubscriber.hpp*

### Struct Documentation

**template** <class *T*>
**struct OnComponentRemoved**

#### Public Members

*Entity* ***entity**

T ***component**

### Struct OnEntityCreated

- Defined in *File EventSubscriber.hpp*

### Struct Documentation

**struct OnEntityCreated**

#### Public Members

*Entity* ***entity**

### Struct OnEntityDestroyed

- Defined in *File EventSubscriber.hpp*

### Struct Documentation

**struct OnEntityDestroyed**

#### Public Members

*Entity* ***entity**

### Struct OnFrameData

- Defined in *File EventSubscriber.hpp*

### Struct Documentation

**struct OnFrameData**

#### Public Members

*FrameData* ***frame_data**

### Struct GamepadAxis

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct GamepadAxis**

#### Public Types

**enum Enum**

*Values:*

**LeftX**

**LeftY**

**LeftZ**

**RightX**

**RightY**

**RightZ**

**Count**

### Struct GamepadEvent

- Defined in *File IoEvents.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Event` (*Struct Event*)

### Struct Documentation

**struct GamepadEvent** : **public** ari::*Event*

#### Public Functions

**GamepadEvent**()

#### Public Members

*GamepadHandle* **m_gamepad**

bool **m_connected**

### Struct GamepadHandle

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct GamepadHandle**

#### Public Members

uint16_t **idx**

### Struct GamepadState

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct GamepadState**

#### Public Functions

**GamepadState**()

#### Public Members

int32_t **m_axis**[**Count**]

### Struct InitParams

- Defined in *File Engine.hpp*

### Struct Documentation

**struct InitParams**

#### Public Functions

**InitParams**()

#### Public Members

uint32_t **Height**

uint32_t **Width**

bool **FullScreen**

*IProgram* ***Program**

### Struct InputBinding

- Defined in *File Input.hpp*

### Struct Documentation

**struct InputBinding**

#### Public Functions

void **set**(*Key*::*Enum* *_key*, uint8_t *_modifiers*, uint8_t *_flags*, *InputBindingFn* *_fn*, **const** void *_*userData* = NULL)

void **end**()

#### Public Members

*Key*::*Enum* **m_key**

uint8_t **m_modifiers**

uint8_t **m_flags**

*InputBindingFn* **m_fn**

**const** void ***m_userData**

### Struct Key

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct Key**

#### Public Types

**enum Enum**

*Values:*

**None** = 0

**Esc**

**Return**

**Tab**

**Space**

**Backspace**

**Up**

**Down**

**Left**

**Right**

**Insert**

**Delete**

**Home**

**End**

**PageUp**

**PageDown**

**Print**

**Plus**

**Minus**

**LeftBracket**

**RightBracket**

**Semicolon**

**Quote**

**Comma**

**Period**

**Slash**

**Backslash**

**Tilde**

**F1**

**F2**

**F3**

**F4**

**F5**

**F6**

**F7**

**F8**

**F9**

**F10**

**F11**

**F12**

**NumPad0**

**NumPad1**

**NumPad2**

**NumPad3**

**NumPad4**

**NumPad5**

**NumPad6**

**NumPad7**

**NumPad8**

**NumPad9**

**Key0**

**Key1**

**Key2**

**Key3**

**Key4**

**Key5**

**Key6**

**Key7**

**Key8**

**Key9**

**KeyA**

**KeyB**

**KeyC**

**KeyD**

**KeyE**

**KeyF**

**KeyG**

**KeyH**

**KeyI**

**KeyJ**

**KeyK**

**KeyL**

**KeyM**

**KeyN**

**KeyO**

**KeyP**

**KeyQ**

**KeyR**

**KeyS**

**KeyT**

**KeyU**

**KeyV**

**KeyW**

**KeyX**

**KeyY**

**KeyZ**

**GamepadA**

**GamepadB**

**GamepadX**

**GamepadY**

**GamepadThumbL**

**GamepadThumbR**

**GamepadShoulderL**

**GamepadShoulderR**

**GamepadUp**

**GamepadDown**

**GamepadLeft**

**GamepadRight**

**GamepadBack**

**GamepadStart**

**GamepadGuide**

**Count**

### Struct KeyEvent

- Defined in *File IoEvents.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Event` (*Struct Event*)

### Struct Documentation

**struct KeyEvent** : **public** ari::*Event*

#### Public Functions

**KeyEvent**()

#### Public Members

*Key*::*Enum* **m_key**

uint8_t **m_modifiers**

bool **m_down**

### Struct Modifier

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct Modifier**

#### Public Types

**enum Enum**

*Values:*

**None** = 0

**LeftAlt** = 0x01

**RightAlt** = 0x02

**LeftCtrl** = 0x04

**RightCtrl** = 0x08

**LeftShift** = 0x10

**RightShift** = 0x20

**LeftMeta** = 0x40

**RightMeta** = 0x80

### Struct MouseButton

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct MouseButton**

#### Public Types

**enum Enum**

*Values:*

**None**

**Left**

**Middle**

**Right**

**Count**

### Struct MouseEvent

- Defined in *File IoEvents.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Event` (*Struct Event*)

### Struct Documentation

**struct MouseEvent** : **public** ari::*Event*

#### Public Functions

**MouseEvent**()

#### Public Members

int32_t **m_mx**

int32_t **m_my**

int32_t **m_mz**

*MouseButton*::*Enum* **m_button**

bool **m_down**

bool **m_move**

### Struct MouseState

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct MouseState**

#### Public Functions

**MouseState**()

#### Public Members

int32_t **m_mx**

int32_t **m_my**

int32_t **m_mz**

uint8_t **m_buttons**[**Count**]

### Struct PosVertex

- Defined in *File Vertices.hpp*

### Struct Documentation

**struct PosVertex**

#### Public Members

float **x**

float **y**

float **z**

### Template Struct Rect

- Defined in *File Rect.hpp*

### Struct Documentation

**template** <class *T*>
**struct Rect**

#### Public Functions

**Rect**()

**Rect**(**const** T *_x*, **const** T *_y*, **const** T *_width*, **const** T *_height*)

void **Set**(**const** T *_x*, **const** T *_y*, **const** T *_width*, **const** T *_height*)

bool **operator==**(**const** *Rect*<T> &*v*) **const**

bool **operator!=**(**const** *Rect*<T> &*v*) **const**

#### Public Members

template<>
T **p**[4]

T **x**

T **y**

T **width**

T **height**

**union ari::Rect::@0 ari::Rect::@1**

### Struct SizeEvent

- Defined in *File IoEvents.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Event` (*Struct Event*)

### Struct Documentation

**struct SizeEvent** : **public** ari::*Event*

#### Public Functions

**SizeEvent**()

#### Public Members

uint32_t **m_width**

uint32_t **m_height**

### Struct Suspend

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct Suspend**

#### Public Types

**enum Enum**

*Values:*

**WillSuspend**

**DidSuspend**

**WillResume**

**DidResume**

**Count**

### Struct SuspendEvent

- Defined in *File IoEvents.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Event` (*Struct Event*)

### Struct Documentation

**struct SuspendEvent** : **public** ari::*Event*

#### Public Functions

**SuspendEvent**()

#### Public Members

*Suspend*::*Enum* **m_state**

### Struct TextureParams

- Defined in *File Texture.hpp*

### Struct Documentation

**struct TextureParams**

#### Public Members

uint32_t **Flags** = BGFX_TEXTURE_NONE

bgfx::TextureInfo ***Info** = nullptr

bimg::Orientation::Enum ***Orientation** = nullptr

### Struct TinyStlAllocator

- Defined in *File aridef.hpp*

### Struct Documentation

**struct TinyStlAllocator**

#### Public Static Functions

**static** void ***static_allocate**(size_t *_bytes*)

**static** void **static_deallocate**(void **_ptr*, size_t)

### Struct Vector3

- Defined in *File Vector.hpp*

### Struct Documentation

**struct Vector3**

#### Public Functions

**Vector3**()

**Vector3**(**const** float *_x*, **const** float *_y*, **const** float *_z*)

void **Set**(**const** float *_x*, **const** float *_y*, **const** float *_z*)

*Vector3* **operator-**(**const** *Vector3* &*v*) **const**

void **Cross**(**const** *Vector3* &*_v1*, **const** *Vector3* &*_v2*)

float **GetLength**() **const**
Returns the vector length.

void **Normalize**()

bx::Vec3 **ToVec3**() **const**

#### Public Members

float **v**[3]

float **x**

float **y**

float **z**

**union ari::Vector3::@4 ari::Vector3::@5**

### Struct WindowEvent

- Defined in *File IoEvents.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Event` (*Struct Event*)

### Struct Documentation

**struct WindowEvent** : **public** ari::*Event*

#### Public Functions

**WindowEvent**()

#### Public Members

void ***m_nwh**

### Struct WindowHandle

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct WindowHandle**

#### Public Members

uint16_t **idx**

### Struct WindowState

- Defined in *File IoEnums.hpp*

### Struct Documentation

**struct WindowState**

#### Public Functions

**WindowState**()

#### Public Members

*WindowHandle* **m_handle**

uint32_t **m_width**

uint32_t **m_height**

*MouseState* **m_mouse**

void ***m_nwh**

bx::FilePath **m_dropFile**

### Struct FileInfo

- Defined in *File DirectoryTree.hpp*

### Struct Documentation

**struct FileInfo**

#### Public Members

std::string **Name**

### Class BoxShape

- Defined in *File BoxShape.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Node3D` (*Class Node3D*)

### Class Documentation

**class BoxShape** : **public** ari::*Node3D*

#### Public Functions

**BoxShape**()

**virtual ~BoxShape**()
Destructor.

**virtual** void **Render**(**const** Matrix &*matrix*, bgfx::Encoder **encoder*, uint16_t *_view_id*)
Render.

#### Public Static Functions

**static** void **Init**(*RenderSystem* **render_system*)

**static** void **Shutdown**()

#### Public Static Attributes

bgfx::VertexBufferHandle **m_sVBPos**

bgfx::VertexBufferHandle **m_sVBColor**

bgfx::IndexBufferHandle **m_sIB**

bgfx::ProgramHandle **m_sProgram**

### Class Button

- Defined in *File Button.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class Button** : **public** ari::*Gui*

#### Public Functions

bool **BeginRender**()

#### Public Members

*DelegateNoParam*<void> **OnClick**

char ***Label**

### Class Camera

- Defined in *File Camera.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Node3D` (*Class Node3D*)

### Class Documentation

**class Camera** : **public** ari::*Node3D*

#### Public Functions

**Camera**()
Constructor.

**virtual ~Camera**()
Destructor.

void **Rotate**(float *_angle*, **const** *Vector3* &*_axis*)
Rotate the camera around an axis.

void **RotateByMouse**(int *_x*, int *_y*, float *_speed*)
Rotate by mouse movement.

void **MoveBF**(**const** float &*_speed*)
Move back & forward.

void **MoveLR**(**const** float &*_speed*)
Move left & right.

void **MoveUD**(**const** float &*_speed*)
Move up & down.

#### Public Members

*Vector3* **Target**

*Vector3* **Up**

*Vector3* **Right**

Matrix **_view**

Matrix **_proj**

bool **_isActive**

#### Protected Attributes

float **m_fCurRotX** = 0.0f

float **m_fLastRotX** = 0.0f

### Class CheckBox

- Defined in *File CheckBox.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class CheckBox** : **public** ari::*Gui*

#### Public Functions

**CheckBox**()

**~CheckBox**()

bool **BeginRender**()

#### Public Members

bool **Checked**

char ***Label**

### Class Component

- Defined in *File Component.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Node` (*Class Node*)

### Derived Types

- `public ari::Gui` (*Class Gui*)
- `public ari::Node3D` (*Class Node3D*)
- `public ari::Viewport` (*Class Viewport*)

### Class Documentation

**class Component** : **public** ari::*Node*
Subclassed by *ari::Gui*, *ari::Node3D*, *ari::Viewport*

#### Public Functions

**Component**()
Constructor.

**virtual ~Component**()
Destructor.

#### Public Members

bool **_isFromNode3D**

bool **_isFromGui**

### Template Class DelegateEightParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateEightParam::BaseFuncEightParam*
- *Template Class DelegateEightParam::MemFuncEightParam*

### Class Documentation

**template <class** Treturn, **class** Targ1, **class** Targ2, **class** Targ3, **class** Targ4, **class** Targ5, **class** Targ6, **class** Targ7
**class DelegateEightParam**

#### Public Functions

**DelegateEightParam**()

**~DelegateEightParam**()

void **Bind**(Treturn (*_fun*)) Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8

```
template <class Tclass>
void ari::DelegateEightParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2, Tar
```

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8)

*BaseFuncEightParam* ***m_pMemFun**

### Class DelegateEightParam::BaseFuncEightParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateEightParam*.

### Class Documentation

**class BaseFuncEightParam**

#### Public Functions

template<>
**virtual ~BaseFuncEightParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*) = 0

### Template Class DelegateEightParam::MemFuncEightParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateEightParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateEightParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8 >::BaseFuncEightParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncEightParam`** : **`public`** ari::DelegateEightParam<Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8

#### Public Functions

**`ari::DelegateEightParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Tar`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8)

### Template Class DelegateFiveParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateFiveParam::BaseFuncFiveParam*
- *Template Class DelegateFiveParam::MemFuncFiveParam*

### Class Documentation

**template** <**class** Treturn, **class** Targ1, **class** Targ2, **class** Targ3, **class** Targ4, **class** Targ5>
**class DelegateFiveParam**

#### Public Functions

**DelegateFiveParam**()

**~DelegateFiveParam**()

void **Bind**(Treturn (*_*fun*)) Targ1, Targ2, Targ3, Targ4, Targ5

**template** <**class** Tclass>
**void ari::DelegateFiveParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2, Targ**

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2, Targ3, Targ4, Targ5)

*BaseFuncFiveParam* ***m_pMemFun**

### Class DelegateFiveParam::BaseFuncFiveParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateFiveParam*.

### Class Documentation

**class BaseFuncFiveParam**

#### Public Functions

template<>
**virtual ~BaseFuncFiveParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*) = 0

### Template Class DelegateFiveParam::MemFuncFiveParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateFiveParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateFiveParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5 >::BaseFuncFiveParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncFiveParam`** : **`public`** ari::DelegateFiveParam<Treturn, Targ1, Targ2, Targ3, Targ4, Targ5>::BaseFuncFiveParam

#### Public Functions

**`ari::DelegateFiveParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5 >::MemFuncFiveParam`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**) (Targ1, Targ2, Targ3, Targ4, Targ5)

### Template Class DelegateFourParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateFourParam::BaseFuncFourParam*
- *Template Class DelegateFourParam::MemFuncFourParam*

### Class Documentation

**template** <**class** Treturn, **class** Targ1, **class** Targ2, **class** Targ3, **class** Targ4>
**class** **DelegateFourParam**

#### Public Functions

**DelegateFourParam**()

**~DelegateFourParam**()

void **Bind**(Treturn (*_*fun*)) Targ1, Targ2, Targ3, Targ4

**template** <**class** Tclass>
**void** **ari::DelegateFourParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2, Targ**

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2, Targ3, Targ4)

*BaseFuncFourParam* ***m_pMemFun**

### Class DelegateFourParam::BaseFuncFourParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateFourParam*.

### Class Documentation

**class** **BaseFuncFourParam**

#### Public Functions

template<>
**virtual** **~BaseFuncFourParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*) = 0

### Template Class DelegateFourParam::MemFuncFourParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateFourParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateFourParam< Treturn, Targ1, Targ2, Targ3, Targ4 >::BaseFuncFourParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncFourParam`** : **`public`** ari::DelegateFourParam<Treturn, Targ1, Targ2, Targ3, Targ4>::BaseFuncFourParam

#### Public Functions

**`ari::DelegateFourParam< Treturn, Targ1, Targ2, Targ3, Targ4 >::MemFuncFourParam::MemFu`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**)(Targ1, Targ2, Targ3, Targ4)

### Template Class DelegateNineParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateNineParam::BaseFuncNineParam*
- *Template Class DelegateNineParam::MemFuncNineParam*

### Class Documentation

**template** <**class** Treturn, **class** Targ1, **class** Targ2, **class** Targ3, **class** Targ4, **class** Targ5, **class** Targ6, **class** Targ7
**class DelegateNineParam**

#### Public Functions

**DelegateNineParam**()

**~DelegateNineParam**()

void **Bind**(Treturn (*_*fun*)) Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Targ9

**template** <**class** Tclass>
**void ari::DelegateNineParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2, Targ**

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*, Targ9 *arg9*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Targ9)

*BaseFuncNineParam* ***m_pMemFun**

### Class DelegateNineParam::BaseFuncNineParam

- Defined in *File Delegate.hpp*

#### Nested Relationships

This class is a nested type of *Template Class DelegateNineParam*.

### Class Documentation

**class BaseFuncNineParam**

#### Public Functions

template<>
**virtual ~BaseFuncNineParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*, Targ9 *arg9*) = 0

### Template Class DelegateNineParam::MemFuncNineParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateNineParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateNineParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Targ9 >::BaseFuncNineParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncNineParam`** : **`public`** ari::DelegateNineParam<Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, T

#### Public Functions

**`ari::DelegateNineParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*, Targ9 *arg9*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Targ9)

### Template Class DelegateNoParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateNoParam::BaseFuncNoParam*
- *Template Class DelegateNoParam::MemFuncNoParam*

### Class Documentation

**template** <class *Treturn*>
**class DelegateNoParam**

#### Public Functions

**DelegateNoParam**()

**~DelegateNoParam**()

void **Bind**(Treturn (*_*fun*))

**template <class** Tclass>
**void ari::DelegateNoParam::Bind(Tclass * _obj, Treturn(Tclass::*)() _fun)**

bool **IsBound**()

Treturn **Execute**()

#### Protected Attributes

template<>
Treturn (***m_pFun**)()

*BaseFuncNoParam* ***m_pMemFun**

### Class DelegateNoParam::BaseFuncNoParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateNoParam*.

### Class Documentation

**class BaseFuncNoParam**

#### Public Functions

template<>
**virtual ~BaseFuncNoParam**()

template<>
**virtual** Treturn **Call**() = 0

### Template Class DelegateNoParam::MemFuncNoParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateNoParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateNoParam< Treturn >::BaseFuncNoParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncNoParam`** : **`public`** ari::DelegateNoParam<Treturn>::BaseFuncNoParam

#### Public Functions

**`ari::DelegateNoParam< Treturn >::MemFuncNoParam::MemFuncNoParam(Tclass * _obj, Treturn`**

template<>
Treturn **`Call`**()

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**)()

### Template Class DelegateOneParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateOneParam::BaseFuncOneParam*
- *Template Class DelegateOneParam::MemFuncOneParam*

### Class Documentation

**template** <class *Treturn*, class *Targ1*>
**class DelegateOneParam**

#### Public Functions

**DelegateOneParam**()

**~DelegateOneParam**()

void **Bind**(Treturn (*_*fun*)) Targ1

```
template <class Tclass>
void ari::DelegateOneParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1) _fun)
```

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1)

*BaseFuncOneParam* ***m_pMemFun**

### Class DelegateOneParam::BaseFuncOneParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateOneParam*.

### Class Documentation

**class BaseFuncOneParam**

#### Public Functions

template<>
**virtual ~BaseFuncOneParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*) = 0

### Template Class DelegateOneParam::MemFuncOneParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateOneParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateOneParam< Treturn, Targ1 >::BaseFuncOneParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncOneParam`** : **`public`** ari::DelegateOneParam<Treturn, Targ1>::BaseFuncOneParam

#### Public Functions

**`ari::DelegateOneParam< Treturn, Targ1 >::MemFuncOneParam::MemFuncOneParam(Tclass * _ob`**

template<>
Treturn **`Call`**(Targ1 *arg1*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**) (Targ1)

### Template Class DelegateSevenParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateSevenParam::BaseFuncSevenParam*
- *Template Class DelegateSevenParam::MemFuncSevenParam*

### Class Documentation

**template** <**class** Treturn, **class** Targ1, **class** Targ2, **class** Targ3, **class** Targ4, **class** Targ5, **class** Targ6, **class** Targ7
**class DelegateSevenParam**

#### Public Functions

**DelegateSevenParam**()

**~DelegateSevenParam**()

void **Bind**(Treturn (*_*fun*)) Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7

**template** <**class** Tclass>
**void ari::DelegateSevenParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2, Tar**

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7)

*BaseFuncSevenParam* ***m_pMemFun**

### Class DelegateSevenParam::BaseFuncSevenParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateSevenParam*.

### Class Documentation

**class BaseFuncSevenParam**

#### Public Functions

template<>
**virtual ~BaseFuncSevenParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*) = 0

### Template Class DelegateSevenParam::MemFuncSevenParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateSevenParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateSevenParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7 >::BaseFuncSevenParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncSevenParam`** : **`public`** ari::DelegateSevenParam<Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7>::Bas

#### Public Functions

**`ari::DelegateSevenParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7 >::M`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7)

### Template Class DelegateSixParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateSixParam::BaseFuncSixParam*
- *Template Class DelegateSixParam::MemFuncSixParam*

### Class Documentation

**template** <**class** Treturn, **class** Targ1, **class** Targ2, **class** Targ3, **class** Targ4, **class** Targ5, **class** Targ6>
**class DelegateSixParam**

#### Public Functions

**DelegateSixParam**()

**~DelegateSixParam**()

void **Bind**(Treturn (*_*fun*)) Targ1, Targ2, Targ3, Targ4, Targ5, Targ6

**template** <**class** Tclass>
**void ari::DelegateSixParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2, Targ3**

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6)

*BaseFuncSixParam* ***m_pMemFun**

### Class DelegateSixParam::BaseFuncSixParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateSixParam*.

### Class Documentation

**class BaseFuncSixParam**

#### Public Functions

template<>
**virtual ~BaseFuncSixParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*) = 0

### Template Class DelegateSixParam::MemFuncSixParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateSixParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateSixParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6 >::BaseFuncSixParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncSixParam`** : **`public`** ari::DelegateSixParam<Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6>::BaseFuncSixPar

#### Public Functions

**`ari::DelegateSixParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6 >::MemFuncSix`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6)

### Template Class DelegateTenParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateTenParam::BaseFuncTenParam*
- *Template Class DelegateTenParam::MemFuncTenParam*

### Class Documentation

**template <class** Treturn, **class** Targ1, **class** Targ2, **class** Targ3, **class** Targ4, **class** Targ5, **class** Targ6, **class** Targ7
**class DelegateTenParam**

#### Public Functions

**DelegateTenParam**()

**~DelegateTenParam**()

void **Bind**(Treturn (*_*fun*)) Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Targ9, Targ10

**template <class** Tclass>
**void ari::DelegateTenParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2, Targ3**

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*, Targ9 *arg9*, Targ10 *arg10*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Targ9, Targ10)

*BaseFuncTenParam* ***m_pMemFun**

### Class DelegateTenParam::BaseFuncTenParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateTenParam*.

### Class Documentation

**class BaseFuncTenParam**

#### Public Functions

template<>
**virtual ~BaseFuncTenParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*, Targ9 *arg9*, Targ10 *arg10*) = 0

### Template Class DelegateTenParam::MemFuncTenParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateTenParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateTenParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Targ9, Targ10 >::BaseFuncTenParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncTenParam`** : **`public`** ari::DelegateTenParam<Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Tar

#### Public Functions

**`ari::DelegateTenParam< Treturn, Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*, Targ4 *arg4*, Targ5 *arg5*, Targ6 *arg6*, Targ7 *arg7*, Targ8 *arg8*, Targ9 *arg9*, Targ10 *arg10*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**) (Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8, Targ9, Targ10)

### Template Class DelegateThreeParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateThreeParam::BaseFuncThreeParam*
- *Template Class DelegateThreeParam::MemFuncThreeParam*

### Class Documentation

**template** <**class** Treturn, **class** Targ1, **class** Targ2, **class** Targ3>
**class DelegateThreeParam**

#### Public Functions

**DelegateThreeParam**()

**~DelegateThreeParam**()

void **Bind**(Treturn (*_*fun*)) Targ1, Targ2, Targ3

**template** <**class** Tclass>
**void ari::DelegateThreeParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2, Tar**

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2, Targ3)

*BaseFuncThreeParam* ***m_pMemFun**

### Class DelegateThreeParam::BaseFuncThreeParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateThreeParam*.

### Class Documentation

**class BaseFuncThreeParam**

#### Public Functions

template<>
**virtual ~BaseFuncThreeParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*) = 0

### Template Class DelegateThreeParam::MemFuncThreeParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateThreeParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateThreeParam< Treturn, Targ1, Targ2, Targ3 >::BaseFuncThreeParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncThreeParam`** : **`public`** ari::DelegateThreeParam<Treturn, Targ1, Targ2, Targ3>::BaseFuncThreeParam

#### Public Functions

**`ari::DelegateThreeParam< Treturn, Targ1, Targ2, Targ3 >::MemFuncThreeParam::MemFuncThr`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*, Targ3 *arg3*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**) (Targ1, Targ2, Targ3)

### Template Class DelegateTwoParam

- Defined in *File Delegate.hpp*

### Nested Relationships

### Nested Types

- *Class DelegateTwoParam::BaseFuncTwoParam*
- *Template Class DelegateTwoParam::MemFuncTwoParam*

### Class Documentation

**template** <class *Treturn*, class *Targ1*, class *Targ2*>
**class DelegateTwoParam**

#### Public Functions

**DelegateTwoParam**()

**~DelegateTwoParam**()

void **Bind**(Treturn (*_*fun*)) Targ1, Targ2

```
template <class Tclass>
void ari::DelegateTwoParam::Bind(Tclass * _obj, Treturn(Tclass::*)(Targ1, Targ2) _fun)
```

bool **IsBound**()

Treturn **Execute**(Targ1 *arg1*, Targ2 *arg2*)

#### Protected Attributes

template<>
Treturn (***m_pFun**) (Targ1, Targ2)

*BaseFuncTwoParam* ***m_pMemFun**

### Class DelegateTwoParam::BaseFuncTwoParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateTwoParam*.

### Class Documentation

**class BaseFuncTwoParam**

#### Public Functions

template<>
**virtual ~BaseFuncTwoParam**()

template<>
**virtual** Treturn **Call**(Targ1 *arg1*, Targ2 *arg2*) = 0

### Template Class DelegateTwoParam::MemFuncTwoParam

- Defined in *File Delegate.hpp*

### Nested Relationships

This class is a nested type of *Template Class DelegateTwoParam*.

### Inheritance Relationships

### Base Type

- `public ari::DelegateTwoParam< Treturn, Targ1, Targ2 >::BaseFuncTwoParam`

### Class Documentation

**`template <class`** Tclass>
**`class MemFuncTwoParam`** : **`public`** ari::DelegateTwoParam<Treturn, Targ1, Targ2>::BaseFuncTwoParam

#### Public Functions

**`ari::DelegateTwoParam< Treturn, Targ1, Targ2 >::MemFuncTwoParam::MemFuncTwoParam(Tclas`**

template<>
Treturn **`Call`**(Targ1 *arg1*, Targ2 *arg2*)

#### Protected Attributes

template<>
Tclass ***`m_pObj`**

template<>
template<>
Treturn (**`Tclass`**::***`m_pFun`**) (Targ1, Targ2)

### Class Dock

- Defined in *File Dock.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class Dock** : **public** ari::*Gui*

#### Public Functions

**Dock**()

**~Dock**()

bool **BeginRender**()

void **EndRender**()

#### Public Members

bool **isOpened**

char ***Label**

### Class DockableWindow

- Defined in *File DockableWindow.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class DockableWindow** : **public** ari::*Gui*

#### Public Types

**enum Oriention**

*Values:*

**Center**

**Top**

**Left**

**Right**

**Botton**

### Public Functions

**DockableWindow**(*GuiSystem* *_pGuiSystem)

**~DockableWindow**()

bool **BeginRender**()

void **Dock**(*Orientation* _orientation = *Orientation*::Center, float _raito = 0.5f) **const**

void **DockWith**(*DockableWindow* *_pOtherDock, *Orientation* _orientation = *Orientation*::Center, float _raito = 0.5f) **const**

void **SetTitle**(**const** char *_pTitle) **const**

void **SetAlone**(bool _alone) **const**

void **SetClosable**(bool _closable) **const**

void **SetFillingSpace**(bool _fill) **const**

void **GetLastPosition**(float &_x, float &_y) **const**

void **GetLastSize**(float &_width, float &_height) **const**

*PlatformWindow* ***GetPlatformWindow**() **const**

**Note** : only call this function on OnGui callback

### Public Members

*DelegateNoParam*<void> **OnGui**
This is a callback for when we want to draw the Guis good to get the window size here.

*DelegateNoParam*<void> **OnWindowChanged**
This callback is for when platform window change or assigned. Good for setting the event listeners.

### Protected Attributes

*GuiSystem* ***m_pGuiSystem**

ImWindow::ImwWindow ***m_pWindow**

*PlatformWindow* ***m_pPlatformWindow**

## Class DockSpace

- Defined in *File DockSpace.hpp*

## Inheritance Relationships

## Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class DockSpace** : **public** ari::*Gui*

#### Public Functions

bool **BeginRender**()

void **EndRender**()

### Class Engine

- Defined in *File Engine.hpp*

### Class Documentation

**class Engine**

#### Public Functions

**Engine**()
Constructor.

**~Engine**()
Destructor.

bool **Init**(*InitParams* *params)
Init the engine device.

bool **Run**()

void **LockUpdateThread**()

void **UnlockUpdateThread**()

*Event* ***Poll**()

void **Release**(**const** *Event* *_event)

uint32_t **GetCurrentFrameNumber**() **const**

std::shared_ptr<spdlog::logger> **GetLogger**() **const**

*InitParams* ***GetParams**() **const**

void **SetParams**(*InitParams* *_params)

*PlatformWindow* ***GetMainWindow**() **const**

*PlatformWindow* ***NewWindow**(*PlatformWindow*::*Type* _type)

uint16_t **GetNewViewId**()

uint32_t **GetMsaaFlags**() **const**

float **GetElapsedTime**() **const**

float **GetDeltaTime**() **const**

### Public Members

*PluginManager* **plugin_manager**

*TextureManager* **texture_manager**

### Public Static Functions

**static** *Engine* &**GetSingleton**()

### Protected Attributes

*InitParams* ***m_params**

std::shared_ptr<spdlog::logger> **Logger**

*PlatformWindow* ***m_pWindow**

uint32_t **m_debug**

uint32_t **m_reset**

uint32_t **m_frame_number**

uint16_t **m_viewId** = 0

int64_t **m_time_offset**

bx::Thread ***m_pGfxThread**

bx::Mutex ***m_pMutex** = nullptr

int **m_iLockStatus** = 0

ftl::TaskScheduler ***m_pTaskMgr**
0 = No action, 1 = Lock, 2 = Unlock the update thread.

*MouseState* **m_MouseState**

bool **m_bRun**

bool **m_bNeedReset**

float **m_fElapsedTime** = 0.0f

float **m_fDeltaTime** = 0.0f

### Protected Static Functions

**static** int **InitBgfxInThread**(bx::Thread **_thread*, void **_userData*)

## Class Entity

- Defined in *File Entity.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Node` (*Class Node*)

### Class Documentation

**class Entity** : **public** ari::*Node*

#### Public Functions

**Entity**()
Constuctor.

**~Entity**()
Destructor.

### Class EventQueue

- Defined in *File IoEvents.hpp*

### Class Documentation

**class EventQueue**

#### Public Functions

**EventQueue**()

**~EventQueue**()

void **postAxisEvent**(*GamepadHandle* *_gamepad*, *GamepadAxis*::*Enum* *_axis*, int32_t *_value*)

void **postCharEvent**(uint8_t *_len*, **const** uint8_t *_char*[4])

void **postExitEvent**()

void **postGamepadEvent**(*GamepadHandle* *_gamepad*, bool *_connected*)

void **postKeyEvent**(*Key*::*Enum* *_key*, uint8_t *_modifiers*, bool *_down*)

void **postMouseEvent**(int32_t *_mx*, int32_t *_my*, int32_t *_mz*)

void **postMouseEvent**(int32_t *_mx*, int32_t *_my*, int32_t *_mz*, *MouseButton*::*Enum* *_button*, bool *_down*)

void **postSizeEvent**(uint32_t *_width*, uint32_t *_height*)

void **postWindowEvent**(void *_nwh* = NULL)

void **postSuspendEvent**(*Suspend*::*Enum* *_state*)

void **postDropFileEvent**(**const** bx::FilePath &*_filePath*)

**const** *Event* ***poll**()

void **release**(**const** *Event* **_event*) **const**

### Template Class EventSubscriber

- Defined in *File EventSubscriber.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Internal::BaseEventSubscriber` (*Class BaseEventSubscriber*)

### Class Documentation

**template** <typename *T*>
**class EventSubscriber** : **public** ari::Internal::*BaseEventSubscriber*

Subclass this as EventSubscriber<EventType> and then call *World::subscribe()* in order to subscribe to events. Make sure to call *World::unsubscribe()* or *World::unsubscribeAll()* when your subscriber is deleted!

#### Public Functions

**virtual ~EventSubscriber**()

**virtual** void **Receive**(*World* **world*, **const** T &*event*) = 0

Called when an event is emitted by the world.

### Class FrameData

- Defined in *File FrameData.hpp*

### Class Documentation

**class FrameData**

#### Public Functions

**FrameData**()

#### Public Members

tinystl::vector<*Node3D* *> **Nodes**

tinystl::vector<Matrix> **WorldMatrices**

uint32_t **FrameNumber**

*Camera* ***Camera**

### Class Gui

- Defined in *File Gui.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Component` (*Class Component*)

### Derived Types

- `public ari::Button` (*Class Button*)
- `public ari::CheckBox` (*Class CheckBox*)
- `public ari::Dock` (*Class Dock*)
- `public ari::DockableWindow` (*Class DockableWindow*)
- `public ari::DockSpace` (*Class DockSpace*)
- `public ari::Image` (*Class Image*)
- `public ari::Label` (*Class Label*)
- `public ari::Popup` (*Class Popup*)
- `public ari::TextBox` (*Class TextBox*)
- `public ari::Window` (*Class Window*)
- `public shiva::ProjectGui` (*Class ProjectGui*)

### Class Documentation

**class Gui** : **public** ari::*Component*

Subclassed by *ari::Button*, *ari::CheckBox*, *ari::Dock*, *ari::DockableWindow*, *ari::DockSpace*, *ari::Image*, *ari::Label*, *ari::Popup*, *ari::TextBox*, *ari::Window*, *shiva::ProjectGui*

#### Public Functions

**Gui**()

**virtual ~Gui**()

**virtual** bool **BeginRender**()

**virtual** void **EndRender**()

#### Public Members

bool **SameLine** = false

bool **Separator** = false

bool **Visible** = true

### Class GuiSystem

- Defined in *File GuiSystem.hpp*

### Inheritance Relationships

### Base Types

- `public ari::System` (*Class System*)
- `public ari::EventSubscriber< events::OnComponentAssigned< Dock > >` (*Template Class EventSubscriber*)

### Class Documentation

**class GuiSystem** : **public** ari::*System*, **public** ari::*EventSubscriber*<events::*OnComponentAssigned*<*Dock*>>

#### Public Functions

**GuiSystem**()

**virtual ~GuiSystem**()

void **Update**(*World* *p_world, UpdateState *state*)
Update the system.

void **Configure**(*World* *p_world)
Configure the system after adding it to the world.

void **Unconfigure**(*World* *p_world)
Unconfigure the system before removing it from the world.

Type **GetSystemType**()
Returns the system type.

bool **NeedUpdateOnState**(UpdateState *state*)
Ask the system if needs update on different states.

void **Receive**(*World* *world, **const** events::*OnComponentAssigned*<*Dock*> &*event*)
Called when an event is emitted by the world.

#### Protected Functions

void **RenderGui**(*Node* *node)

#### Protected Attributes

bool **m_bIsDockCreated**

### Class Image

- Defined in *File Image.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class Image** : **public** ari::*Gui*

#### Public Functions

bool **BeginRender**()

#### Public Members

std::shared_ptr<*Texture*> **ImageTexture**

ImVec2 **Size**

*DelegateNoParam*<void> **OnHovered**

### Class BaseEventSubscriber

- Defined in *File EventSubscriber.hpp*

### Inheritance Relationships

### Derived Types

- `public ari::EventSubscriber< T >` (*Template Class EventSubscriber*)
- `public ari::EventSubscriber< events::OnComponentAssigned< BoxShape > >` (*Template Class EventSubscriber*)

- public ari::EventSubscriber< events::OnComponentAssigned< Camera > > (*Template Class EventSubscriber*)
- public ari::EventSubscriber< events::OnComponentAssigned< Dock > > (*Template Class EventSubscriber*)
- public ari::EventSubscriber< events::OnComponentRemoved< BoxShape > > (*Template Class EventSubscriber*)
- public ari::EventSubscriber< events::OnComponentRemoved< Camera > > (*Template Class EventSubscriber*)
- public ari::EventSubscriber< events::OnEntityCreated > (*Template Class EventSubscriber*)
- public ari::EventSubscriber< events::OnEntityDestroyed > (*Template Class EventSubscriber*)
- public ari::EventSubscriber< events::OnFrameData > (*Template Class EventSubscriber*)

### Class Documentation

**class BaseEventSubscriber**

Subclassed by *ari::EventSubscriber< T >*, *ari::EventSubscriber< events::OnComponentAssigned< BoxShape > >*, *ari::EventSubscriber< events::OnComponentAssigned< Camera > >*, *ari::EventSubscriber< events::OnComponentAssigned< Dock > >*, *ari::EventSubscriber< events::OnComponentRemoved< BoxShape > >*, *ari::EventSubscriber< events::OnComponentRemoved< Camera > >*, *ari::EventSubscriber< events::OnEntityCreated >*, *ari::EventSubscriber< events::OnEntityDestroyed >*, *ari::EventSubscriber< events::OnFrameData >*

#### Public Functions

**virtual ~BaseEventSubscriber**()

### Class IProgram

- Defined in *File Program.hpp*

### Class Documentation

**class IProgram**

#### Public Functions

**IProgram**(**const** char **programName*)

**virtual ~IProgram**()

**virtual** void **Init**() = 0

**virtual** bool **Update**(uint32_t *frame_number*, float *elasped*) = 0

**virtual** int **Shutdown**() = 0

tinystl::string **GetProgramName**() **const**

#### Protected Attributes

tinystl::string **m_sProgramName**

### Class Label

- Defined in *File Label.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class Label** : **public** ari::*Gui*

#### Public Functions

**Label**()
Constructor.

**~Label**()

bool **BeginRender**()

#### Public Members

**const** char ***Text**

### Class Node

- Defined in *File Node.hpp*

### Inheritance Relationships

### Derived Types

- `public ari::Component` (*Class Component*)
- `public ari::Entity` (*Class Entity*)

### Class Documentation

**class Node**

Subclassed by *ari::Component*, *ari::Entity*

#### Public Types

**enum Type**

*Values:*

**Entity** = 0

**Component**

**Unknown**

#### Public Functions

**Node**()

Constructor.

**virtual ~Node**()

Destructor.

**template <class** T>
T ***AddChild**(T **child*)

Adds a node as child.

**template <class** T>
T ***GetChild**()

**template <class** T>
tinystl::vector<*Node* *> **GetChildren**()

**template <class** T>
void **RemoveChild**(T **child*)

Removes a child from this node.

**Parameters**

- `child`: The pointer to the child.

void **RemoveChildren**(bool *_delete* = false)

Removes all children of this node.

**virtual** *Node* ***GetParent**()

Returns the node parent.

**virtual** void **SetParent**(*Node* **parent*)

Sets the node parent.

*Node*::*Type* **GetType**() **const**

Returns the node type.

*Entity* ***GetParentEntity**() **const**

Returns the parent *Entity* in the tree.

**const** tinystl::vector<*Node* *> &**GetChildren**() **const**

*World* ***GetWorld**() **const***

void **Destroy**(bool *addToDestroyQueue* = true)
Send the node to the destroy queue. It will be deleted in next two frame;

uint32_t **IsInDestroyQueue**() **const**

#### Protected Attributes

*Node* ***m_pParent**

tinystl::vector<*Node* *> **m_vChilds**

*Node*::*Type* **m_eNodeType**

*World* ***m_pWorld**

uint32_t **m_iIsInDestroyQueue** = 0

std::unordered_map<*TypeIndex*, tinystl::vector<*Node* *>> **childs**

### Class Node3D

- Defined in *File Node3D.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Component` (*Class Component*)

### Derived Types

- `public ari::BoxShape` (*Class BoxShape*)
- `public ari::Camera` (*Class Camera*)

### Class Documentation

**class Node3D** : **public** ari::*Component*
Subclassed by *ari::BoxShape*, *ari::Camera*

#### Public Functions

**Node3D**()
Constructor.

**virtual ~Node3D**()
Destructor.

**virtual** void **Render**(**const** Matrix &*matrix*, bgfx::Encoder **encoder*, uint16_t *_view_id*)
Render.

#### Public Members

*Vector3* **Position**

*Vector3* **Rotation**

*Vector3* **Scale**

Matrix **_finalMat**

bool **_isRenderable**

### Class PlatformWindow

- Defined in *File PlatformWindow.hpp*

### Class Documentation

**class PlatformWindow**

#### Public Types

**enum Type**

*Values:*

**Main**

**Child**

**Popup**

#### Public Functions

**PlatformWindow**(*Type* *_type*)

**virtual ~PlatformWindow**()

**virtual** bool **Init**(int *_posx*, int *_posy*, int *_width*, int *_height*, uint32_t *_flags*, **const** char *_*title*) = 0

**virtual** bool **Run**() = 0

**virtual** void **Show**(bool *_show*) = 0

**virtual** void **SetMousePos**(int *_x*, int *_y*) = 0

**virtual** void **SetTitle**(**const** char *_*title*) = 0

**virtual** void **SetFlags**(uint32_t *_flags*, bool *_addFlags* = false) = 0

**virtual** void **GetPos**(int &*_x*, int &*_y*) = 0

**virtual** void **SetPos**(int *_x*, int *_y*) = 0

**virtual** void **GetSize**(int &*_width*, int &*_height*)

**virtual** void **SetSize**(int *_width*, int *_height*) = 0

**virtual** void **SetAlpha**(unsigned char *_alpha*) = 0

**virtual** void **SetMouseLock**(bool *_lock*) = 0

**virtual** void **ToggleFrame**() = 0

**virtual** bool **IsWindowMaximized**() = 0

**virtual** void **SetWindowMaximized**(bool *_maximize*) = 0

**virtual** bool **IsWindowMinimized**() = 0

**virtual** void **SetWindowMinimized**(bool *_minimize*) = 0

**virtual** void ***GetHandle**() = 0

void **AddOnKeyDelegate**(*DelegateTwoParam*<void, *Key*::*Enum*, bool> **_pDelegate*)

void **RemoveOnKeyDelegate**(*DelegateTwoParam*<void, *Key*::*Enum*, bool> **_pDelegate*)

void **AddOnCharDelegate**(*DelegateTwoParam*<void, uint8_t, uint8_t *> **_pDelegate*)

void **RemoveOnCharDelegate**(*DelegateTwoParam*<void, uint8_t, uint8_t *> **_pDelegate*)

void **AddOnMouseButtonDelegate**(*DelegateTwoParam*<void, *MouseButton*::*Enum*, bool> **_pDelegate*)

void **RemoveOnMouseButtonDelegate**(*DelegateTwoParam*<void, *MouseButton*::*Enum*, bool> **_pDelegate*)

void **AddOnMouseMoveDelegate**(*DelegateTwoParam*<void, int, int> **_pDelegate*)

void **RemoveOnMouseMoveDelegate**(*DelegateTwoParam*<void, int, int> **_pDelegate*)

void **AddOnMouseWheelDelegate**(*DelegateOneParam*<void, int> **_pDelegate*)

void **RemoveOnMouseWheelDelegate**(*DelegateOneParam*<void, int> **_pDelegate*)

void **AddOnSizeDelegate**(*DelegateTwoParam*<void, int, int> **_pDelegate*)

void **RemoveOnSizeDelegate**(*DelegateTwoParam*<void, int, int> **_pDelegate*)

bool **ProcessEvents**(uint32_t &*_width*, uint32_t &*_height*, uint32_t &*_debug*, uint32_t &*_reset*, *MouseState* **_mouse*)

### Protected Attributes

*Type* **m_Type**

uint32_t **m_width**

uint32_t **m_height**

uint32_t **m_oldWidth**

uint32_t **m_oldHeight**

uint32_t **m_frameWidth**

uint32_t **m_frameHeight**

float **m_aspectRatio**

*EventQueue* **m_eventQueue**

tinystl::vector<*DelegateTwoParam*<void, *Key*::*Enum*, bool> *> **m_vOnKeys**

tinystl::vector<*DelegateTwoParam*<void, uint8_t, uint8_t *> *> **m_vOnChar**

tinystl::vector<*DelegateTwoParam*<void, *MouseButton*::*Enum*, bool> *> **m_vOnMouseButtons**

tinystl::vector<*DelegateTwoParam*<void, int, int> *> **m_vOnMouseMove**

tinystl::vector<*DelegateOneParam*<void, int> *> **m_vOnMouseWheel**

tinystl::vector<*DelegateTwoParam*<void, int, int> *> **m_vOnSize**

### Class Plugin

- Defined in *File Plugin.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Resource` (*Class Resource*)

### Class Documentation

**class Plugin** : **public** ari::*Resource*

#### Public Types

**enum Type**

*Values:*

**TextureLoader**

**MeshLoader**

**Unknown**

#### Public Functions

**Plugin**(**const** uint32_t &*_handel*, **const** std::string &*_fileName*)

**virtual ~Plugin**()

**virtual** void ***Create**() = 0

#### Protected Attributes

*Type* **m_eType** = Type::Unknown

### Class PluginManager

- Defined in *File PluginManager.hpp*

### Inheritance Relationships

### Base Type

- `public ari::ResourceManager< Plugin >` (*Template Class ResourceManager*)

### Class Documentation

**`class PluginManager`** : **`public`** ari::*ResourceManager*<*Plugin*>

#### Protected Functions

bool **`LoadResource`**(*Plugin* ***ppOut*, uint32_t *handle*, **`const`** std::string &*filename*, void **extra-Params*)

### Class Popup

- Defined in *File Popup.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**`class Popup`** : **`public`** ari::*Gui*

#### Public Functions

bool **`BeginRender`**()

void **`EndRender`**()

void **`Show`**()

void **`Hide`**()

#### Public Members

char ***Name** = nullptr
Name must be unique.

#### Protected Attributes

bool **m_bDoEndPopup** = false

bool **m_bOpenPopup** = false

bool **m_bClosePopup** = false

### Class RenderSystem

- Defined in *File RenderSystem.hpp*

### Inheritance Relationships

### Base Types

- `public ari::System` (*Class System*)
- `public ari::EventSubscriber< events::OnComponentAssigned< BoxShape > >` (*Template Class EventSubscriber*)
- `public ari::EventSubscriber< events::OnFrameData >` (*Template Class EventSubscriber*)

### Class Documentation

**class RenderSystem** : **public** ari::*System*, **public** ari::*EventSubscriber*<events::*OnComponentAssigned*<*BoxShape*>>, **publ**

#### Public Types

**enum VertexType**
*Values:*

**Pos**

**Color**

**Count**

#### Public Functions

**RenderSystem**()

**~RenderSystem**()

void **Update**(*World* *p_world*, UpdateState *state*)
Update the system.

void **Configure**(*World* *p_world)
Configure the system after adding it to the world.

void **Unconfigure**(*World* *p_world)
Unconfigure the system before removing it from the world.

Type **GetSystemType**()
Returns the system type.

bool **NeedUpdateOnState**(UpdateState *state*)
Ask the system if needs update on different states.

void **Receive**(*World* *world, **const** events::*OnComponentAssigned*<*BoxShape*> &*event*)
Called when an event is emitted by the world.

void **Receive**(*World* *world, **const** events::*OnFrameData* &*event*)
Called when an event is emitted by the world.

bgfx::VertexDecl ***GetVertexDecl**(*VertexType vertex_type*) **const**

bgfx::ProgramHandle ***GetProgram**() **const**

#### Protected Attributes

bgfx::VertexDecl ***m_pVertexDeclArray**

bgfx::ProgramHandle ***m_Program**

*FrameData* ***m_pFrameDataCurrent**

*FrameData* ***m_pFrameDataNext**

uint16_t **m_view_id** = 0

### Class Resource

- Defined in *File Resource.hpp*

### Inheritance Relationships

### Derived Types

- `public ari::Plugin` (*Class Plugin*)
- `public ari::Texture` (*Class Texture*)

### Class Documentation

**class Resource**
Subclassed by *ari::Plugin*, *ari::Texture*

#### Public Functions

**Resource**()

**Resource**(**const** uint32_t &*_handel*, **const** std::string &*_fileName*)

**virtual** **~Resource**()

uint32_t **GetHandel**() **const**

std::string **GetFileName**() **const**

#### Protected Attributes

uint32_t **m_iHandle**

std::string **m_sFileName**

### Class ResourceLoader

- Defined in *File ResourceLoader.hpp*

### Class Documentation

**class ResourceLoader**

#### Public Functions

**ResourceLoader**()
Constructor.

**virtual** **~ResourceLoader**()
Destructor.

**virtual** bool **IsALoadableFileExtension**(std::string *_extention*)
returns true if the file maybe is able to be loaded by this Loader based on the file extension (e.g. ".mesh")

**virtual** *Resource* ***LoadResource**(bx::FileReaderI **pStream*, uint32_t *_handle*, **const** std::string &*_filename*, void **_extraParams*) = 0
Loads a resource from a FileSystem and return its pointer.

**Return** Returns the created resource pointer. Note resource may not loaded yet.

**Parameters**

- `pStream`:
- `_extraParams`:

#### Protected Attributes

std::vector<std::string> **m_aFileExtension**
The file extension list that this loader is capable to load.

bool **m_bSwapEndian**
Swap the loaded data or not.

### Template Class ResourceManager

- Defined in *File ResourceManager.hpp*

### Class Documentation

**template** <class *T*>
**class ResourceManager**

#### Public Functions

**virtual ~ResourceManager**()

std::shared_ptr<T> **Load**(**const** std::string &*filename*, void **extraParams*)

void **AddLoader**(*ResourceLoader* **pLoader*)

uint32_t **GetNewHandle**()

std::shared_ptr<T> **AddResource**(T **_resource*)

#### Protected Functions

**virtual** bool **LoadResource**(T ***ppOut*, uint32_t *handle*, **const** std::string &*filename*, void **extraParams*) = 0

#### Protected Attributes

std::vector<std::shared_ptr<T>> **m_vResources**
Stores the resources

std::stack<uint32_t> **m_sHandles**
Stores the unused handles number

std::vector<*ResourceLoader* *> **m_vLoaders**
Stores the resource loaders.

### Class SceneSystem

- Defined in *File SceneSystem.hpp*

### Inheritance Relationships

### Base Types

- `public ari::System` (*Class System*)
- `public ari::EventSubscriber< events::OnEntityCreated >` (*Template Class EventSubscriber*)
- `public ari::EventSubscriber< events::OnEntityDestroyed >` (*Template Class EventSubscriber*)
- `public ari::EventSubscriber< events::OnComponentAssigned< Camera > >` (*Template Class EventSubscriber*)
- `public ari::EventSubscriber< events::OnComponentRemoved< Camera > >` (*Template Class EventSubscriber*)
- `public ari::EventSubscriber< events::OnComponentAssigned< BoxShape > >` (*Template Class EventSubscriber*)
- `public ari::EventSubscriber< events::OnComponentRemoved< BoxShape > >` (*Template Class EventSubscriber*)

### Class Documentation

**class SceneSystem** : **public** ari::*System*, **public** ari::*EventSubscriber*<events::*OnEntityCreated*>, **public** ari::*EventSubscri*

#### Public Functions

**SceneSystem**()
Constructor.

**~SceneSystem**()
Destructor.

void **Update**(*World* *p_world, UpdateState *state*)
Update the system.

void **Configure**(*World* *p_world)
Configure the system after adding it to the world.

void **Unconfigure**(*World* *p_world)
Unconfigure the system before removing it from the world.

Type **GetSystemType**()
Returns the system type.

bool **NeedUpdateOnState**(UpdateState *state*)
Ask the system if needs update on different states.

void **Receive**(*World* *world, **const** events::*OnEntityCreated* &*event*)
Called when an event is emitted by the world.

void **Receive**(*World* *world, **const** events::*OnEntityDestroyed* &*event*)
Called when an event is emitted by the world.

void **Receive**(*World* *world, **const** events::*OnComponentAssigned*<*Camera*> &event)
Called when an event is emitted by the world.

void **Receive**(*World* *world, **const** events::*OnComponentRemoved*<*Camera*> &event)
Called when an event is emitted by the world.

void **Receive**(*World* *world, **const** events::*OnComponentAssigned*<*BoxShape*> &event)
Called when an event is emitted by the world.

void **Receive**(*World* *world, **const** events::*OnComponentRemoved*<*BoxShape*> &event)
Called when an event is emitted by the world.

#### Protected Functions

void **CalcTransform**(*Node* *node, Matrix *parentMat)

#### Protected Attributes

*Camera* ***m_pActiveCamera**

*FrameData* ***m_FrameDatasUnused**

*FrameData* ***m_FrameDatasTransforms**

*FrameData* ***m_FrameDatasVisible**

### Class System

- Defined in *File System.hpp*

### Inheritance Relationships

### Derived Types

- `public ari::GuiSystem` (*Class GuiSystem*)
- `public ari::RenderSystem` (*Class RenderSystem*)
- `public ari::SceneSystem` (*Class SceneSystem*)

### Class Documentation

**class System**
Subclassed by *ari::GuiSystem*, *ari::RenderSystem*, *ari::SceneSystem*

#### Public Types

**enum Type**
*Values:*

**GameplaySystem**

**SceneSystem**

**RenderSystem**

**enum UpdateState**
*Values:*

**GameplayState**

**SceneManagerState**

**MainThreadState**

### Public Functions

**System**()
Constructor.

**virtual ~System**()
Destructor.

**virtual** void **Update**(*World* *p_world, *UpdateState* state) = 0
Update the system.

**virtual** void **Configure**(*World* *p_world) = 0
Configure the system after adding it to the world.

**virtual** void **Unconfigure**(*World* *p_world) = 0
Unconfigure the system before removing it from the world.

**virtual** *Type* **GetSystemType**() = 0
Returns the system type.

**virtual** bool **NeedUpdateOnState**(*UpdateState* state) = 0
Ask the system if needs update on different states.

## Class TextBox

- Defined in *File TextBox.hpp*

## Inheritance Relationships

## Base Type

- `public ari::Gui` (*Class Gui*)

## Class Documentation

**class TextBox** : **public** ari::*Gui*

#### Public Functions

**TextBox**(size_t *maxLength* = 128)

**~TextBox**()

bool **BeginRender**()

void **SetText**(**const** char *_*text*) **const**

#### Public Members

char ***Text**

char ***Label**

### Class Texture

- Defined in *File Texture.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Resource` (*Class Resource*)

### Class Documentation

**class Texture** : **public** ari::*Resource*

#### Public Functions

**Texture**()

**Texture**(**const** uint32_t &*_handel*, **const** std::string &*_fileName*)

**~Texture**()

#### Public Members

bgfx::TextureHandle **Handle** = BGFX_INVALID_HANDLE

### Class TextureManager

- Defined in *File TextureManager.hpp*

### Inheritance Relationships

### Base Type

- `public ari::ResourceManager< Texture >` (*Template Class ResourceManager*)

### Class Documentation

**class TextureManager** : **public** ari::*ResourceManager*<*Texture*>

#### Public Functions

**~TextureManager**()

#### Protected Functions

bool **LoadResource**(*Texture* ***ppOut*, uint32_t *handle*, **const** std::string &*filename*, void **extra-Params*)

### Class Viewport

- Defined in *File Viewport.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Component` (*Class Component*)

### Class Documentation

**class Viewport** : **public** ari::*Component*

#### Public Members

*RectI* **Rect**

bgfx::TextureFormat::Enum **TextureFormat** = bgfx::TextureFormat::Count

bool **CreateDepth** = false

bool **UseMSAA** = false

bgfx::FrameBufferHandle **m_frame_buffer_handle** = BGFX_INVALID_HANDLE

*RectI* **m_last_rect**

bgfx::ViewId **m_view_id** = 0

*Texture* **m_texture**

*Texture* **m_depth_texture**

### Class Window

- Defined in *File Window.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class Window** : **public** ari::*Gui*

#### Public Functions

**Window**()

**~Window**()

bool **BeginRender**()

void **EndRender**()

#### Public Members

char ***Name**

bool **CloseButton**

bool **isOpen**

ImVec2 **Pos**

ImVec2 **Size**

ImGuiWindowFlags **Flags**

### Class World

- Defined in *File World.hpp*

### Class Documentation

**class World**

### Public Types

**enum UpdateType**
*Values:*

**Sync**

**Async**

### Public Functions

**World**()
Constructor.

**~World**()
Destructor.

void **SetUpdateType**(*UpdateType* *type*)

*UpdateType* **GetUpdateType**() **const**

void **AddSystem**(*System* **p_system*)
Add a new system to the world

void **RemoveSystem**(*System* **p_system*)
Removes a system from world

void **AddEntity**(*Entity* **p_entity*)
Adds a new entity to the world

void **RemoveEntity**(*Entity* **p_entity*)
Removes an entity from world

void **Update**(float *tick*)
Updates the world

void **_AddToDestroyQueue**(*Node* **node*)
internal use *Node::Destroy()* instead. Add a node to destroy queue

**template** <**typename** T>
void **subscribe**(*EventSubscriber*<T> **subscriber*)
Subscribe to an event.

**template** <**typename** T>
void **unsubscribe**(*EventSubscriber*<T> **subscriber*)
Unsubscribe from an event.

void **unsubscribeAll**(void **subscriber*)
Unsubscribe from all events. Don't be afraid of the void pointer, just pass in your subscriber as normal.

**template** <**typename** T>
void **emit**(**const** T &*event*)
Emit an event. This will do nothing if there are no subscribers for the event type.

**const** tinystl::vector<*Entity* *> &**GetAllEntities**() **const**

ftl::TaskScheduler ***GetTaskScheduler**() **const**

#### Protected Functions

void **CheckDestroyQueue**()

#### Protected Attributes

std::unordered_map<*TypeIndex*, tinystl::vector<Internal::*BaseEventSubscriber* *>> **subscribers**

bx::SpScUnboundedQueueT<*Node*> **m_qDestroyQueue**

tinystl::vector<*System* *> **systems**

tinystl::vector<*Entity* *> **Entities**

ftl::TaskScheduler ***m_pTaskScheduler**

*UpdateType* **m_UpdateType**

### Class AssetBrowser

- Defined in *File AssetBrowser.hpp*

### Inheritance Relationships

### Base Type

- `public shiva::DockWindow` (*Class DockWindow*)

### Class Documentation

**class AssetBrowser** : **public** shiva::*DockWindow*

#### Public Functions

**~AssetBrowser**()

void **Init**(ari::*World* *p_world)

### Class DirectoryTree

- Defined in *File DirectoryTree.hpp*

### Class Documentation

**class DirectoryTree**

#### Public Functions

void **Update**()

#### Public Members

std::string **Name**

bx::FilePath **Path**

std::vector<*FileInfo*> **FileList**

std::vector<*DirectoryTree*> **Directories**

bool **IsRoot** = false

### Class DockWindow

- Defined in *File DockWindow.hpp*

### Inheritance Relationships

### Derived Types

- `public shiva::AssetBrowser` (*Class AssetBrowser*)
- `private shiva::ProjectBrowser` (*Class ProjectBrowser*)
- `public shiva::PropertyEditor` (*Class PropertyEditor*)
- `public shiva::Viewport` (*Class Viewport*)

### Class Documentation

**class DockWindow**

*DockWindow* is the base class for other windows in editor.

Subclassed by *shiva::AssetBrowser*, *shiva::ProjectBrowser*, *shiva::PropertyEditor*, *shiva::Viewport*

#### Public Functions

**virtual ~DockWindow**()

ari::*DockableWindow* ***GetDock**() **const**

**virtual** void **Init**(ari::*World* *p_world)

**virtual** void **Shutdown**()

#### Protected Attributes

ari::*Entity* ***m_pEntity** = nullptr

ari::*DockableWindow* ***m_pWindow** = nullptr

### Class Editor

- Defined in *File Editor.hpp*

### Class Documentation

**class Editor**

#### Public Functions

**Editor**()

**~Editor**()

void **Init**()

void **Update**(float *elasped*)

void **LoadProject**(*Project* **project*)

*Project* ***GetCurrentProject**() **const**

ari::*GuiSystem* ***GetGuiSystem**()

#### Protected Attributes

ari::*World* **m_EditorWorld**

ari::*GuiSystem* **m_GuiSystem**

ari::*RenderSystem* **m_RenderSystem**

ari::*SceneSystem* **m_SceneSystem**

*ProjectBrowser* **m_ProjectBrowser**

*EditorWindowManager* **m_EditorWindow**

*Project* ***m_pCurrentProject** = nullptr

### Class EditorSettings

- Defined in *File EditorSettings.hpp*

### Class Documentation

**class EditorSettings**

#### Public Members

std::string **LastProjectPath**

#### Public Static Functions

**static** *EditorSettings* &**Get**()

**static** void **Save**()

**static** void **Load**()

### Class EditorWindowManager

- Defined in *File EditorWindowManager.hpp*

### Class Documentation

**class EditorWindowManager**

#### Public Functions

**EditorWindowManager**()

**~EditorWindowManager**()

void **Init**(ari::*World* *pWorld)

void **Shutdown**()

#### Protected Attributes

ari::*Entity* ***m_pEntity** = nullptr

*AssetBrowser* ***m_pAssetBrowser** = nullptr

*Viewport* ***m_pViewport** = nullptr

*PropertyEditor* ***m_pPropertyEditor** = nullptr

### Class Project

- Defined in *File Project.hpp*

### Class Documentation

**class Project**

#### Public Functions

**Project**()

**~Project**()

void **Save**()

void **UpdateProjectTree**()

**const** *DirectoryTree* &**GetTree**() **const**

**const** bx::FilePath &**GetPath**() **const**

#### Public Static Functions

**static** *Project* ***New**(bx::FilePath *projectPath*, std::string *name*, bx::Error **err*)

**static** *Project* ***Load**(bx::FilePath *path*, bx::Error **err*)

### Class ProjectBrowser

- Defined in *File ProjectBrowser.hpp*

### Inheritance Relationships

### Base Type

- `private shiva::DockWindow` (*Class DockWindow*)

### Class Documentation

**class ProjectBrowser** : shiva::*DockWindow*

#### Public Functions

**ProjectBrowser**()

**~ProjectBrowser**()

void **Init**(ari::*World* **p_world*)

void **Shutdown**()

#### Protected Functions

void **OnNewProjectClick**()

void **OnOpenProjectClick**()

void **OnClickMbOk**()

void **ProjectOpened**(*Project* *project)

#### Protected Attributes

ari::*TextBox* ***m_pNewProjectName**

ari::*TextBox* ***m_pNewProjectPath**

ari::*Button* ***m_pNewProjectBtn**

ari::*TextBox* ***m_pOpenProjectPath**

ari::*Button* ***m_pOpenProjectBtn**

ari::*Popup* ***m_pMessageBox**

ari::*Label* ***m_pMbLabel**

ari::*Button* ***m_pMbOkBtn**

### Class ProjectGui

- Defined in *File ProjectBrowser.hpp*

### Inheritance Relationships

### Base Type

- `public ari::Gui` (*Class Gui*)

### Class Documentation

**class ProjectGui** : **public** ari::*Gui*
Custom GUI component to show a project in the list.

#### Public Functions

bool **BeginRender**()

void **EndRender**()

### Class PropertyEditor

- Defined in *File PropertyEditor.hpp*

### Inheritance Relationships

### Base Type

- `public shiva::DockWindow` (*Class DockWindow*)

### Class Documentation

**class PropertyEditor** : **public** shiva::*DockWindow*

#### Public Functions

void **Init**(ari::*World* *p_world)

### Class Viewport

- Defined in *File Viewport.hpp*

### Inheritance Relationships

### Base Type

- `public shiva::DockWindow` (*Class DockWindow*)

### Class Documentation

**class Viewport** : **public** shiva::*DockWindow*

#### Public Functions

void **Init**(ari::*World* *p_world)

#### Protected Functions

void **OnGui**()

void **OnHovered**()

#### Protected Attributes

ari::*Camera* ***m_pCamera** = nullptr

ari::*Viewport* ***m_pViewport** = nullptr

ari::*Image* ***m_pView** = nullptr

ari::*PlatformWindow* ***m_pPlatformWindow** = nullptr

ari::*DelegateTwoParam*<void, int, int> **m_OnMouseMove**

### 3.3.3 Functions

#### Function ari::BX_ALIGN_DECL_16

- Defined in *File Matrix.hpp*

#### Function Documentation

**ari::BX_ALIGN_DECL_16(struct)**

#### Function ari::getDefaultAllocator

- Defined in *File aridef.hpp*

#### Function Documentation

bx::AllocatorI *ari::**getDefaultAllocator**()

#### Function ari::getName

- Defined in *File IoEnums.hpp*

#### Function Documentation

**const** char *ari::**getName**(*Key*::*Enum* *_key*)

#### Template Function ari::getTypeIndex

- Defined in *File aridef.hpp*

#### Function Documentation

**template** <**typename** T>
*TypeIndex* ari::**getTypeIndex**()

#### Function ari::inputAddBindings

- Defined in *File Input.hpp*

#### Function Documentation

void ari::**inputAddBindings**(**const** char *_name*, **const** *InputBinding* *_bindings*)

### Function ari::inputChar

- Defined in *File Input.hpp*

### Function Documentation

void ari::**inputChar**(uint8_t *_len*, **const** uint8_t *_char*[4])
Adds single UTF-8 encoded character into input buffer.

### Function ari::inputCharFlush

- Defined in *File Input.hpp*

### Function Documentation

void ari::**inputCharFlush**()
Flush internal input buffer.

### Function ari::inputGetChar

- Defined in *File Input.hpp*

### Function Documentation

**const** uint8_t *ari::**inputGetChar**()
Returns single UTF-8 encoded character from input buffer.

### Function ari::inputGetGamepadAxis

- Defined in *File Input.hpp*

### Function Documentation

int32_t ari::**inputGetGamepadAxis**(*GamepadHandle* *_handle*, *GamepadAxis*::*Enum* *_axis*)

### Function ari::inputGetKeyState

- Defined in *File Input.hpp*

### Function Documentation

bool ari::**inputGetKeyState**(*Key*::*Enum* *_key*, uint8_t *_*modifiers* = NULL)

### Function ari::inputGetModifiersState

- Defined in *File Input.hpp*

### Function Documentation

uint8_t ari::**inputGetModifiersState**()

### Function ari::inputGetMouse

- Defined in *File Input.hpp*

### Function Documentation

void ari::**inputGetMouse**(float *_mouse*[3])

### Function ari::inputInit

- Defined in *File Input.hpp*

### Function Documentation

void ari::**inputInit**()

### Function ari::inputIsMouseLocked

- Defined in *File Input.hpp*

### Function Documentation

bool ari::**inputIsMouseLocked**()

### Function ari::inputProcess

- Defined in *File Input.hpp*

### Function Documentation

void ari::**inputProcess**()

### Function ari::inputRemoveBindings

- Defined in *File Input.hpp*

### Function Documentation

void ari::**inputRemoveBindings**(**const** char *_name*)

### Function ari::inputSetGamepadAxis

- Defined in *File Input.hpp*

### Function Documentation

void `ari::`**`inputSetGamepadAxis`**(*GamepadHandle* *_handle*, *GamepadAxis*::*Enum* *_axis*, int32_t *_value*)

### Function ari::inputSetKeyState

- Defined in *File Input.hpp*

### Function Documentation

void `ari::`**`inputSetKeyState`**(*Key*::*Enum* *_key*, uint8_t *_modifiers*, bool *_down*)

### Function ari::inputSetMouseButtonState

- Defined in *File Input.hpp*

### Function Documentation

void `ari::`**`inputSetMouseButtonState`**(*MouseButton*::*Enum* *_button*, uint8_t *_state*)

### Function ari::inputSetMouseLock

- Defined in *File Input.hpp*

### Function Documentation

void `ari::`**`inputSetMouseLock`**(bool *_lock*)

### Function ari::inputSetMousePos

- Defined in *File Input.hpp*

### Function Documentation

void `ari::`**`inputSetMousePos`**(int32_t *_mx*, int32_t *_my*, int32_t *_mz*)

### Function ari::inputSetMouseResolution

- Defined in *File Input.hpp*

### Function Documentation

void ari::**inputSetMouseResolution**(uint16_t *_width*, uint16_t *_height*)

### Function ari::inputShutdown

- Defined in *File Input.hpp*

### Function Documentation

void ari::**inputShutdown**()

### Function ari::isValid(WindowHandle)

- Defined in *File IoEnums.hpp*

### Function Documentation

bool ari::**isValid**(*WindowHandle* *_handle*)

### Function ari::isValid(GamepadHandle)

- Defined in *File IoEnums.hpp*

### Function Documentation

bool ari::**isValid**(*GamepadHandle* *_handle*)

### Function ari::poll()

- Defined in *File IoEvents.hpp*

### Function Documentation

**const** *Event* *ari::**poll**()

### Function ari::poll(WindowHandle)

- Defined in *File IoEvents.hpp*

### Function Documentation

**const** *Event* *ari::**poll**(*WindowHandle*)

### Function ari::release

- Defined in *File IoEvents.hpp*

### Function Documentation

void `ari::release`(**const** *Event* *_*event*)

### Function BX_ERROR_RESULT

- Defined in *File Project.hpp*

### Function Documentation

```
BX_ERROR_RESULT(SH_ERROR_NOT_EMPTY_DIRECTPRY, BX_MAKEFOURCC('s', 'h', 0, 0))
```

### Template Function castToString(const T&)

- Defined in *File StringCast.h*

### Function Documentation

**template** <**typename** T>
std::string **castToString**(**const** T &*value*)

### Function castToString(const bool&)

- Defined in *File StringCast.h*

### Function Documentation

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

### Function castToString(const int&)

- Defined in *File StringCast.h*

### Function Documentation

std::string **castToString**(**const** int &*value*)

### Function castToString(const float&)

- Defined in *File StringCast.h*

### Function Documentation

std::string **castToString**(**const** float &*value*)

### Function castToString(const std::string&)

- Defined in *File StringCast.h*

### Function Documentation

std::string **castToString**(**const** std::string &*value*)

### Template Function from_json(const json&, T&)

- Defined in *File JsonCast.h*

### Function Documentation

**template** <**typename** T>
void **from_json**(**const** *json* &*j*, T &*obj*)

### Template Function from_json(const json&, T&)

- Defined in *File JsonCast.inl*

### Function Documentation

**template** <**typename** T>
void **from_json**(**const** *json* &*j*, T &*obj*)

### Template Function fromString(const std::string&)

- Defined in *File StringCast.h*

### Function Documentation

**template** <**typename** T>
T **fromString**(**const** std::string &*value*)

### Function fromString(const std::string&)

- Defined in *File StringCast.h*

### Function Documentation

**template** <**typename** T>
T **fromString**(**const** std::string &*value*)

### Function fromString(const std::string&)

- Defined in *File StringCast.h*

### Function Documentation

**template** <**typename** T>
T **fromString**(**const** std::string &*value*)

### Function fromString(const std::string&)

- Defined in *File StringCast.h*

### Function Documentation

**template** <**typename** T>
T **fromString**(**const** std::string &*value*)

### Function fromString(const std::string&)

- Defined in *File StringCast.h*

### Function Documentation

**template** <**typename** T>
T **fromString**(**const** std::string &*value*)

### Template Function meta::deserialize(Class&, const json&)

- Defined in *File JsonCast.h*

### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function "meta::deserialize" with arguments (Class&, const json&) in doxygen xml output for project "Ariyana engine" from directory: doxyoutput/xml. Potential matches:

```
- template <typename Class, typename = std::enable_if_t<!meta::isRegistered<Class>
↪()>, typename = void>
  void meta::deserialize(Class&, const json&)
- template <typename Class, typename = std::enable_if_t<meta::isRegistered<Class>()>
↪>
  void meta::deserialize(Class&, const json&)
- template <typename Class>
  Class meta::deserialize(const json&)
- template <typename K, typename V>
  void meta::deserialize(std::unordered_map<K, V>&, const json&)
- template <typename T>
  void meta::deserialize(std::vector<T>&, const json&)
```

### Template Function meta::deserialize(Class&, const json&)

- Defined in *File JsonCast.h*

### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function "meta::deserialize" with arguments (Class&, const json&) in doxygen xml output for project "Ariyana engine" from directory: doxyoutput/xml. Potential matches:

```
- template <typename Class, typename = std::enable_if_t<!meta::isRegistered<Class>
↪()>, typename = void>
  void meta::deserialize(Class&, const json&)
- template <typename Class, typename = std::enable_if_t<meta::isRegistered<Class>()>
↪>
  void meta::deserialize(Class&, const json&)
- template <typename Class>
  Class meta::deserialize(const json&)
- template <typename K, typename V>
  void meta::deserialize(std::unordered_map<K, V>&, const json&)
- template <typename T>
  void meta::deserialize(std::vector<T>&, const json&)
```

### Template Function meta::deserialize(std::vector<T>&, const json&)

- Defined in *File JsonCast.h*

### Function Documentation

**template <typename** T>
void meta**::deserialize**(std::vector<T> &*obj*, **const** *json* &*object*)

### Template Function meta::deserialize(std::unordered_map<K, V>&, const json&)

- Defined in *File JsonCast.h*

### Function Documentation

**template** <**typename** K, **typename** V>
void meta::**deserialize**(std::unordered_map<K, V> &*obj*, **const** *json* &*object*)

### Template Function meta::deserialize(const json&)

- Defined in *File JsonCast.inl*

### Function Documentation

**template** <**typename** Class>
Class meta::**deserialize**(**const** *json* &*obj*)

### Function meta::registerMembers< shiva::EditorSettings >

- Defined in *File EditorSettings.hpp*

### Function Documentation

**Warning:** doxygenfunction: Cannot find function "meta::registerMembers< shiva::EditorSettings >" in doxygen xml output for project "Ariyana engine" from directory: doxyoutput/xml

### Function meta::registerMembers< shiva::Project >

- Defined in *File Project.hpp*

### Function Documentation

**Warning:** doxygenfunction: Cannot find function "meta::registerMembers< shiva::Project >" in doxygen xml output for project "Ariyana engine" from directory: doxyoutput/xml

### Template Function meta::serialize(const Class&)

- Defined in *File JsonCast.h*

### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function "meta::serialize" with arguments (const Class&) in doxygen xml output for project "Ariyana engine" from directory: doxyoutput/xml. Potential matches:

```
- template <typename Class, typename = std::enable_if_t <!meta::isRegistered<Class>
↪()>, typename = void>
  json meta::serialize(const Class&)
- template <typename Class, typename = std::enable_if_t <meta::isRegistered<Class>
↪()>>
  json meta::serialize(const Class&)
```

### Template Function meta::serialize(const Class&)

- Defined in *File JsonCast.h*

### Function Documentation

**Warning:** doxygenfunction: Unable to resolve multiple matches for function "meta::serialize" with arguments (const Class&) in doxygen xml output for project "Ariyana engine" from directory: doxyoutput/xml. Potential matches:

```
- template <typename Class, typename = std::enable_if_t <!meta::isRegistered<Class>
↪()>, typename = void>
  json meta::serialize(const Class&)
- template <typename Class, typename = std::enable_if_t <meta::isRegistered<Class>
↪()>>
  json meta::serialize(const Class&)
```

### Template Function meta::serialize_basic(const Class&)

- Defined in *File JsonCast.h*

### Function Documentation

**template** <**typename** Class>
*json* meta::**serialize_basic**(**const** Class &*obj*)

### Template Function meta::serialize_basic(const std::vector<T>&)

- Defined in *File JsonCast.h*

### Function Documentation

**template** <**typename** T>
*json* meta::**serialize_basic**(**const** std::vector<T> &*obj*)

### Template Function meta::serialize_basic(const std::unordered_map<K, V>&)

- Defined in *File JsonCast.h*

#### Function Documentation

**template** <**typename** K, **typename** V>
*json* meta::**serialize_basic**(**const** std::unordered_map<K, V> &*obj*)

#### Template Function to_json(json&, const T&)

- Defined in *File JsonCast.h*

#### Function Documentation

**template** <**typename** T>
void **to_json**(*json* &*j*, **const** T &*obj*)

#### Template Function to_json(json&, const T&)

- Defined in *File JsonCast.inl*

#### Function Documentation

**template** <**typename** T>
void **to_json**(*json* &*j*, **const** T &*obj*)

### 3.3.4 Variables

#### Variable ari::fDegToRad

- Defined in *File arimath.hpp*

#### Variable Documentation

**const** float ari::**fDegToRad** = 0.0174532925f

#### Variable ari::fEpsilon

- Defined in *File arimath.hpp*

#### Variable Documentation

**const** float ari::**fEpsilon** = 0.000001f

#### Variable ari::fRadToDeg

- Defined in *File arimath.hpp*

### Variable Documentation

**const** float ari::**fRadToDeg** = 57.295779513f

### Variable ari::g_pEngine

- Defined in *File Engine.hpp*

### Variable Documentation

*Engine* *ari::**g_pEngine**

### Variable ari::PI

- Defined in *File arimath.hpp*

### Variable Documentation

**const** float ari::**PI** = 3.141592654f
P number.

### Variable ari::PiOver2

- Defined in *File arimath.hpp*

### Variable Documentation

**const** float ari::**PiOver2** = 1.570796326f

### Variable ari::TwoPI

- Defined in *File arimath.hpp*

### Variable Documentation

**const** float ari::**TwoPI** = 6.283185307f

### Variable g_allocator

- Defined in *File IoEvents.hpp*

### Variable Documentation

bx::AllocatorI ***g_allocator**

#### Variable shiva::g_pEditor

- Defined in *File Editor.hpp*

#### Variable Documentation

*Editor* *shiva::**g_pEditor**

### 3.3.5 Defines

#### Define ARI_API

- Defined in *File aridef.hpp*

#### Define Documentation

**ARI_API**

#### Define ARI_CONFIG_MAX_WINDOW

- Defined in *File aridef.hpp*

#### Define Documentation

**ARI_CONFIG_MAX_WINDOW**

#### Define ARI_DECLARE_TYPE

- Defined in *File aridef.hpp*

#### Define Documentation

**ARI_DECLARE_TYPE**

#### Define ARI_DEFINE_TYPE

- Defined in *File aridef.hpp*

#### Define Documentation

**ARI_DEFINE_TYPE**(name)

#### Define ARI_EXPORT_DLL

- Defined in *File aridef.hpp*

### Define Documentation

**ARI_EXPORT_DLL**

### Define ARI_IMPORT_DLL

- Defined in *File aridef.hpp*

### Define Documentation

**ARI_IMPORT_DLL**

### Define ARI_PLUGIN_API

- Defined in *File aridef.hpp*

### Define Documentation

**ARI_PLUGIN_API**

### Define ARI_TYPE_IMPLEMENTATION

- Defined in *File aridef.hpp*

### Define Documentation

**ARI_TYPE_IMPLEMENTATION**

### Define ENTRY_CONFIG_MAX_GAMEPADS

- Defined in *File aridef.hpp*

### Define Documentation

**ENTRY_CONFIG_MAX_GAMEPADS**

### Define ENTRY_IMPLEMENT_EVENT

- Defined in *File IoEvents.hpp*

### Define Documentation

**ENTRY_IMPLEMENT_EVENT**(_class, _type)

### Define ENTRY_WINDOW_FLAG_ASPECT_RATIO

- Defined in *File aridef.hpp*

### Define Documentation

**ENTRY_WINDOW_FLAG_ASPECT_RATIO**

### Define ENTRY_WINDOW_FLAG_FRAME

- Defined in *File aridef.hpp*

### Define Documentation

**ENTRY_WINDOW_FLAG_FRAME**

### Define ENTRY_WINDOW_FLAG_NONE

- Defined in *File aridef.hpp*

### Define Documentation

**ENTRY_WINDOW_FLAG_NONE**

### Define INPUT_BINDING_END

- Defined in *File Input.hpp*

### Define Documentation

**INPUT_BINDING_END**

### Define SHIVA_API

- Defined in *File shivadef.hpp*

### Define Documentation

**SHIVA_API**

### Define TINYSTL_ALLOCATOR

- Defined in *File aridef.hpp*

#### Define Documentation

**TINYSTL_ALLOCATOR**

### 3.3.6 Typedefs

#### Typedef ari::InputBindingFn

- Defined in *File Input.hpp*

#### Typedef Documentation

**typedef** void (*ari::**InputBindingFn**)(**const** void *_userData)

#### Typedef ari::RectF

- Defined in *File Rect.hpp*

#### Typedef Documentation

**typedef** *Rect*<float> ari::**RectF**

#### Typedef ari::RectI

- Defined in *File Rect.hpp*

#### Typedef Documentation

**typedef** *Rect*<int> ari::**RectI**

#### Typedef ari::RectU16

- Defined in *File Rect.hpp*

#### Typedef Documentation

**typedef** *Rect*<uint16_t> ari::**RectU16**

#### Typedef ari::TypeIndex

- Defined in *File aridef.hpp*

#### Typedef Documentation

**typedef** std::type_index ari::**TypeIndex**

#### Typedef json

- Defined in *File JsonCast.h*

#### Typedef Documentation

**`using json`** = nlohmann::json

### 3.3.7 Directories

#### Directory include

*Directory path:* `include`

#### Subdirectories

- *Directory ari*
- *Directory shiva*

#### Directory ari

*Parent directory* (`include`)

*Directory path:* `include/ari`

#### Subdirectories

- *Directory en*
- *Directory gfx*
- *Directory io*
- *Directory math*

#### Files

- *File aridef.hpp*
- *File Delegate.hpp*
- *File Engine.hpp*
- *File JsonCast.h*
- *File JsonCast.inl*
- *File Plugin.hpp*
- *File PluginManager.hpp*
- *File Program.hpp*
- *File Resource.hpp*

- *File ResourceLoader.hpp*
- *File ResourceManager.hpp*
- *File StringCast.h*

### Directory en

*Parent directory* (`include/ari`)

*Directory path:* `include/ari/en`

### Subdirectories

- *Directory 2d*
- *Directory 3d*
- *Directory gui*

### Files

- *File Component.hpp*
- *File Entity.hpp*
- *File EventSubscriber.hpp*
- *File Node.hpp*
- *File System.hpp*
- *File World.hpp*
- *File WorldManager.hpp*

### Directory 2d

*Parent directory* (`include/ari/en`)

*Directory path:* `include/ari/en/2d`

### Files

- *File Viewport.hpp*

### Directory 3d

*Parent directory* (`include/ari/en`)

*Directory path:* `include/ari/en/3d`

### Files

- *File BoxShape.hpp*
- *File Camera.hpp*
- *File Node3D.hpp*
- *File RenderSystem.hpp*
- *File SceneSystem.hpp*

### Directory gui

*Parent directory* (`include/ari/en`)

*Directory path:* `include/ari/en/gui`

### Files

- *File Button.hpp*
- *File CheckBox.hpp*
- *File Dock.hpp*
- *File DockableWindow.hpp*
- *File DockSpace.hpp*
- *File Gui.hpp*
- *File GuiSystem.hpp*
- *File Image.hpp*
- *File Label.hpp*
- *File Popup.hpp*
- *File TextBox.hpp*
- *File Window.hpp*

### Directory gfx

*Parent directory* (`include/ari`)

*Directory path:* `include/ari/gfx`

### Files

- *File FrameData.hpp*
- *File Texture.hpp*
- *File TextureManager.hpp*
- *File Vertices.hpp*

### Directory io

*Parent directory* (`include/ari`)

*Directory path:* `include/ari/io`

### Files

- *File Input.hpp*
- *File IoEnums.hpp*
- *File IoEvents.hpp*
- *File PlatformWindow.hpp*

### Directory math

*Parent directory* (`include/ari`)

*Directory path:* `include/ari/math`

### Files

- *File arimath.hpp*
- *File Matrix.hpp*
- *File Rect.hpp*
- *File Vector.hpp*

### Directory shiva

*Parent directory* (`include`)

*Directory path:* `include/shiva`

### Subdirectories

- *Directory windows*

### Files

- *File DirectoryTree.hpp*
- *File Editor.hpp*
- *File EditorSettings.hpp*
- *File Project.hpp*
- *File shivadef.hpp*

#### Directory windows

*Parent directory* (`include/shiva`)

*Directory path:* `include/shiva/windows`

#### Files

- *File AssetBrowser.hpp*
- *File DockWindow.hpp*
- *File EditorWindowManager.hpp*
- *File PIE.hpp*
- *File ProjectBrowser.hpp*
- *File PropertyEditor.hpp*
- *File Tools.hpp*
- *File Viewport.hpp*

### 3.3.8 Files

#### File aridef.hpp

*Parent directory* (`include/ari`)

**Contents**



#### Definition (`include/ari/aridef.hpp`)

#### Program Listing for File aridef.hpp

*Return to documentation for file* (`include/ari/aridef.hpp`)

```
#pragma once

#if defined( _MSC_VER )
#   pragma warning(disable:4251) // dll interface for std types
#   define ARI_EXPORT_DLL __declspec(dllexport)
#   define ARI_IMPORT_DLL __declspec(dllimport)
#else
#   define ARI_EXPORT_DLL __attribute__((visibility("default")))
#   define ARI_IMPORT_DLL
#endif

#ifdef ARI_EXPORT
#   define ARI_API ARI_EXPORT_DLL
#else
#   define ARI_API ARI_IMPORT_DLL
#endif

#ifdef ARI_PLUGIN_EXPORT
#   define ARI_PLUGIN_API ARI_EXPORT_DLL
#else
#   define ARI_PLUGIN_API ARI_IMPORT_DLL
#endif

#define ARI_CONFIG_MAX_WINDOW          8
#define ENTRY_CONFIG_MAX_GAMEPADS      4
#define ENTRY_WINDOW_FLAG_NONE         UINT32_C(0x00000000)
#define ENTRY_WINDOW_FLAG_ASPECT_RATIO UINT32_C(0x00000001)
#define ENTRY_WINDOW_FLAG_FRAME        UINT32_C(0x00000002)

// Define ARI_NO_RTTI to turn off RTTI. This requires using the ARI_DEFINE_TYPE and␣
↪ARI_DECLARE_TYPE macros on all types
// that you wish to use as components or events. If you use ARI_NO_RTTI, also place␣
↪ARI_TYPE_IMPLEMENTATION in a single cpp file.
//#define ARI_NO_RTTI

#ifndef ARI_NO_RTTI

#include <typeindex>
#include <typeinfo>
#define ARI_TYPE_IMPLEMENTATION

#else

#define ARI_TYPE_IMPLEMENTATION \
    ari::TypeIndex ari::Internal::TypeRegistry::nextIndex = 1; \
    \
    ARI_DEFINE_TYPE(ari::Events::OnEntityCreated);\
    ARI_DEFINE_TYPE(ari::Events::OnEntityDestroyed); \

#endif

namespace bx
{
    struct AllocatorI;
}
namespace ari
{
```

(continues on next page)

(continued from previous page)

```
    ARI_API bx::AllocatorI* getDefaultAllocator();

    struct ARI_API TinyStlAllocator
    {
        static void* static_allocate(size_t _bytes);
        static void static_deallocate(void* _ptr, size_t /*_bytes*/);
    };

#ifndef ARI_NO_RTTI
    typedef std::type_index TypeIndex;

#define ARI_DECLARE_TYPE
#define ARI_DEFINE_TYPE(name)

    template<typename T>
    TypeIndex getTypeIndex()
    {
        return std::type_index(typeid(T));
    }

#else
    typedef uint32_t TypeIndex;

    namespace Internal
    {
        class TypeRegistry
        {
        public:
            TypeRegistry()
            {
                index = nextIndex;
                ++nextIndex;
            }

            TypeIndex getIndex() const
            {
                return index;
            }

        private:
            static TypeIndex nextIndex;
            TypeIndex index;
        };
    }

#define ARI_DECLARE_TYPE public: static ari::Internal::TypeRegistry __ari_type_reg
#define ARI_DEFINE_TYPE(name) ari::Internal::TypeRegistry name::__ari_type_reg

    template<typename T>
    TypeIndex getTypeIndex()
    {
        return T::__ari_type_reg.getIndex();
    }
#endif

} // namespace ari
```

(continues on next page)

(continued from previous page)

```
#   define TINYSTL_ALLOCATOR ari::TinyStlAllocator
```

### Includes

- typeindex
- typeinfo

### Included By

- *File Node.hpp*
- *File Texture.hpp*
- *File Resource.hpp*
- *File Matrix.hpp*
- *File System.hpp*
- *File EventSubscriber.hpp*
- *File FrameData.hpp*
- *File World.hpp*
- *File Engine.hpp*
- *File PlatformWindow.hpp*
- *File PluginManager.hpp*
- *File Input.hpp*
- *File ResourceLoader.hpp*

### Namespaces

- *Namespace ari*
- *Namespace bx*

### Classes

- *Struct TinyStlAllocator*

### Functions

- *Function ari::getDefaultAllocator*
- *Template Function ari::getTypeIndex*

### Defines

- *Define ARI_API*
- *Define ARI_CONFIG_MAX_WINDOW*
- *Define ARI_DECLARE_TYPE*
- *Define ARI_DEFINE_TYPE*
- *Define ARI_EXPORT_DLL*
- *Define ARI_IMPORT_DLL*
- *Define ARI_PLUGIN_API*
- *Define ARI_TYPE_IMPLEMENTATION*
- *Define ENTRY_CONFIG_MAX_GAMEPADS*
- *Define ENTRY_WINDOW_FLAG_ASPECT_RATIO*
- *Define ENTRY_WINDOW_FLAG_FRAME*
- *Define ENTRY_WINDOW_FLAG_NONE*
- *Define TINYSTL_ALLOCATOR*

### Typedefs

- *Typedef ari::TypeIndex*

### File arimath.hpp

*Parent directory* (`include/ari/math`)

**Contents**



### Definition (`include/ari/math/arimath.hpp`)

### Program Listing for File arimath.hpp

*Return to documentation for file* (`include/ari/math/arimath.hpp`)

```
#pragma once


namespace ari
{
```

(continues on next page)

(continued from previous page)

```
    const float PI          = 3.141592654f;
    const float TwoPI       = 6.283185307f;
    const float PiOver2     = 1.570796326f;

    const float fDegToRad = 0.0174532925f;
    const float fRadToDeg = 57.295779513f;

    const float fEpsilon  = 0.000001f;

} // ari
```

### Included By

- *File Vector.hpp*

### Namespaces

- *Namespace ari*

### Variables

- *Variable ari::fDegToRad*
- *Variable ari::fEpsilon*
- *Variable ari::fRadToDeg*
- *Variable ari::PI*
- *Variable ari::PiOver2*
- *Variable ari::TwoPI*

## File AssetBrowser.hpp

*Parent directory* (`include/shiva/windows`)

**Contents**

### Definition (include/shiva/windows/AssetBrowser.hpp)

### Program Listing for File AssetBrowser.hpp

*Return to documentation for file* (include/shiva/windows/AssetBrowser.hpp)

```
#pragma once
#include "shiva/shivadef.hpp"
#include "shiva/DirectoryTree.hpp"
#include "../../../src/editor/windows/AssetGui.hpp"
#include "ari/en/gui/DockableWindow.hpp"
#include "DockWindow.hpp"

namespace ari {
    class Button;
    class Dock;
    class DockSpace;
}

namespace shiva
{
    class AssetGui;

    class SHIVA_API AssetBrowser : public DockWindow
    {
    public:

        ~AssetBrowser();

        void Init(ari::World* p_world);

    private:

        void UpdateAssets(const DirectoryTree& _tree);
        static DirectoryTree* FindPathTree(DirectoryTree* _tree, const std::string& _
↪path);
        void OnDblClick(AssetGui* _sender);
        void OnRightClick(AssetGui* _sender);

        std::vector<AssetGui*>  m_vAssets;

    }; // AssetBrowser

} // shiva
```

### Includes

- ../../../src/editor/windows/AssetGui.hpp
- DockWindow.hpp (*File DockWindow.hpp*)
- ari/Delegate.hpp (*File Delegate.hpp*)
- ari/en/gui/DockableWindow.hpp (*File DockableWindow.hpp*)
- ari/en/gui/Gui.hpp (*File Gui.hpp*)
- ari/gfx/Texture.hpp (*File Texture.hpp*)

- shiva/DirectoryTree.hpp (*File DirectoryTree.hpp*)
- shiva/shivadef.hpp (*File shivadef.hpp*)

### Namespaces

- *Namespace ari*
- *Namespace shiva*

### Classes

- *Class AssetBrowser*

### File BoxShape.hpp

*Parent directory* (include/ari/en/3d)

**Contents**



### Definition (include/ari/en/3d/BoxShape.hpp)

### Program Listing for File BoxShape.hpp

*Return to documentation for file* (include/ari/en/3d/BoxShape.hpp)

```
#pragma once
#include "Node3D.hpp"

namespace bgfx
{
    struct VertexBufferHandle;
    struct IndexBufferHandle;
    struct ProgramHandle;
}

namespace ari
{
    class RenderSystem;

    class ARI_API BoxShape: public Node3D
    {
    public:
```

(continues on next page)

(continued from previous page)

```
        // Constructor
        BoxShape() { _isRenderable = true; }

        virtual ~BoxShape() = default;

        virtual void Render(const Matrix& matrix, bgfx::Encoder* encoder, uint16_t _
↪view_id) override;

        static void Init(RenderSystem* render_system);
        static void Shutdown();

        static bgfx::VertexBufferHandle m_sVBPos;
        static bgfx::VertexBufferHandle m_sVBColor;
        static bgfx::IndexBufferHandle  m_sIB;
        static bgfx::ProgramHandle      m_sProgram;

    }; // BoxShape
}
```

### Includes

- Node3D.hpp (*File Node3D.hpp*)

### Namespaces

- *Namespace ari*
- *Namespace bgfx*

### Classes

- *Class BoxShape*

### File Button.hpp

*Parent directory* (include/ari/en/gui)

**Contents**

### Definition (include/ari/en/gui/Button.hpp)

### Program Listing for File Button.hpp

*Return to documentation for file* (include/ari/en/gui/Button.hpp)

```
#pragma once
#include "Gui.hpp"
#include "../../Delegate.hpp"

namespace ari
{
    class ARI_API Button: public Gui
    {
    public:

        bool BeginRender() override;

        DelegateNoParam<void> OnClick;

        char* Label;
    };
} // ari
```

### Includes

- ../../Delegate.hpp
- Gui.hpp (*File Gui.hpp*)

### Included By

- *File ProjectBrowser.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class Button*

### File Camera.hpp

*Parent directory* (include/ari/en/3d)

**Contents**

- *Includes*
- *Namespaces*
- *Classes*

### Definition (`include/ari/en/3d/Camera.hpp`)

### Program Listing for File Camera.hpp

*Return to documentation for file* (include/ari/en/3d/Camera.hpp)

```
#pragma once

#include "Node3D.hpp"
#include "../../math/Matrix.hpp"

namespace ari
{
    class ARI_API Camera: public Node3D
    {
    public:

        Camera() : Up(0.0f, 1.0f, 0.0f), _isActive(false) { }

        virtual ~Camera() = default;

        Vector3 Target,
            Up,
            Right;
        Matrix _view,
            _proj;
        bool _isActive;

        void Rotate(float _angle, const Vector3& _axis);

        void RotateByMouse(int _x, int _y, float _speed);

        void MoveBF(const float& _speed);

        void MoveLR(const float& _speed);

        void MoveUD(const float& _speed);

    protected:

        float           m_fCurRotX  = 0.0f, // Current Rotation X
                        m_fLastRotX = 0.0f; // Last Rotation X

    }; // Camera

} // ari
```

### Includes

- `../../math/Matrix.hpp`
- `Node3D.hpp` (*File Node3D.hpp*)

### Namespaces

- *Namespace ari*

### Classes

- *Class Camera*

### File CheckBox.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/CheckBox.hpp`)

### Program Listing for File CheckBox.hpp

*Return to documentation for file* (`include/ari/en/gui/CheckBox.hpp`)

```
#pragma once
#include "Gui.hpp"

namespace ari
{
    class ARI_API CheckBox: public Gui
    {
    public:

        CheckBox();

        ~CheckBox() = default;

        bool BeginRender() override;

        bool        Checked;
        char    *   Label;
```

(continues on next page)

(continued from previous page)

```
    };

} // ari
```

### Includes

- `Gui.hpp` (*File Gui.hpp*)

### Namespaces

- *Namespace ari*

### Classes

- *Class CheckBox*

### File Component.hpp

*Parent directory* (`include/ari/en`)

**Contents**



### Definition (`include/ari/en/Component.hpp`)

### Program Listing for File Component.hpp

*Return to documentation for file* (`include/ari/en/Component.hpp`)

```
#pragma once
#include "Node.hpp"

namespace ari
{
    class ARI_API Component: public Node
    {
    public:

        Component();
```

(continues on next page)

(continued from previous page)

```
        virtual ~Component() = default;

        bool _isFromNode3D;
        bool _isFromGui;

    }; // Component

} // ari
```

### Includes

- `Node.hpp` (*File Node.hpp*)

### Included By

- *File Viewport.hpp*
- *File Node3D.hpp*
- *File Gui.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class Component*

### File Delegate.hpp

*Parent directory* (`include/ari`)

**Contents**

### Definition (include/ari/Delegate.hpp)

### Program Listing for File Delegate.hpp

*Return to documentation for file* (include/ari/Delegate.hpp)

```
#pragma once

namespace ari
{
    template <class Treturn>
    class DelegateNoParam
    {
        class BaseFuncNoParam
        {
        public:
            virtual ~BaseFuncNoParam() = default;

            virtual Treturn Call() = 0;
        };

        template <class Tclass>
        class MemFuncNoParam : public BaseFuncNoParam
        {
        public:

            MemFuncNoParam(Tclass* _obj, Treturn(Tclass::*_fun)()) : m_pObj(_obj), m_
↪pFun(_fun) {}

            Treturn Call() override
            {
                return (*m_pObj.*m_pFun)();
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)();
        };

    public:

        DelegateNoParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateNoParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)()) { m_pFun = _fun; }

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)())
        {
            m_pMemFun = new MemFuncNoParam<Tclass>(_obj, _fun);
        }
```

(continues on next page)

(continued from previous page)

```
        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute()
        {
            if (m_pFun)
                return m_pFun();
            if (m_pMemFun)
                return m_pMemFun->Call();
        }

    protected:
        Treturn(*m_pFun)();
        BaseFuncNoParam *   m_pMemFun;
    };



    template <class Treturn, class Targ1>
    class DelegateOneParam
    {
        class BaseFuncOneParam
        {
        public:
            virtual ~BaseFuncOneParam() = default;

            virtual Treturn Call(Targ1 arg1) = 0;
        };

        template <class Tclass>
        class MemFuncOneParam : public BaseFuncOneParam
        {
        public:

            MemFuncOneParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1)) : m_pObj(_
↪obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1) override
            {
                return (*m_pObj.*m_pFun)(arg1);
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1);
        };

    public:

        DelegateOneParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateOneParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1)) { m_pFun = _fun; }
```

(continues on next page)

(continued from previous page)

```
        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1))
        {
            m_pMemFun = new MemFuncOneParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1)
        {
            if (m_pFun)
                return m_pFun(arg1);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1);
        }

    protected:
        Treturn(*m_pFun)(Targ1);
        BaseFuncOneParam    *   m_pMemFun;
    };



    template <class Treturn, class Targ1, class Targ2>
    class DelegateTwoParam
    {
        class BaseFuncTwoParam
        {
        public:
            virtual ~BaseFuncTwoParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2) = 0;
        };

        template <class Tclass>
        class MemFuncTwoParam : public BaseFuncTwoParam
        {
        public:

            MemFuncTwoParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2)) : m_
↪pObj(_obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1, Targ2 arg2) override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2);
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1, Targ2);
        };

    public:

        DelegateTwoParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }
```

(continues on next page)

(continued from previous page)

```
        ~DelegateTwoParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2)) { m_pFun = _fun; }

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2))
        {
            m_pMemFun = new MemFuncTwoParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1, Targ2 arg2)
        {
            if (m_pFun)
                return m_pFun(arg1, arg2);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2);
        BaseFuncTwoParam    *   m_pMemFun;
    };



    template <class Treturn, class Targ1, class Targ2, class Targ3>
    class DelegateThreeParam
    {
        class BaseFuncThreeParam
        {
        public:
            virtual ~BaseFuncThreeParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3) = 0;
        };

        template <class Tclass>
        class MemFuncThreeParam : public BaseFuncThreeParam
        {
        public:

            MemFuncThreeParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2,␣
↪Targ3)) : m_pObj(_obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3) override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2, arg3);
            }

        protected:
```

(continues on next page)

(continued from previous page)

```
            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1, Targ2, Targ3);
        };

    public:

        DelegateThreeParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateThreeParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2, Targ3)) { m_pFun = _fun; }

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3))
        {
            m_pMemFun = new MemFuncThreeParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1, Targ2 arg2, Targ3 arg3)
        {
            if (m_pFun)
                return m_pFun(arg1, arg2, arg3);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2, arg3);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2, Targ3);
        BaseFuncThreeParam  *   m_pMemFun;
    };



    template <class Treturn, class Targ1, class Targ2, class Targ3, class Targ4>
    class DelegateFourParam
    {
        class BaseFuncFourParam
        {
        public:
            virtual ~BaseFuncFourParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4) = 0;
        };

        template <class Tclass>
        class MemFuncFourParam : public BaseFuncFourParam
        {
        public:

            MemFuncFourParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3,
↪ Targ4)) : m_pObj(_obj), m_pFun(_fun) {}
```

(continues on next page)

(continued from previous page)

```
            Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4) override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2, arg3, arg4);
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1, Targ2, Targ3, Targ4);
        };

    public:

        DelegateFourParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateFourParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2, Targ3, Targ4)) { m_pFun = _fun; }

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3, Targ4))
        {
            m_pMemFun = new MemFuncFourParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4)
        {
            if (m_pFun)
                return m_pFun(arg1, arg2, arg3, arg4);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2, arg3, arg4);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2, Targ3, Targ4);
        BaseFuncFourParam   *   m_pMemFun;
    };



    template <class Treturn, class Targ1, class Targ2, class Targ3, class Targ4,␣
↪class Targ5>
    class DelegateFiveParam
    {
        class BaseFuncFiveParam
        {
        public:
            virtual ~BaseFuncFiveParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4,␣
↪Targ5 arg5) = 0;
        };
```

(continues on next page)

(continued from previous page)

```
        template <class Tclass>
        class MemFuncFiveParam : public BaseFuncFiveParam
        {
        public:

            MemFuncFiveParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3,
↪ Targ4, Targ5)) : m_pObj(_obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5)␣
↪override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2, arg3, arg4, arg5);
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5);
        };

    public:

        DelegateFiveParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateFiveParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2, Targ3, Targ4, Targ5)) { m_pFun = _fun;␣
↪}

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3, Targ4,␣
↪Targ5))
        {
            m_pMemFun = new MemFuncFiveParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5)
        {
            if (m_pFun)
                return m_pFun(arg1, arg2, arg3, arg4, arg5);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2, arg3, arg4, arg5);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5);
        BaseFuncFiveParam   *   m_pMemFun;
    };
```

(continues on next page)

(continued from previous page)

```
    template <class Treturn, class Targ1, class Targ2, class Targ3, class Targ4,␣
↪class Targ5, class Targ6>
    class DelegateSixParam
    {
        class BaseFuncSixParam
        {
        public:
            virtual ~BaseFuncSixParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4,␣
↪Targ5 arg5, Targ6 arg6) = 0;
        };

        template <class Tclass>
        class MemFuncSixParam : public BaseFuncSixParam
        {
        public:

            MemFuncSixParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3,␣
↪Targ4, Targ5, Targ6)) : m_pObj(_obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6) override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2, arg3, arg4, arg5, arg6);
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6);
        };

    public:

        DelegateSixParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateSixParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6)) { m_pFun␣
↪= _fun; }

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3, Targ4,␣
↪Targ5, Targ6))
        {
            m_pMemFun = new MemFuncSixParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6)
        {
```

(continues on next page)

(continued from previous page)

```
            if (m_pFun)
                return m_pFun(arg1, arg2, arg3, arg4, arg5, arg6);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2, arg3, arg4, arg5, arg6);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6);
        BaseFuncSixParam    *   m_pMemFun;
    };



    template <class Treturn, class Targ1, class Targ2, class Targ3, class Targ4,␣
↪class Targ5, class Targ6, class Targ7>
    class DelegateSevenParam
    {
        class BaseFuncSevenParam
        {
        public:
            virtual ~BaseFuncSevenParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4,␣
↪Targ5 arg5, Targ6 arg6, Targ7 arg7) = 0;
        };

        template <class Tclass>
        class MemFuncSevenParam : public BaseFuncSevenParam
        {
        public:

            MemFuncSevenParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2,␣
↪Targ3, Targ4, Targ5, Targ6, Targ7)) : m_pObj(_obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6, Targ7 arg7) override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2, arg3, arg4, arg5, arg6, arg7);
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7);
        };

    public:

        DelegateSevenParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateSevenParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7)) {␣
↪m_pFun = _fun; }
```

(continues on next page)

(continued from previous page)

```
        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3, Targ4,␣
↪Targ5, Targ6, Targ7))
        {
            m_pMemFun = new MemFuncSevenParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6, Targ7 arg7)
        {
            if (m_pFun)
                return m_pFun(arg1, arg2, arg3, arg4, arg5, arg6, arg7);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2, arg3, arg4, arg5, arg6, arg7);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7);
        BaseFuncSevenParam  *   m_pMemFun;
    };



    template <class Treturn, class Targ1, class Targ2, class Targ3, class Targ4,␣
↪class Targ5, class Targ6, class Targ7, class Targ8>
    class DelegateEightParam
    {
        class BaseFuncEightParam
        {
        public:
            virtual ~BaseFuncEightParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4,␣
↪Targ5 arg5, Targ6 arg6, Targ7 arg7, Targ8 arg8) = 0;
        };

        template <class Tclass>
        class MemFuncEightParam : public BaseFuncEightParam
        {
        public:

            MemFuncEightParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2,␣
↪Targ3, Targ4, Targ5, Targ6, Targ7, Targ8)) : m_pObj(_obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6, Targ7 arg7, Targ8 arg8) override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2, arg3, arg4, arg5, arg6, arg7,␣
↪arg8);
            }

        protected:

            Tclass  *   m_pObj;
```

(continues on next page)

(continued from previous page)

```
            Treturn(Tclass::*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7,␣
↪Targ8);
        };

    public:

        DelegateEightParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateEightParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7,␣
↪Targ8)) { m_pFun = _fun; }

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3, Targ4,␣
↪Targ5, Targ6, Targ7, Targ8))
        {
            m_pMemFun = new MemFuncEightParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6, Targ7 arg7, Targ8 arg8)
        {
            if (m_pFun)
                return m_pFun(arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2, arg3, arg4, arg5, arg6, arg7,␣
↪arg8);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8);
        BaseFuncEightParam  *   m_pMemFun;
    };



    template <class Treturn, class Targ1, class Targ2, class Targ3, class Targ4,␣
↪class Targ5, class Targ6, class Targ7, class Targ8, class Targ9>
    class DelegateNineParam
    {
        class BaseFuncNineParam
        {
        public:
            virtual ~BaseFuncNineParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4,␣
↪Targ5 arg5, Targ6 arg6, Targ7 arg7, Targ8 arg8, Targ9 arg9) = 0;
        };

        template <class Tclass>
        class MemFuncNineParam : public BaseFuncNineParam
```

(continues on next page)

(continued from previous page)

```
        {
        public:

            MemFuncNineParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3,
↪ Targ4, Targ5, Targ6, Targ7, Targ8, Targ9)) : m_pObj(_obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6, Targ7 arg7, Targ8 arg8, Targ9 arg9) override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2, arg3, arg4, arg5, arg6, arg7,␣
↪arg8, arg9);
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7,␣
↪Targ8, Targ9);
        };

    public:

        DelegateNineParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateNineParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7,␣
↪Targ8, Targ9)) { m_pFun = _fun; }

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3, Targ4,␣
↪Targ5, Targ6, Targ7, Targ8, Targ9))
        {
            m_pMemFun = new MemFuncNineParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }

        Treturn Execute(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6, Targ7 arg7, Targ8 arg8, Targ9 arg9)
        {
            if (m_pFun)
                return m_pFun(arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8,
↪ arg9);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8,␣
↪Targ9);
        BaseFuncNineParam   *   m_pMemFun;
    };
```

(continues on next page)

(continued from previous page)

```
    template <class Treturn, class Targ1, class Targ2, class Targ3, class Targ4,␣
↪class Targ5, class Targ6, class Targ7, class Targ8, class Targ9, class Targ10>
    class DelegateTenParam
    {
        class BaseFuncTenParam
        {
        public:
            virtual ~BaseFuncTenParam() = default;

            virtual Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4,␣
↪Targ5 arg5, Targ6 arg6, Targ7 arg7, Targ8 arg8, Targ9 arg9, Targ10 arg10) = 0;
        };

        template <class Tclass>
        class MemFuncTenParam : public BaseFuncTenParam
        {
        public:

            MemFuncTenParam(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3,␣
↪Targ4, Targ5, Targ6, Targ7, Targ8, Targ9, Targ10)) : m_pObj(_obj), m_pFun(_fun) {}

            Treturn Call(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6, Targ7 arg7, Targ8 arg8, Targ9 arg9, Targ10 arg10) override
            {
                return (*m_pObj.*m_pFun)(arg1, arg2, arg3, arg4, arg5, arg6, arg7,␣
↪arg8, arg9, arg10);
            }

        protected:

            Tclass  *   m_pObj;
            Treturn(Tclass::*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7,␣
↪Targ8, Targ9, Targ10);
        };

    public:

        DelegateTenParam() : m_pFun(nullptr), m_pMemFun(nullptr) { }

        ~DelegateTenParam()
        {
            delete m_pMemFun;
        }

        void Bind(Treturn(*_fun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7,␣
↪Targ8, Targ9, Targ10)) { m_pFun = _fun; }

        template <class Tclass>
        void Bind(Tclass* _obj, Treturn(Tclass::*_fun)(Targ1, Targ2, Targ3, Targ4,␣
↪Targ5, Targ6, Targ7, Targ8, Targ9, Targ10))
        {
            m_pMemFun = new MemFuncTenParam<Tclass>(_obj, _fun);
        }

        bool IsBound() { return m_pFun || m_pMemFun; }
```

(continues on next page)

(continued from previous page)

```
        Treturn Execute(Targ1 arg1, Targ2 arg2, Targ3 arg3, Targ4 arg4, Targ5 arg5,␣
↪Targ6 arg6, Targ7 arg7, Targ8 arg8, Targ9 arg9, Targ10 arg10)
        {
            if (m_pFun)
                return m_pFun(arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9,␣
↪arg10);
            if (m_pMemFun)
                return m_pMemFun->Call(arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8,
↪ arg9, arg10);
        }

    protected:
        Treturn(*m_pFun)(Targ1, Targ2, Targ3, Targ4, Targ5, Targ6, Targ7, Targ8,␣
↪Targ9, Targ10);
        BaseFuncTenParam    *   m_pMemFun;
    };



}
```

### Included By

- *File Button.hpp*
- *File DockableWindow.hpp*
- *File Image.hpp*
- *File PlatformWindow.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Template Class DelegateEightParam*
- *Class DelegateEightParam::BaseFuncEightParam*
- *Template Class DelegateEightParam::MemFuncEightParam*
- *Template Class DelegateFiveParam*
- *Class DelegateFiveParam::BaseFuncFiveParam*
- *Template Class DelegateFiveParam::MemFuncFiveParam*
- *Template Class DelegateFourParam*
- *Class DelegateFourParam::BaseFuncFourParam*
- *Template Class DelegateFourParam::MemFuncFourParam*
- *Template Class DelegateNineParam*

### File DirectoryTree.hpp

*Parent directory* (`include/shiva`)

**Contents**

### Definition (include/shiva/DirectoryTree.hpp)

### Program Listing for File DirectoryTree.hpp

*Return to documentation for file* (include/shiva/DirectoryTree.hpp)

```
#pragma once
#include "shivadef.hpp"
#include <string>
#include "bx/file.h"
#include <vector>

namespace shiva
{
    struct FileInfo
    {
        std::string Name;
    };

    class SHIVA_API DirectoryTree
    {
    public:

        std::string Name;
        bx::FilePath Path;
        std::vector<FileInfo> FileList;
        std::vector<DirectoryTree> Directories;
        bool IsRoot = false;

        void Update();

    };

} // shiva
```

### Includes

- bx/file.h
- shivadef.hpp (*File shivadef.hpp*)
- string
- vector

### Included By

- *File Project.hpp*
- *File AssetBrowser.hpp*

### Namespaces

- *Namespace shiva*

### Classes

- *Struct FileInfo*
- *Class DirectoryTree*

### File Dock.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/Dock.hpp`)

### Program Listing for File Dock.hpp

*Return to documentation for file* (`include/ari/en/gui/Dock.hpp`)

```
#pragma once
#include "Gui.hpp"
#include "dear-imgui/imgui.h"

namespace ari
{
    class ARI_API Dock: public Gui
    {
    public:

        Dock();

        ~Dock() = default;

        bool BeginRender() override;

        void EndRender() override;

        bool    isOpened;
        char*   Label;

    }; // Dock

} // ari
```

### Includes

- `Gui.hpp` (*File Gui.hpp*)
- `dear-imgui/imgui.h`

### Namespaces

- *Namespace ari*

### Classes

- *Class Dock*

### File DockableWindow.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/DockableWindow.hpp`)

### Program Listing for File DockableWindow.hpp

*Return to documentation for file* (`include/ari/en/gui/DockableWindow.hpp`)

```
#pragma once
#include "Gui.hpp"
#include "../../Delegate.hpp"

namespace ImWindow
{
    class ImwWindow;
}

namespace ari
{
    class GuiSystem;
    class PlatformWindow;

    class ARI_API DockableWindow: public Gui
```

(continues on next page)

(continued from previous page)

```
    {
        friend class AriImwWindow;

    public:
        enum class Orientation
        {
            Center,
            Top,
            Left,
            Right,
            Botton
        };

        DockableWindow(GuiSystem* _pGuiSystem);
        ~DockableWindow() override;

        bool BeginRender() override;

        void Dock(Orientation _orientation = Orientation::Center, float _raito = 0.5f)␣
↪const;

        void DockWith(DockableWindow* _pOtherDock, Orientation _orientation =␣
↪Orientation::Center,
            float _raito = 0.5f) const;

        void SetTitle(const char* _pTitle) const;
        void SetAlone(bool _alone) const;
        void SetClosable(bool _closable) const;
        void SetFillingSpace(bool _fill) const;

        void GetLastPosition(float& _x, float& _y) const;
        void GetLastSize(float& _width, float& _height) const;

        DelegateNoParam<void> OnGui;

        DelegateNoParam<void> OnWindowChanged;

        PlatformWindow* GetPlatformWindow() const;

    protected:

        GuiSystem           *   m_pGuiSystem;
        ImWindow::ImwWindow *   m_pWindow;
        PlatformWindow      *   m_pPlatformWindow;

    };

} // ari
```

### Includes

- `../../Delegate.hpp`
- `Gui.hpp` (*File Gui.hpp*)

### Included By

- *File DockWindow.hpp*
- *File AssetBrowser.hpp*

### Namespaces

- *Namespace ari*
- *Namespace ImWindow*

### Classes

- *Class DockableWindow*

### File DockSpace.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/DockSpace.hpp`)

### Program Listing for File DockSpace.hpp

*Return to documentation for file* (`include/ari/en/gui/DockSpace.hpp`)

```
#pragma once
#include "Gui.hpp"

namespace ari
{
    class ARI_API DockSpace: public Gui
    {
    public:

        bool BeginRender() override;
        void EndRender() override;

    }; // DockSpace

} // ari
```

### Includes

- `Gui.hpp` (*File Gui.hpp*)

### Namespaces

- *Namespace ari*

### Classes

- *Class DockSpace*

### File DockWindow.hpp

*Parent directory* (`include/shiva/windows`)

**Contents**



### Definition (`include/shiva/windows/DockWindow.hpp`)

### Program Listing for File DockWindow.hpp

*Return to documentation for file* (`include/shiva/windows/DockWindow.hpp`)

```
#pragma once
#include "shiva/shivadef.hpp"
#include "ari/en/gui/DockableWindow.hpp"
#include "ari/en/Entity.hpp"

namespace shiva
{
    class SHIVA_API DockWindow
    {
    public:

        virtual ~DockWindow() = default;

        ari::DockableWindow* GetDock() const { return m_pWindow; }

        virtual void Init(ari::World* p_world);
```

(continues on next page)

(continued from previous page)

```
        virtual void Shutdown();

    protected:

        ari::Entity          *   m_pEntity = nullptr;
        ari::DockableWindow  *   m_pWindow = nullptr;

    };

} // shiva
```

### Includes

- `ari/en/Entity.hpp` (*File Entity.hpp*)
- `ari/en/gui/DockableWindow.hpp` (*File DockableWindow.hpp*)
- `shiva/shivadef.hpp` (*File shivadef.hpp*)

### Included By

- *File ProjectBrowser.hpp*
- *File AssetBrowser.hpp*
- *File PropertyEditor.hpp*
- *File Viewport.hpp*

### Namespaces

- *Namespace shiva*

### Classes

- *Class DockWindow*

### File Editor.hpp

*Parent directory* (`include/shiva`)

**Contents**

### Definition (include/shiva/Editor.hpp)

### Program Listing for File Editor.hpp

*Return to documentation for file* (include/shiva/Editor.hpp)

```
#pragma once
#include "shivadef.hpp"
#include "ari/en/World.hpp"
#include "ari/en/gui/GuiSystem.hpp"
#include "ari/en/3d/RenderSystem.hpp"
#include "ari/en/3d/SceneSystem.hpp"
#include "windows/ProjectBrowser.hpp"
#include "windows/EditorWindowManager.hpp"

namespace shiva
{
    class Project;

    class SHIVA_API Editor
    {
    public:

        Editor();

        ~Editor();

        void Init();

        void Update(float elasped);

        void LoadProject(Project* project);

        Project* GetCurrentProject() const { return m_pCurrentProject; }

        ari::GuiSystem* GetGuiSystem() { return &m_GuiSystem; }

    protected:

        ari::World          m_EditorWorld;
        ari::GuiSystem      m_GuiSystem;
        ari::RenderSystem   m_RenderSystem;
        ari::SceneSystem    m_SceneSystem;
        ProjectBrowser      m_ProjectBrowser;
        EditorWindowManager m_EditorWindow;
        Project         *   m_pCurrentProject = nullptr;

    }; // Editor

    extern SHIVA_API Editor* g_pEditor;

} // shiva
```

### Includes

- ari/en/3d/RenderSystem.hpp (*File RenderSystem.hpp*)

- ari/en/3d/SceneSystem.hpp (*File SceneSystem.hpp*)
- ari/en/World.hpp (*File World.hpp*)
- ari/en/gui/GuiSystem.hpp (*File GuiSystem.hpp*)
- shivadef.hpp (*File shivadef.hpp*)
- windows/EditorWindowManager.hpp (*File EditorWindowManager.hpp*)
- windows/ProjectBrowser.hpp (*File ProjectBrowser.hpp*)

### Namespaces

- *Namespace shiva*

### Classes

- *Class Editor*

### Variables

- *Variable shiva::g_pEditor*

### File EditorSettings.hpp

*Parent directory* (include/shiva)

**Contents**



### Definition (include/shiva/EditorSettings.hpp)

### Program Listing for File EditorSettings.hpp

*Return to documentation for file* (include/shiva/EditorSettings.hpp)

```
#pragma once
#include <Meta.h>

namespace shiva
{
    class EditorSettings
```

(continues on next page)

(continued from previous page)

```
    {
    public:

        std::string         LastProjectPath;

        static EditorSettings& Get();
        static void Save();
        static void Load();
    };

} // shiva

namespace meta {

    template <>
    inline auto registerMembers<shiva::EditorSettings>()
    {
        return members(
            member("last_project_path", &shiva::EditorSettings::LastProjectPath)
        );
    }

} // end of namespace meta
```

### Includes

- Meta.h

### Namespaces

- *Namespace meta*
- *Namespace shiva*

### Classes

- *Class EditorSettings*

### Functions

- *Function meta::registerMembers< shiva::EditorSettings >*

### File EditorWindowManager.hpp

*Parent directory* (include/shiva/windows)

**Contents**

- *Definition (include/shiva/windows/EditorWindowManager.hpp)*

- *Includes*
- *Included By*
- *Namespaces*
- *Classes*

### Definition (include/shiva/windows/EditorWindowManager.hpp)

### Program Listing for File EditorWindowManager.hpp

*Return to documentation for file* (include/shiva/windows/EditorWindowManager.hpp)

```
#pragma once
#include "shiva/shivadef.hpp"

namespace ari
{
    class World;
    class Entity;
    class DockSpace;
    class Window;
}

namespace shiva
{
    class AssetBrowser;
    class Viewport;
    class Project;
    class PropertyEditor;

    class SHIVA_API EditorWindowManager
    {
    public:

        EditorWindowManager();

        ~EditorWindowManager();

        void Init(ari::World* pWorld);

        void Shutdown();

    protected:

        ari::Entity         *   m_pEntity           = nullptr;
        AssetBrowser        *   m_pAssetBrowser     = nullptr;
        Viewport            *   m_pViewport         = nullptr;
        PropertyEditor      *   m_pPropertyEditor   = nullptr;

    }; // EditorWindowManager

} // shiva
```

### Includes

- shiva/shivadef.hpp (*File shivadef.hpp*)

### Included By

- *File Editor.hpp*

### Namespaces

- *Namespace ari*
- *Namespace shiva*

### Classes

- *Class EditorWindowManager*

### File Engine.hpp

*Parent directory* (include/ari)

**Contents**



### Definition (include/ari/Engine.hpp)

### Program Listing for File Engine.hpp

*Return to documentation for file* (include/ari/Engine.hpp)

```
#pragma once
#include "aridef.hpp"
#include <memory>
#include "io/IoEnums.hpp"
#include "io/PlatformWindow.hpp"
#include "Program.hpp"
#include "PluginManager.hpp"
#include "gfx/TextureManager.hpp"

namespace bx
```

(continues on next page)

(continued from previous page)

```
{
    class Thread;
    class Mutex;
}
namespace spdlog
{
    class logger;
}
namespace ftl
{
    class TaskScheduler;
}

namespace ari
{
    struct Event;

    struct InitParams
    {
        InitParams(): Height(600), Width(800), FullScreen(false)
        {}

        uint32_t Height,
                 Width;

        bool FullScreen;
        IProgram* Program;

    }; // InitParams

    class ARI_API Engine
    {
        friend class PlatformWindow;
        friend class GuiSystem;
    public:

        Engine();

        ~Engine();

        static Engine& GetSingleton();

        bool Init(InitParams* params);

        bool Run();

        void LockUpdateThread();

        void UnlockUpdateThread();

        Event* Poll();

        void Release(const Event * _event);

        uint32_t GetCurrentFrameNumber() const { return m_frame_number; }

        std::shared_ptr<spdlog::logger> GetLogger() const { return  Logger; }
```

(continues on next page)

(continued from previous page)

```
        InitParams* GetParams() const { return m_params; }

        void SetParams(InitParams* _params) { m_params = _params; }

        PlatformWindow* GetMainWindow() const { return m_pWindow; }

        PlatformWindow* NewWindow(PlatformWindow::Type _type);

        uint16_t GetNewViewId();

        uint32_t GetMsaaFlags() const;

        float GetElapsedTime() const { return m_fElapsedTime; }

        float GetDeltaTime() const { return m_fDeltaTime; }

        PluginManager plugin_manager;
        TextureManager texture_manager;

    protected:

        static int InitBgfxInThread(bx::Thread* _thread, void* _userData);

        InitParams                    *   m_params;
        std::shared_ptr<spdlog::logger> Logger;
        PlatformWindow                *   m_pWindow;
        uint32_t                          m_debug, m_reset, m_frame_number;
        uint16_t                          m_viewId = 0;
        int64_t                           m_time_offset;
        bx::Thread                    *   m_pGfxThread;
        bx::Mutex                     *   m_pMutex = nullptr;
        int                               m_iLockStatus = 0;
        ftl::TaskScheduler            *   m_pTaskMgr;
        MouseState                        m_MouseState;
        bool                              m_bRun;
        bool                              m_bNeedReset;
        float                             m_fElapsedTime = 0.0f,
                                          m_fDeltaTime = 0.0f;

    }; // Engine

    extern ARI_API Engine* g_pEngine;

}
```

### Includes

- PluginManager.hpp (*File PluginManager.hpp*)
- Program.hpp (*File Program.hpp*)
- aridef.hpp (*File aridef.hpp*)
- gfx/TextureManager.hpp (*File TextureManager.hpp*)
- io/IoEnums.hpp (*File IoEnums.hpp*)

- io/PlatformWindow.hpp (*File PlatformWindow.hpp*)
- memory

### Namespaces

- *Namespace ari*
- *Namespace bx*
- *Namespace ftl*
- *Namespace spdlog*

### Classes

- *Struct InitParams*
- *Class Engine*

### Variables

- *Variable ari::g_pEngine*

### File Entity.hpp

*Parent directory* (include/ari/en)

**Contents**



### Definition (include/ari/en/Entity.hpp)

### Program Listing for File Entity.hpp

*Return to documentation for file* (include/ari/en/Entity.hpp)

```
#pragma once
#include "Node.hpp"

namespace ari
{
    class ARI_API Entity: public Node
```

(continues on next page)

(continued from previous page)

```
    {
    public:

        Entity();

        ~Entity();

    }; // Entity

} // ari
```

### Includes

- `Node.hpp` (*File Node.hpp*)

### Included By

- *File ProjectBrowser.hpp*
- *File DockWindow.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class Entity*

### File EventSubscriber.hpp

*Parent directory* (`include/ari/en`)

**Contents**

### Definition (include/ari/en/EventSubscriber.hpp)

### Program Listing for File EventSubscriber.hpp

*Return to documentation for file* (include/ari/en/EventSubscriber.hpp)

```
#pragma once
#include "../aridef.hpp"
#include <memory>

namespace ari
{
    class World;
    class Entity;
    class Component;
    class FrameData;

    namespace Internal
    {
        class ARI_API BaseEventSubscriber
        {
        public:

            virtual ~BaseEventSubscriber() = default;
        };

    } // Internal

    template<typename T>
    class ARI_API EventSubscriber: public Internal::BaseEventSubscriber
    {
    public:

        virtual ~EventSubscriber() = default;

        virtual void Receive(World* world, const T& event) = 0;

    }; // EventSubscriber


    namespace events
    {
        // Called when a new entity is created.
        struct OnEntityCreated
        {
            ARI_DECLARE_TYPE;

            Entity* entity;
        };

        // Called when an entity is about to be destroyed.
        struct OnEntityDestroyed
        {
            ARI_DECLARE_TYPE;

            Entity* entity;
        };
```

(continues on next page)

(continued from previous page)

```
        // Called when a component is assigned (not necessarily created).
        template <class T>
        struct OnComponentAssigned
        {
            ARI_DECLARE_TYPE;

            Entity* entity;
            T* component;
        };

        // Called when a component is removed
        template <class T>
        struct OnComponentRemoved
        {
            ARI_DECLARE_TYPE;

            Entity* entity;
            T* component;
        };

        struct OnFrameData
        {
            ARI_DECLARE_TYPE;

            FrameData* frame_data;

        };

    } // events

} // ari
```

### Includes

- ../aridef.hpp
- memory

### Included By

- *File RenderSystem.hpp*
- *File SceneSystem.hpp*
- *File GuiSystem.hpp*
- *File World.hpp*
- *File IoEvents.hpp*

### Namespaces

- *Namespace ari*

- *Namespace ari::events*
- *Namespace ari::Internal*

### Classes

- *Template Struct OnComponentAssigned*
- *Template Struct OnComponentRemoved*
- *Struct OnEntityCreated*
- *Struct OnEntityDestroyed*
- *Struct OnFrameData*
- *Template Class EventSubscriber*
- *Class BaseEventSubscriber*

### File FrameData.hpp

*Parent directory* (`include/ari/gfx`)

**Contents**



### Definition (`include/ari/gfx/FrameData.hpp`)

### Program Listing for File FrameData.hpp

*Return to documentation for file* (`include/ari/gfx/FrameData.hpp`)

```
#pragma once
#include "../aridef.hpp"
#include "../math/Matrix.hpp"
#include <tinystl/vector.h>

namespace ari
{
    class Node3D;
    class Camera;

    class ARI_API FrameData
    {
    public:
        FrameData(): FrameNumber(0)
```

(continues on next page)

(continued from previous page)

```
        {}

        tinystl::vector<Node3D*> Nodes;
        tinystl::vector<Matrix> WorldMatrices;
        uint32_t FrameNumber;
        Camera* Camera;

    }; // FrameData

} // ari
```

### Includes

- ../aridef.hpp
- ../math/Matrix.hpp
- tinystl/vector.h

### Included By

- *File SceneSystem.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class FrameData*

### File Gui.hpp

*Parent directory* (include/ari/en/gui)

**Contents**

### Definition (include/ari/en/gui/Gui.hpp)

### Program Listing for File Gui.hpp

*Return to documentation for file* (include/ari/en/gui/Gui.hpp)

```
#pragma once
#include "../Component.hpp"

namespace ari
{
    class ARI_API Gui: public Component
    {
    public:

        // Constructor
        Gui() { _isFromGui = true; }

        virtual ~Gui() = default;

        virtual bool BeginRender() { return true; }

        virtual void EndRender() { }

        bool SameLine = false;

        bool Separator = false;

        bool Visible = true;

    }; // Gui

} // ari
```

### Includes

- ../Component.hpp

### Included By

- *File Button.hpp*
- *File CheckBox.hpp*
- *File Dock.hpp*
- *File DockableWindow.hpp*
- *File DockSpace.hpp*
- *File Image.hpp*
- *File Label.hpp*
- *File Popup.hpp*
- *File TextBox.hpp*

- *File Window.hpp*
- *File ProjectBrowser.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class Gui*

### File GuiSystem.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/GuiSystem.hpp`)

### Program Listing for File GuiSystem.hpp

*Return to documentation for file* (`include/ari/en/gui/GuiSystem.hpp`)

```
#pragma once
#include "../System.hpp"
#include "../EventSubscriber.hpp"

namespace ari
{
    class Node;
    class Dock;

    class ARI_API GuiSystem: public System,
        public EventSubscriber<events::OnComponentAssigned<Dock>>
    {
        friend class AriImwWindow;

    public:

        GuiSystem();
        virtual ~GuiSystem();
```

(continues on next page)

(continued from previous page)

```
        void Update(World* p_world, UpdateState state) override;
        void Configure(World* p_world) override;
        void Unconfigure(World* p_world) override;
        Type GetSystemType() override;
        bool NeedUpdateOnState(UpdateState state) override;

        void Receive(World* world, const events::OnComponentAssigned<Dock>& event)␣
↪override;

    protected:
        bool    m_bIsDockCreated;

        void RenderGui(Node* node);
    };
}
```

### Includes

- ../EventSubscriber.hpp
- ../System.hpp

### Included By

- *File Editor.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class GuiSystem*

### File Image.hpp

*Parent directory* (include/ari/en/gui)

**Contents**

### Definition (include/ari/en/gui/Image.hpp)

### Program Listing for File Image.hpp

*Return to documentation for file* (include/ari/en/gui/Image.hpp)

```
#pragma once
#include "Gui.hpp"
#include "../../gfx/Texture.hpp"
#include "dear-imgui/imgui.h"
#include "../../Delegate.hpp"

namespace ari
{
    class ARI_API Image: public Gui
    {
    public:
        bool BeginRender() override;

        std::shared_ptr<Texture>    ImageTexture;
        ImVec2                      Size;
        DelegateNoParam<void>       OnHovered;

    }; // Image

} // ari
```

### Includes

- ../../Delegate.hpp
- ../../gfx/Texture.hpp
- Gui.hpp (*File Gui.hpp*)
- dear-imgui/imgui.h

### Namespaces

- *Namespace ari*

### Classes

- *Class Image*

### File Input.hpp

*Parent directory* (include/ari/io)

**Contents**

### Definition (include/ari/io/Input.hpp)

### Program Listing for File Input.hpp

*Return to documentation for file* (include/ari/io/Input.hpp)

```
#pragma once
#include "IoEnums.hpp"
#include "../aridef.hpp"

namespace ari
{
    typedef void(*InputBindingFn)(const void* _userData);

    struct ARI_API InputBinding
    {
        void set(Key::Enum _key, uint8_t _modifiers, uint8_t _flags, InputBindingFn _
↪fn, const void* _userData = NULL)
        {
            m_key = _key;
            m_modifiers = _modifiers;
            m_flags = _flags;
            m_fn = _fn;
            m_userData = _userData;
        }

        void end()
        {
            m_key = Key::None;
            m_modifiers = Modifier::None;
            m_flags = 0;
            m_fn = NULL;
            m_userData = NULL;
        }

        Key::Enum m_key;
        uint8_t m_modifiers;
        uint8_t m_flags;
        InputBindingFn m_fn;
        const void* m_userData;
    };

#define INPUT_BINDING_END { Key::None, Modifier::None, 0, NULL, NULL }
```

(continues on next page)

(continued from previous page)

```
    void inputInit();

    void inputShutdown();

    void inputAddBindings(const char* _name, const InputBinding* _bindings);

    void inputRemoveBindings(const char* _name);

    void inputProcess();

    void inputSetKeyState(Key::Enum  _key, uint8_t _modifiers, bool _down);

    bool inputGetKeyState(Key::Enum _key, uint8_t* _modifiers = NULL);

    uint8_t inputGetModifiersState();

    void inputChar(uint8_t _len, const uint8_t _char[4]);

    const uint8_t* inputGetChar();

    void inputCharFlush();

    void inputSetMouseResolution(uint16_t _width, uint16_t _height);

    void inputSetMousePos(int32_t _mx, int32_t _my, int32_t _mz);

    void inputSetMouseButtonState(MouseButton::Enum _button, uint8_t _state);

    void inputSetMouseLock(bool _lock);

    void inputGetMouse(float _mouse[3]);

    bool inputIsMouseLocked();

    void inputSetGamepadAxis(GamepadHandle _handle, GamepadAxis::Enum _axis, int32_t _
↪value);

    int32_t inputGetGamepadAxis(GamepadHandle _handle, GamepadAxis::Enum _axis);

}
```

### Includes

- `../aridef.hpp`
- `IoEnums.hpp` (*File IoEnums.hpp*)

### Namespaces

- *Namespace ari*

### Classes

- *Struct InputBinding*

### Functions

- *Function ari::inputAddBindings*
- *Function ari::inputChar*
- *Function ari::inputCharFlush*
- *Function ari::inputGetChar*
- *Function ari::inputGetGamepadAxis*
- *Function ari::inputGetKeyState*
- *Function ari::inputGetModifiersState*
- *Function ari::inputGetMouse*
- *Function ari::inputInit*
- *Function ari::inputIsMouseLocked*
- *Function ari::inputProcess*
- *Function ari::inputRemoveBindings*
- *Function ari::inputSetGamepadAxis*
- *Function ari::inputSetKeyState*
- *Function ari::inputSetMouseButtonState*
- *Function ari::inputSetMouseLock*
- *Function ari::inputSetMousePos*
- *Function ari::inputSetMouseResolution*
- *Function ari::inputShutdown*

### Defines

- *Define INPUT_BINDING_END*

### Typedefs

- *Typedef ari::InputBindingFn*

### File IoEnums.hpp

*Parent directory* (`include/ari/io`)

**Contents**



### Definition (include/ari/io/IoEnums.hpp)

### Program Listing for File IoEnums.hpp

*Return to documentation for file* (include/ari/io/IoEnums.hpp)

```
#pragma once
#include "bx/bx.h"
#include "bx/filepath.h"

namespace ari
{
    struct WindowHandle  { uint16_t idx; };
    inline bool isValid(WindowHandle _handle)  { return UINT16_MAX != _handle.idx; }

    struct GamepadHandle { uint16_t idx; };
    inline bool isValid(GamepadHandle _handle) { return UINT16_MAX != _handle.idx; }

    struct MouseButton
    {
        enum Enum
        {
            None,
            Left,
            Middle,
            Right,

            Count
        };
    };

    struct GamepadAxis
    {
        enum Enum
        {
            LeftX,
            LeftY,
            LeftZ,
            RightX,
            RightY,
            RightZ,

            Count
```

(continues on next page)

(continued from previous page)

```
        };
    };

    struct Modifier
    {
        enum Enum
        {
            None       = 0,
            LeftAlt    = 0x01,
            RightAlt   = 0x02,
            LeftCtrl   = 0x04,
            RightCtrl  = 0x08,
            LeftShift  = 0x10,
            RightShift = 0x20,
            LeftMeta   = 0x40,
            RightMeta  = 0x80,
        };
    };

    struct Key
    {
        enum Enum
        {
            None = 0,
            Esc,
            Return,
            Tab,
            Space,
            Backspace,
            Up,
            Down,
            Left,
            Right,
            Insert,
            Delete,
            Home,
            End,
            PageUp,
            PageDown,
            Print,
            Plus,
            Minus,
            LeftBracket,
            RightBracket,
            Semicolon,
            Quote,
            Comma,
            Period,
            Slash,
            Backslash,
            Tilde,
            F1,
            F2,
            F3,
            F4,
            F5,
            F6,
```

(continues on next page)

(continued from previous page)

```
            F7,
            F8,
            F9,
            F10,
            F11,
            F12,
            NumPad0,
            NumPad1,
            NumPad2,
            NumPad3,
            NumPad4,
            NumPad5,
            NumPad6,
            NumPad7,
            NumPad8,
            NumPad9,
            Key0,
            Key1,
            Key2,
            Key3,
            Key4,
            Key5,
            Key6,
            Key7,
            Key8,
            Key9,
            KeyA,
            KeyB,
            KeyC,
            KeyD,
            KeyE,
            KeyF,
            KeyG,
            KeyH,
            KeyI,
            KeyJ,
            KeyK,
            KeyL,
            KeyM,
            KeyN,
            KeyO,
            KeyP,
            KeyQ,
            KeyR,
            KeyS,
            KeyT,
            KeyU,
            KeyV,
            KeyW,
            KeyX,
            KeyY,
            KeyZ,

            GamepadA,
            GamepadB,
            GamepadX,
            GamepadY,
```

(continues on next page)

(continued from previous page)

```
            GamepadThumbL,
            GamepadThumbR,
            GamepadShoulderL,
            GamepadShoulderR,
            GamepadUp,
            GamepadDown,
            GamepadLeft,
            GamepadRight,
            GamepadBack,
            GamepadStart,
            GamepadGuide,

            Count
        };
    };

    struct Suspend
    {
        enum Enum
        {
            WillSuspend,
            DidSuspend,
            WillResume,
            DidResume,

            Count
        };
    };

    const char* getName(Key::Enum _key);

    struct MouseState
    {
        MouseState()
            : m_mx(0)
            , m_my(0)
            , m_mz(0)
        {
            for (unsigned char & m_button : m_buttons)
            {
                m_button = MouseButton::None;
            }
        }

        int32_t m_mx;
        int32_t m_my;
        int32_t m_mz;
        uint8_t m_buttons[MouseButton::Count];
    };

    struct GamepadState
    {
        GamepadState()
        {
            bx::memSet(m_axis, 0, sizeof(m_axis) );
        }
```

(continues on next page)

(continued from previous page)

```
        int32_t m_axis[GamepadAxis::Count];
    };

    struct WindowState
    {
        WindowState()
            : m_width(0)
            , m_height(0)
            , m_nwh(NULL)
        {
            m_handle.idx = UINT16_MAX;
        }

        WindowHandle m_handle;
        uint32_t     m_width;
        uint32_t     m_height;
        MouseState   m_mouse;
        void*        m_nwh;
        bx::FilePath m_dropFile;
    };


} // ari
```

### Includes

- bx/bx.h
- bx/filepath.h

### Included By

- *File Engine.hpp*
- *File IoEvents.hpp*
- *File Input.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Struct GamepadAxis*
- *Struct GamepadHandle*
- *Struct GamepadState*
- *Struct Key*
- *Struct Modifier*

- *Struct MouseButton*
- *Struct MouseState*
- *Struct Suspend*
- *Struct WindowHandle*
- *Struct WindowState*

### Functions

- *Function ari::getName*
- *Function ari::isValid(WindowHandle)*
- *Function ari::isValid(GamepadHandle)*

### File IoEvents.hpp

*Parent directory* (`include/ari/io`)

**Contents**



### Definition (`include/ari/io/IoEvents.hpp`)

### Program Listing for File IoEvents.hpp

*Return to documentation for file* (`include/ari/io/IoEvents.hpp`)

```
#pragma once
#include <cstdint>
#include "IoEnums.hpp"
#include "bx/filepath.h"
#include "bx/spscqueue.h"
#include "../en/EventSubscriber.hpp"

#define ENTRY_IMPLEMENT_EVENT(_class, _type) \
            _class() : Event(_type) {}
```

(continues on next page)

(continued from previous page)

```
extern bx::AllocatorI* g_allocator;

namespace ari
{
    struct Event
    {
        enum Enum
        {
            Axis,
            Char,
            Exit,
            Gamepad,
            Key,
            Mouse,
            Size,
            Window,
            Suspend,
            DropFile,
        };

        Event(Enum _type)
            : m_type(_type)
        {
        }


        Event::Enum m_type;
    };

    struct AxisEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(AxisEvent, Event::Axis);

        GamepadAxis::Enum m_axis;
        int32_t m_value;
        GamepadHandle m_gamepad;
    };

    struct CharEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(CharEvent, Event::Char);

        uint8_t m_len;
        uint8_t m_char[4];
    };

    struct GamepadEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(GamepadEvent, Event::Gamepad);

        GamepadHandle m_gamepad;
        bool m_connected;
    };

    struct KeyEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(KeyEvent, Event::Key);
```

(continues on next page)

(continued from previous page)

```
        Key::Enum m_key;
        uint8_t m_modifiers;
        bool m_down;
    };

    struct MouseEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(MouseEvent, Event::Mouse);

        int32_t m_mx;
        int32_t m_my;
        int32_t m_mz;
        MouseButton::Enum m_button;
        bool m_down;
        bool m_move;
    };

    struct SizeEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(SizeEvent, Event::Size);

        uint32_t m_width;
        uint32_t m_height;
    };

    struct WindowEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(WindowEvent, Event::Window);

        void* m_nwh;
    };

    struct SuspendEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(SuspendEvent, Event::Suspend);

        Suspend::Enum m_state;
    };

    struct DropFileEvent : public Event
    {
        ENTRY_IMPLEMENT_EVENT(DropFileEvent, Event::DropFile);

        bx::FilePath m_filePath;
    };

    const Event* poll();
    const Event* poll(WindowHandle );
    void release(const Event* _event);

    class EventQueue
    {
    public:
        EventQueue()
            : m_queue(g_allocator)
        {
```

(continues on next page)

(continued from previous page)

```
        }

        ~EventQueue()
        {
            for (const Event* ev = poll(); NULL != ev; ev = poll() )
            {
                release(ev);
            }
        }

        void postAxisEvent(GamepadHandle _gamepad, GamepadAxis::Enum _axis, int32_t _
↪value)
        {
            AxisEvent* ev = BX_NEW(g_allocator, AxisEvent)();
            ev->m_gamepad = _gamepad;
            ev->m_axis    = _axis;
            ev->m_value   = _value;
            m_queue.push(ev);
        }

        void postCharEvent(uint8_t _len, const uint8_t _char[4])
        {
            CharEvent* ev = BX_NEW(g_allocator, CharEvent)();
            ev->m_len = _len;
            bx::memCopy(ev->m_char, _char, 4);
            m_queue.push(ev);
        }

        void postExitEvent()
        {
            Event* ev = BX_NEW(g_allocator, Event)(Event::Exit);
            m_queue.push(ev);
        }

        void postGamepadEvent(GamepadHandle _gamepad, bool _connected)
        {
            GamepadEvent* ev = BX_NEW(g_allocator, GamepadEvent)();
            ev->m_gamepad   = _gamepad;
            ev->m_connected = _connected;
            m_queue.push(ev);
        }

        void postKeyEvent(Key::Enum _key, uint8_t _modifiers, bool _down)
        {
            KeyEvent* ev = BX_NEW(g_allocator, KeyEvent)();
            ev->m_key       = _key;
            ev->m_modifiers = _modifiers;
            ev->m_down      = _down;
            m_queue.push(ev);
        }

        void postMouseEvent(int32_t _mx, int32_t _my, int32_t _mz)
        {
            MouseEvent* ev = BX_NEW(g_allocator, MouseEvent)();
            ev->m_mx     = _mx;
            ev->m_my     = _my;
            ev->m_mz     = _mz;
```

(continues on next page)

(continued from previous page)

```
            ev->m_button = MouseButton::None;
            ev->m_down   = false;
            ev->m_move   = true;
            m_queue.push(ev);
        }

        void postMouseEvent(int32_t _mx, int32_t _my, int32_t _mz, MouseButton::Enum _
↪button, bool _down)
        {
            MouseEvent* ev = BX_NEW(g_allocator, MouseEvent)();
            ev->m_mx     = _mx;
            ev->m_my     = _my;
            ev->m_mz     = _mz;
            ev->m_button = _button;
            ev->m_down   = _down;
            ev->m_move   = false;
            m_queue.push(ev);
        }

        void postSizeEvent(uint32_t _width, uint32_t _height)
        {
            SizeEvent* ev = BX_NEW(g_allocator, SizeEvent)();
            ev->m_width  = _width;
            ev->m_height = _height;
            m_queue.push(ev);
        }

        void postWindowEvent(void* _nwh = NULL)
        {
            WindowEvent* ev = BX_NEW(g_allocator, WindowEvent)();
            ev->m_nwh = _nwh;
            m_queue.push(ev);
        }

        void postSuspendEvent(Suspend::Enum _state)
        {
            SuspendEvent* ev = BX_NEW(g_allocator, SuspendEvent)();
            ev->m_state = _state;
            m_queue.push(ev);
        }

        void postDropFileEvent(const bx::FilePath& _filePath)
        {
            DropFileEvent* ev = BX_NEW(g_allocator, DropFileEvent)();
            ev->m_filePath = _filePath;
            m_queue.push(ev);
        }

        const Event* poll()
        {
            return m_queue.pop();
        }

        void release(const Event* _event) const
        {
            BX_DELETE(g_allocator, const_cast<Event*>(_event) );
        }
```

(continues on next page)

(continued from previous page)

```
    private:
        bx::SpScUnboundedQueueT<Event> m_queue;
    };

} // ari
```

### Includes

- `../en/EventSubscriber.hpp`
- `IoEnums.hpp` (*File IoEnums.hpp*)
- `bx/filepath.h`
- `bx/spscqueue.h`
- `cstdint`

### Included By

- *File PlatformWindow.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Struct AxisEvent*
- *Struct CharEvent*
- *Struct DropFileEvent*
- *Struct Event*
- *Struct GamepadEvent*
- *Struct KeyEvent*
- *Struct MouseEvent*
- *Struct SizeEvent*
- *Struct SuspendEvent*
- *Struct WindowEvent*
- *Class EventQueue*

### Functions

- *Function ari::poll(WindowHandle)*
- *Function ari::poll()*
- *Function ari::release*

### Defines

- *Define ENTRY_IMPLEMENT_EVENT*

### Variables

- *Variable g_allocator*

### File JsonCast.h

*Parent directory* (`include/ari`)

**Contents**



### Definition (`include/ari/JsonCast.h`)

### Program Listing for File JsonCast.h

*Return to documentation for file* (`include/ari/JsonCast.h`)

```
#pragma once

#include <string>
#include <vector>
#include <unordered_map>

#include <json.hpp>

#include <Meta.h>
#include "StringCast.h"

using json = nlohmann::json;
```

(continues on next page)

(continued from previous page)

```
template <typename T>
void to_json(json& j, const T& obj);

template <typename T>
void from_json(const json& j, T& obj);

namespace meta
{


template <typename Class,
    typename = std::enable_if_t <meta::isRegistered<Class>()>>
json serialize(const Class& obj);

template <typename Class,
    typename = std::enable_if_t <!meta::isRegistered<Class>()>,
    typename = void>
json serialize(const Class& obj);

template <typename Class>
json serialize_basic(const Class& obj);

// specialization for std::vector
template <typename T>
json serialize_basic(const std::vector<T>& obj);

// specialization for std::unodered_map
template <typename K, typename V>
json serialize_basic(const std::unordered_map<K, V>& obj);


//
//template<typename Class>
//Class deserialize(const json& obj);

template <typename Class,
    typename = std::enable_if_t<meta::isRegistered<Class>()>>
void deserialize(Class& obj, const json& object);

template <typename Class,
    typename = std::enable_if_t<!meta::isRegistered<Class>()>,
    typename = void>
void deserialize(Class& obj, const json& object);

// specialization for std::vector
template <typename T>
void deserialize(std::vector<T>& obj, const json& object);

// specialization for std::unodered_map
template <typename K, typename V>
void deserialize(std::unordered_map<K, V>& obj, const json& object);

}

#include "JsonCast.inl"
```

### Includes

- `JsonCast.inl` (*File JsonCast.inl*)
- `Meta.h`
- `StringCast.h` (*File StringCast.h*)
- `json.hpp`
- `string`
- `unordered_map`
- `vector`

### Included By

- *File JsonCast.inl*

### Namespaces

- *Namespace meta*

### Functions

- *Template Function from_json(const json&, T&)*
- *Template Function meta::deserialize(Class&, const json&)*
- *Template Function meta::deserialize(std::unordered_map<K, V>&, const json&)*
- *Template Function meta::deserialize(std::vector<T>&, const json&)*
- *Template Function meta::deserialize(Class&, const json&)*
- *Template Function meta::serialize(const Class&)*
- *Template Function meta::serialize(const Class&)*
- *Template Function meta::serialize_basic(const std::vector<T>&)*
- *Template Function meta::serialize_basic(const std::unordered_map<K, V>&)*
- *Template Function meta::serialize_basic(const Class&)*
- *Template Function to_json(json&, const T&)*

### Typedefs

- *Typedef json*

### File JsonCast.inl

*Parent directory* (include/ari)

**Contents**



### Definition (include/ari/JsonCast.inl)

### Program Listing for File JsonCast.inl

*Return to documentation for file* (include/ari/JsonCast.inl)

```
#include "JsonCast.h"

template <typename T>
void to_json(json& j, const T& obj)
{
    j = meta::serialize(obj);
}

template <typename T>
void from_json(const json& j, T& obj)
{
    meta::deserialize(obj, j);
}

namespace meta
{


template <typename Class,
    typename>
json serialize(const Class& obj)
{
    json value;
    meta::doForAllMembers<Class>(
        [&obj, &value](auto& member)
        {
            auto& valueName = value[member.getName()];
            if (member.canGetConstRef()) {
                valueName = member.get(obj);
            } else if (member.hasGetter()) {
                valueName = member.getCopy(obj); // passing copy as const ref, it's␣
↪okay
            }
```

(continues on next page)

(continued from previous page)

```
        }
    );
    return value;
}

template <typename Class,
    typename, typename>
json serialize(const Class& obj)
{
    return serialize_basic(obj);
}

template <typename Class>
json serialize_basic(const Class& obj)
{
    return json(obj);
}

// specialization for std::vector
template <typename T>
json serialize_basic(const std::vector<T>& obj)
{
    json value;
    int i = 0;
    for (auto& elem : obj) {
        value[i] = elem;
        ++i;
    }
    return value;
}

// specialization for std::unordered_map
template <typename K, typename V>
json serialize_basic(const std::unordered_map<K, V>& obj)
{
    json value;
    for (auto& pair : obj) {
        value.emplace(castToString(pair.first), pair.second);
    }
    return value;
}


template <typename Class>
Class deserialize(const json& obj)
{
    Class c;
    deserialize(c, obj);
    return c;
}

template <typename Class,
    typename>
void deserialize(Class& obj, const json& object)
{
    if (object.is_object()) {
        meta::doForAllMembers<Class>(
```

(continues on next page)

(continued from previous page)

```
            [&obj, &object](auto& member)
            {
                auto& objName = object[member.getName()];
                if (!objName.is_null()) {
                    using MemberT = meta::get_member_type<decltype(member)>;
                    if (member.hasSetter()) {
                        member.set(obj, objName.template get<MemberT>());
                    } else if (member.canGetRef()) {
                        member.getRef(obj) = objName.template get<MemberT>();
                    } else {
                        throw std::runtime_error("Error: can't deserialize member
↪because it's read only");
                    }
                }
            }
        );
    } else {
        throw std::runtime_error("Error: can't deserialize from Json::json to Class.
↪");
    }
}

template <typename Class,
    typename, typename>
void deserialize(Class& obj, const json& object)
{
    obj = object.get<Class>();
}

// specialization for std::vector
template <typename T>
void deserialize(std::vector<T>& obj, const json& object)
{
    obj.reserve(object.size()); // vector.resize() works only for default
↪constructible types
    for (auto& elem : object) {
        obj.push_back(elem); // push rvalue
    }
}

// specialization for std::unodered_map
template <typename K, typename V>
void deserialize(std::unordered_map<K, V>& obj, const json& object)
{
    for (auto it = object.begin(); it != object.end(); ++it) {
        obj.emplace(fromString<K>(it.key()), it.value());
    }
}

}
```

### Includes

- `JsonCast.h` (*File JsonCast.h*)

### Included By

- *File JsonCast.h*

### Namespaces

- *Namespace meta*

### Functions

- *Template Function from_json(const json&, T&)*
- *Template Function meta::deserialize(const json&)*
- *Template Function to_json(json&, const T&)*

### File Label.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/Label.hpp`)

### Program Listing for File Label.hpp

*Return to documentation for file* (`include/ari/en/gui/Label.hpp`)

```
#pragma once
#include "Gui.hpp"

namespace ari
{
    class ARI_API Label : public Gui
    {
    public:

        Label();

        ~Label() = default;

        bool BeginRender() override;
```

(continues on next page)

(continued from previous page)

```
        const char  *   Text;
    };

}
```

### Includes

- `Gui.hpp` (*File Gui.hpp*)

### Included By

- *File ProjectBrowser.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class Label*

### File Matrix.hpp

*Parent directory* (`include/ari/math`)

**Contents**



### Definition (`include/ari/math/Matrix.hpp`)

### Program Listing for File Matrix.hpp

*Return to documentation for file* (`include/ari/math/Matrix.hpp`)

```
#pragma once

#include "../aridef.hpp"
#include "bx/macros.h"
#include "bx/float4x4_t.h"
#include "Vector.hpp"

namespace ari
{
    BX_ALIGN_DECL_16(struct) Matrix
    {
        union
        {
            float v[16];
            struct
            {
                float _11, _12, _13, _14;
                float _21, _22, _23, _24;
                float _31, _32, _33, _34;
                float _41, _42, _43, _44;
            };
            bx::float4x4_t f;
        };

        Matrix() :
            _11(1.0f), _12(0.0f), _13(0.0f), _14(0.0f),
            _21(0.0f), _22(1.0f), _23(0.0f), _24(0.0f),
            _31(0.0f), _32(0.0f), _33(1.0f), _34(0.0f),
            _41(0.0f), _42(0.0f), _43(0.0f), _44(1.0f)
        { }

        void Identity();

        Matrix operator *(const Matrix &m) const;

        void operator *=(const Matrix &m);

        void SetPositionRotation(const Vector3& position,
            const Vector3& rotation);

        void SetTransform(const Vector3& position,
            const Vector3& rotation,
            const Vector3& scale);

    }; // Matrix

} // ari
```

### Includes

- ../aridef.hpp
- Vector.hpp (*File Vector.hpp*)
- bx/float4x4_t.h
- bx/macros.h

### Included By

- *File Node3D.hpp*
- *File Camera.hpp*
- *File FrameData.hpp*

### Namespaces

- *Namespace ari*

### Functions

- *Function ari::BX_ALIGN_DECL_16*

### File Node.hpp

*Parent directory* (`include/ari/en`)

**Contents**



### Definition (`include/ari/en/Node.hpp`)

### Program Listing for File Node.hpp

*Return to documentation for file* (`include/ari/en/Node.hpp`)

```
#pragma once

#include "../aridef.hpp"
#include "tinystl/vector.h"
#include <memory>
#include <cassert>
#include <unordered_map>

namespace ari
{
    class World;
    class Entity;

    class ARI_API Node
```

(continues on next page)

(continued from previous page)

```
    {
        friend class World;

    public:

        enum class Type
        {
            Entity = 0,
            Component,

            Unknown

        };

        Node();

        virtual ~Node();

        template <class T>
        T* AddChild(T* child)
        {
            m_vChilds.push_back(child);
            child->m_pWorld = m_pWorld;
            child->SetParent(this);

            // Add the child to map
            auto index = getTypeIndex<T>();
            auto found = childs.find(index);
            if (found == childs.end())
            {
                tinystl::vector<Node*> subList;
                subList.push_back(child);

                childs.insert({ index, subList });
            }
            else
            {
                found->second.push_back(child);
            }

            if (child->m_eNodeType == Type::Component)
            {
                assert(m_pWorld);
                m_pWorld->emit<events::OnComponentAssigned<T>>({ child->
↪GetParentEntity(), child });
            }

            return child;

        } // AddChild

        // Returns the first attached Node.
        template <class T>
        T* GetChild()
        {
            auto found = childs.find(getTypeIndex<T>());
            if (found != childs.end())
```

(continues on next page)

(continued from previous page)

```
            {
                return reinterpret_cast<T*>(found->second[0]);
            }
            return nullptr;
        }

        template <class T>
        tinystl::vector<Node*> GetChildren()
        {
            auto found = childs.find(getTypeIndex<T>());
            if (found != childs.end())
            {
                return found->second;
            }
            return tinystl::vector<Node*>();
        }

        template <class T>
        void RemoveChild(T* child)
        {
            for (tinystl::vector<Node*>::iterator it = m_vChilds.begin();
                it != m_vChilds.end(); ++it)
            {
                if ((*it) == child)
                {
                    child->m_pParent = nullptr;
                    m_vChilds.erase(it);

                    // Remove it from map
                    auto index = getTypeIndex<T>();
                    auto found = childs.find(index);
                    if (found != childs.end())
                    {
                        found->second.erase(std::remove(found->second.begin(), found->
↪second.end(), child), found->second.end());
                        if (found->second.size() == 0)
                        {
                            childs.erase(found);
                        }
                    }

                    if (child->m_eNodeType == Type::Component)
                    {
                        assert(m_pWorld);
                        m_pWorld->emit<events::OnComponentRemoved<T>>({ child->
↪GetParentEntity(), child });
                    }
                    return;
                }
            }

        } // RemoveChild

        void RemoveChildren(bool _delete = false);

        virtual Node* GetParent() { return m_pParent; }
```

(continues on next page)

(continued from previous page)

```
        virtual void SetParent(Node* parent);

        Node::Type GetType() const { return m_eNodeType; }

        Entity* GetParentEntity() const;

        const tinystl::vector<Node*>& GetChildren() const { return m_vChilds; }

        World* GetWorld() const { return m_pWorld; }

        void Destroy(bool addToDestroyQueue = true);

        uint32_t IsInDestroyQueue() const { return m_iIsInDestroyQueue; }

    protected:

        Node* m_pParent;
        tinystl::vector<Node*> m_vChilds;
        Node::Type m_eNodeType;
        World* m_pWorld;
        uint32_t m_iIsInDestroyQueue = 0;
        std::unordered_map<TypeIndex,
            tinystl::vector<Node*>> childs;

    }; // Node

} // ari
```

### Includes

- ../aridef.hpp
- cassert
- memory
- tinystl/vector.h
- unordered_map

### Included By

- *File Component.hpp*
- *File Entity.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class Node*

### File Node3D.hpp

*Parent directory* (include/ari/en/3d)

**Contents**



### Definition (include/ari/en/3d/Node3D.hpp)

### Program Listing for File Node3D.hpp

*Return to documentation for file* (include/ari/en/3d/Node3D.hpp)

```
#pragma once

#include "../Component.hpp"
#include "../../math/Vector.hpp"
#include "../../math/Matrix.hpp"

namespace bgfx
{
    struct Encoder;
}
namespace ari
{
    class ARI_API Node3D: public Component
    {
    public:

        Node3D() : Scale(1.0f, 1.0f, 1.0f), _isRenderable(false) { _isFromNode3D =␣
↪true; }

        virtual ~Node3D() = default;

        virtual void Render(const Matrix& matrix, bgfx::Encoder* encoder, uint16_t _
↪view_id) { BX_UNUSED(matrix, encoder); }

        Vector3 Position,
            Rotation,
            Scale;

        Matrix _finalMat;
        bool _isRenderable;

    }; // Node3D

} // ari
```

### Includes

- `../../math/Matrix.hpp`
- `../../math/Vector.hpp`
- `../Component.hpp`

### Included By

- *File BoxShape.hpp*
- *File Camera.hpp*

### Namespaces

- *Namespace ari*
- *Namespace bgfx*

### Classes

- *Class Node3D*

### File PIE.hpp

*Parent directory* (`include/shiva/windows`)

**Contents**



### Definition (`include/shiva/windows/PIE.hpp`)

### Program Listing for File PIE.hpp

*Return to documentation for file* (`include/shiva/windows/PIE.hpp`)

```
#pragma once

namespace shiva
{

} // shiva
```

### Namespaces

- *Namespace shiva*

### File PlatformWindow.hpp

*Parent directory* (include/ari/io)

**Contents**



### Definition (include/ari/io/PlatformWindow.hpp)

### Program Listing for File PlatformWindow.hpp

*Return to documentation for file* (include/ari/io/PlatformWindow.hpp)

```
#pragma once
#include "../aridef.hpp"
#include <cstdint>
#include "IoEvents.hpp"
#include "../Delegate.hpp"
#include "tinystl/vector.h"

namespace ari
{
    class ARI_API PlatformWindow
    {
        friend class Engine;

    public:

        enum class Type
        {
            Main,
            Child,
            Popup
        };

        PlatformWindow(Type _type): m_Type(_type) {}
        virtual ~PlatformWindow() {}

        virtual bool Init(int _posx, int _posy, int _width, int _height, uint32_t _
↪flags,
            const char* _title) = 0;
```

(continues on next page)

(continued from previous page)

```
        virtual bool Run() = 0;

        virtual void Show(bool _show) = 0;

        virtual void SetMousePos(int _x, int _y) = 0;

        virtual void SetTitle(const char* _title) = 0;

        virtual void SetFlags(uint32_t _flags, bool _addFlags = false) = 0;

        virtual void GetPos(int& _x, int& _y) = 0;
        virtual void SetPos(int _x, int _y) = 0;

        virtual void GetSize(int& _width, int& _height);
        virtual void SetSize(int _width, int _height) = 0;

        virtual void SetAlpha(unsigned char _alpha) = 0;

        virtual void SetMouseLock(bool _lock) = 0;

        virtual void ToggleFrame() = 0;

        virtual bool IsWindowMaximized() = 0;
        virtual void SetWindowMaximized(bool _maximize) = 0;
        virtual bool IsWindowMinimized() = 0;
        virtual void SetWindowMinimized(bool _minimize) = 0;

        virtual void* GetHandle() = 0;

        void AddOnKeyDelegate(DelegateTwoParam<void, Key::Enum, bool>* _pDelegate);
        void RemoveOnKeyDelegate(DelegateTwoParam<void, Key::Enum, bool>* _pDelegate);

        void AddOnCharDelegate(DelegateTwoParam<void, uint8_t, uint8_t*>* _pDelegate);
        void RemoveOnCharDelegate(DelegateTwoParam<void, uint8_t, uint8_t*>* _
↪pDelegate);

        void AddOnMouseButtonDelegate(DelegateTwoParam<void, MouseButton::Enum, bool>
↪* _pDelegate);
        void RemoveOnMouseButtonDelegate(DelegateTwoParam<void, MouseButton::Enum,␣
↪bool>* _pDelegate);

        void AddOnMouseMoveDelegate(DelegateTwoParam<void, int, int>* _pDelegate);
        void RemoveOnMouseMoveDelegate(DelegateTwoParam<void, int, int>* _pDelegate);

        void AddOnMouseWheelDelegate(DelegateOneParam<void, int>* _pDelegate);
        void RemoveOnMouseWheelDelegate(DelegateOneParam<void, int>* _pDelegate);

        void AddOnSizeDelegate(DelegateTwoParam<void, int, int>* _pDelegate);
        void RemoveOnSizeDelegate(DelegateTwoParam<void, int, int>* _pDelegate);

        bool ProcessEvents(uint32_t& _width, uint32_t& _height, uint32_t& _debug,␣
↪uint32_t& _reset,
            MouseState* _mouse);

    protected:
```

(continues on next page)

(continued from previous page)

```
        Type        m_Type;
        uint32_t    m_width;
        uint32_t    m_height;
        uint32_t    m_oldWidth;
        uint32_t    m_oldHeight;
        uint32_t    m_frameWidth;
        uint32_t    m_frameHeight;
        float       m_aspectRatio;
        EventQueue  m_eventQueue;
        tinystl::vector<DelegateTwoParam<void, Key::Enum, bool>*>
                    m_vOnKeys;
        tinystl::vector<DelegateTwoParam<void, uint8_t, uint8_t*>*>
                    m_vOnChar;
        tinystl::vector<DelegateTwoParam<void, MouseButton::Enum, bool>*>
                    m_vOnMouseButtons;
        tinystl::vector<DelegateTwoParam<void, int, int>*>
                    m_vOnMouseMove;
        tinystl::vector<DelegateOneParam<void, int>*>
                    m_vOnMouseWheel;
        tinystl::vector<DelegateTwoParam<void, int, int>*>
                    m_vOnSize;

    }; // Window

} // ari
```

### Includes

- ../Delegate.hpp
- ../aridef.hpp
- IoEvents.hpp (*File IoEvents.hpp*)
- cstdint
- tinystl/vector.h

### Included By

- *File Engine.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class PlatformWindow*

### File Plugin.hpp

*Parent directory* (include/ari)

**Contents**



### Definition (include/ari/Plugin.hpp)

### Program Listing for File Plugin.hpp

*Return to documentation for file* (include/ari/Plugin.hpp)

```
#pragma once
#include "Resource.hpp"

namespace ari
{
    class Plugin: public Resource
    {
    public:

        enum class Type
        {
            TextureLoader,
            MeshLoader,

            Unknown
        };

        Plugin(const uint32_t& _handel, const std::string& _fileName)
            : Resource(_handel, _fileName)
        {
        }

        virtual ~Plugin() = default;

        virtual void* Create() = 0;

    protected:

        Type    m_eType = Type::Unknown;

    }; // Plugin

} // ari
```

### Includes

- `Resource.hpp` (*File Resource.hpp*)

### Included By

- *File PluginManager.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class Plugin*

## File PluginManager.hpp

*Parent directory* (`include/ari`)

**Contents**



## Definition (`include/ari/PluginManager.hpp`)

## Program Listing for File PluginManager.hpp

*Return to documentation for file* (`include/ari/PluginManager.hpp`)

```
#pragma once
#include "ResourceManager.hpp"
#include "Plugin.hpp"
#include "aridef.hpp"

namespace ari
{
    class ARI_API PluginManager: public ResourceManager<Plugin>
    {
    protected:

        bool LoadResource(Plugin** ppOut, uint32_t handle,
```

(continues on next page)

(continued from previous page)

```
            const std::string& filename, void* extraParams) override;

    };

} // ari
```

### Includes

- `Plugin.hpp` (*File Plugin.hpp*)
- `ResourceManager.hpp` (*File ResourceManager.hpp*)
- `aridef.hpp` (*File aridef.hpp*)

### Included By

- *File Engine.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class PluginManager*

### File Popup.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/Popup.hpp`)

### Program Listing for File Popup.hpp

*Return to documentation for file* (`include/ari/en/gui/Popup.hpp`)

```
#pragma once
#include "Gui.hpp"

namespace ari
{
    class ARI_API Popup: public Gui
    {
    public:

        bool BeginRender() override;

        void EndRender() override;

        void Show();

        void Hide();

        char*   Name = nullptr;

    protected:

        bool m_bDoEndPopup = false;
        bool m_bOpenPopup = false;
        bool m_bClosePopup = false;

    }; // Popup

} // ari
```

### Includes

- `Gui.hpp` (*File Gui.hpp*)

### Namespaces

- *Namespace ari*

### Classes

- *Class Popup*

### File Program.hpp

*Parent directory* (`include/ari`)

**Contents**

- *Included By*
- *Namespaces*
- *Classes*

### Definition (`include/ari/Program.hpp`)

### Program Listing for File Program.hpp

*Return to documentation for file* (`include/ari/Program.hpp`)

```
#pragma once
#include <tinystl/string.h>

namespace ari
{
    class IProgram
    {
    public:
        IProgram(const char* programName): m_sProgramName(programName)
        { }

        virtual ~IProgram() = default;

        virtual void Init() = 0;

        virtual bool Update(uint32_t frame_number, float elasped) = 0;

        virtual int Shutdown() = 0;

        tinystl::string GetProgramName() const { return m_sProgramName; }

    protected:

        tinystl::string m_sProgramName;
    };
}
```

### Includes

- `tinystl/string.h`

### Included By

- *File Engine.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class IProgram*

### File Project.hpp

*Parent directory* (include/shiva)

**Contents**



### Definition (include/shiva/Project.hpp)

### Program Listing for File Project.hpp

*Return to documentation for file* (include/shiva/Project.hpp)

```
#pragma once
#include "bx/filepath.h"
#include <string>
#include <Meta.h>
#include <bx/error.h>
#include "DirectoryTree.hpp"

BX_ERROR_RESULT(SH_ERROR_NOT_EMPTY_DIRECTPRY, BX_MAKEFOURCC('s', 'h', 0, 0));

namespace shiva
{
    class Project
    {
        friend auto meta::registerMembers<Project>();
    public:
        Project();

        ~Project();

        static Project* New(bx::FilePath projectPath, std::string name, bx::Error*␣
↪err);

        void Save();
        static Project* Load(bx::FilePath path, bx::Error* err);

        void UpdateProjectTree();

        const DirectoryTree& GetTree() const { return m_Tree; }
```

(continues on next page)

(continued from previous page)

```
        const bx::FilePath& GetPath() const { return m_ProjectPath; }

    private:

        bx::FilePath    m_ProjectPath;
        std::string     m_ProjectName;
        DirectoryTree   m_Tree;

    }; // Project

} // shiva

namespace meta {

    template <>
    inline auto registerMembers<shiva::Project>()
    {
        return members(
            member("name", &shiva::Project::m_ProjectName)
        );
    }

} // end of namespace meta
```

### Includes

- DirectoryTree.hpp (*File DirectoryTree.hpp*)
- Meta.h
- bx/error.h
- bx/filepath.h
- string

### Namespaces

- *Namespace meta*
- *Namespace shiva*

### Classes

- *Class Project*

### Functions

- *Function BX_ERROR_RESULT*
- *Function meta::registerMembers< shiva::Project >*

### File ProjectBrowser.hpp

*Parent directory* (include/shiva/windows)

**Contents**



### Definition (include/shiva/windows/ProjectBrowser.hpp)

### Program Listing for File ProjectBrowser.hpp

*Return to documentation for file* (include/shiva/windows/ProjectBrowser.hpp)

```
#pragma once
#include "shiva/shivadef.hpp"
#include "ari/en/Entity.hpp"
#include "ari/en/gui/Gui.hpp"
#include "ari/en/gui/Button.hpp"
#include "ari/en/gui/Label.hpp"
#include "DockWindow.hpp"

namespace ari
{
    class DockableWindow;
    class World;
    class TextBox;
    class Popup;

} // ari

namespace shiva
{
    class Project;

    class ProjectGui: public ari::Gui
    {
    public:
        bool BeginRender() override;
        void EndRender() override;

    }; // ProjectGui

    class SHIVA_API ProjectBrowser : DockWindow
    {
    public:
        ProjectBrowser();
```

(continues on next page)

(continued from previous page)

```
        ~ProjectBrowser();

        void Init(ari::World* p_world) override;

        void Shutdown() override;

    protected:

        ari::TextBox        *   m_pNewProjectName;
        ari::TextBox        *   m_pNewProjectPath;
        ari::Button         *   m_pNewProjectBtn;
        ari::TextBox        *   m_pOpenProjectPath;
        ari::Button         *   m_pOpenProjectBtn;
        ari::Popup          *   m_pMessageBox;
        ari::Label          *   m_pMbLabel;
        ari::Button         *   m_pMbOkBtn;

        void OnNewProjectClick();
        void OnOpenProjectClick();

        void OnClickMbOk();

        void ProjectOpened(Project* project);

    }; // ProjectBrowser

} // shiva
```

### Includes

- DockWindow.hpp (*File DockWindow.hpp*)
- ari/en/Entity.hpp (*File Entity.hpp*)
- ari/en/gui/Button.hpp (*File Button.hpp*)
- ari/en/gui/Gui.hpp (*File Gui.hpp*)
- ari/en/gui/Label.hpp (*File Label.hpp*)
- shiva/shivadef.hpp (*File shivadef.hpp*)

### Included By

- *File Editor.hpp*

### Namespaces

- *Namespace ari*
- *Namespace shiva*

### Classes

- *Class ProjectBrowser*
- *Class ProjectGui*

### File PropertyEditor.hpp

*Parent directory* (`include/shiva/windows`)

**Contents**



### Definition (`include/shiva/windows/PropertyEditor.hpp`)

### Program Listing for File PropertyEditor.hpp

*Return to documentation for file* (`include/shiva/windows/PropertyEditor.hpp`)

```
#pragma once
#include "DockWindow.hpp"

namespace shiva
{
    class SHIVA_API PropertyEditor : public DockWindow
    {
    public:

        void Init(ari::World* p_world) override;

    };

} // shiva
```

### Includes

- `DockWindow.hpp` (*File DockWindow.hpp*)

### Namespaces

- *Namespace shiva*

### Classes

- *Class PropertyEditor*

### File Rect.hpp

*Parent directory* (include/ari/math)

**Contents**



### Definition (include/ari/math/Rect.hpp)

### Program Listing for File Rect.hpp

*Return to documentation for file* (include/ari/math/Rect.hpp)

```
#pragma once

namespace ari
{
    template <class T>
    struct Rect
    {
        Rect() : x(0), y(0), width(0), height(0) { }

        Rect(const T _x, const T _y, const T _width, const T _height):
            x(_x), y(_y), width(_width), height(_height) { }

        void Set(const T _x, const T _y, const T _width, const T _height)
        {
            x      = _x;
            y      = _y;
            width  = _width;
            height = _height;
        }

        bool operator == (const Rect<T> &v) const
        {
            return (width == v.width && height == v.height && x == v.x && y == v.y);
        }

        bool operator != (const Rect<T> &v) const
        {
            return (width != v.width || height != v.height || x != v.x || y != v.y);
```

(continues on next page)

(continued from previous page)

```
        }

        union
        {
            T p[4];
            struct
            {
                T   x,
                    y,
                    width,
                    height;
            };
        };
    };

    typedef Rect<uint16_t>  RectU16;
    typedef Rect<int>       RectI;
    typedef Rect<float>     RectF;

} // ari
```

### Included By

- *File Viewport.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Template Struct Rect*

### Typedefs

- *Typedef ari::RectF*
- *Typedef ari::RectI*
- *Typedef ari::RectU16*

### File RenderSystem.hpp

*Parent directory* (`include/ari/en/3d`)

**Contents**

- *Definition (`include/ari/en/3d/RenderSystem.hpp`)*

- *Includes*
- *Included By*
- *Namespaces*
- *Classes*

### Definition (include/ari/en/3d/RenderSystem.hpp)

### Program Listing for File RenderSystem.hpp

*Return to documentation for file* (include/ari/en/3d/RenderSystem.hpp)

```
#pragma once
#include "../System.hpp"
#include "../EventSubscriber.hpp"

namespace bgfx
{
    struct VertexDecl;
    struct ProgramHandle;
}

namespace ari
{
    class BoxShape;

    class ARI_API RenderSystem: public System,
        public EventSubscriber<events::OnComponentAssigned<BoxShape>>,
        public EventSubscriber<events::OnFrameData>
    {
    public:

        enum class VertexType
        {
            Pos,
            Color,
            Count
        };

        RenderSystem();
        ~RenderSystem();

        void Update(World* p_world, UpdateState state) override;
        void Configure(World* p_world) override;
        void Unconfigure(World* p_world) override;
        Type GetSystemType() override
        {
            return Type::RenderSystem;
        }
        bool NeedUpdateOnState(UpdateState state) override;

        void Receive(World* world, const events::OnComponentAssigned<BoxShape>&␣
↪event) override;
        void Receive(World* world, const events::OnFrameData& event) override;
```

(continues on next page)

(continued from previous page)

```
        bgfx::VertexDecl* GetVertexDecl(VertexType vertex_type) const;

        bgfx::ProgramHandle* GetProgram() const { return m_Program; }

    protected:

        bgfx::VertexDecl     *   m_pVertexDeclArray;
        bgfx::ProgramHandle  *   m_Program;
        FrameData            *   m_pFrameDataCurrent,
                             *   m_pFrameDataNext;
        uint16_t                 m_view_id = 0;
    };

} // ari
```

### Includes

- ../EventSubscriber.hpp
- ../System.hpp

### Included By

- *File Editor.hpp*

### Namespaces

- *Namespace ari*
- *Namespace bgfx*

### Classes

- *Class RenderSystem*

### File Resource.hpp

*Parent directory* (include/ari)

**Contents**

### Definition (include/ari/Resource.hpp)

### Program Listing for File Resource.hpp

*Return to documentation for file* (include/ari/Resource.hpp)

```
#pragma once
#include <string>
#include "aridef.hpp"

namespace ari
{
    class ARI_API Resource
    {
        template <class T>
        friend class ResourceManager;

    public:

        Resource(): m_iHandle(0) { }

        Resource(const uint32_t& _handel, const std::string& _fileName):
            m_iHandle(_handel), m_sFileName(_fileName) { }

        virtual ~Resource() = default;

        uint32_t GetHandel() const { return m_iHandle; }

        std::string GetFileName() const { return m_sFileName; }

    protected:

        uint32_t    m_iHandle;
        std::string m_sFileName;
    };

} // ari
```

### Includes

- aridef.hpp (*File aridef.hpp*)
- string

### Included By

- *File Texture.hpp*
- *File Plugin.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class Resource*

### File ResourceLoader.hpp

*Parent directory* (include/ari)

**Contents**



### Definition (include/ari/ResourceLoader.hpp)

### Program Listing for File ResourceLoader.hpp

*Return to documentation for file* (include/ari/ResourceLoader.hpp)

```
#pragma once
#include "aridef.hpp"
#include <vector>
#include "bx/readerwriter.h"
#include "bx/file.h"

namespace ari
{
    class Resource;

    class ARI_API ResourceLoader
    {
    public:

        ResourceLoader() : m_bSwapEndian(false)
        {}

        virtual ~ResourceLoader() = default;

        virtual bool IsALoadableFileExtension(std::string _extention);


        virtual Resource* LoadResource(bx::FileReaderI* pStream, uint32_t _handle,
            const std::string& _filename, void* _extraParams) = 0;

    protected:

        std::vector<std::string>            m_aFileExtension;
        bool                                m_bSwapEndian;
```

(continues on next page)

(continued from previous page)

```
    };

} // ari
```

### Includes

- `aridef.hpp` (*File aridef.hpp*)
- `bx/file.h`
- `bx/readerwriter.h`
- `vector`

### Namespaces

- *Namespace ari*

### Classes

- *Class ResourceLoader*

### File ResourceManager.hpp

*Parent directory* (`include/ari`)

**Contents**



### Definition (`include/ari/ResourceManager.hpp`)

### Program Listing for File ResourceManager.hpp

*Return to documentation for file* (`include/ari/ResourceManager.hpp`)

```
#pragma once
#include <string>
#include <vector>
#include <stack>
```

(continues on next page)

(continued from previous page)

```
namespace ari
{
    class ResourceLoader;

    template <class T>
    class ResourceManager
    {
    public:

        virtual ~ResourceManager() = default;

        std::shared_ptr<T> Load(const std::string& filename,    void* extraParams)
        {
            // Searching for resource.
            for (const auto& res: m_vResources)
            {
                if (res)
                {
                    if (!res->GetFileName().empty() && !filename.empty())
                    {
                        if (res->GetFileName() == filename)
                        {
                            return res;
                        }
                    }
                }
            }

            // Resource not loaded yet.
            T* pResource = nullptr;
            const uint32_t handle = GetNewHandle();

            if (!LoadResource(&pResource, handle, filename, extraParams))
                return nullptr;

            return AddResource(pResource);
        }

        void AddLoader(ResourceLoader* pLoader)
        {
            m_vLoaders.push_back(pLoader);
        }

        uint32_t GetNewHandle()
        {
            uint32_t handle;
            if (!m_sHandles.empty())
            {
                handle = m_sHandles.top();
                m_sHandles.pop();
            }
            else
            {
                handle = uint32_t(m_vResources.size());
            }
            return handle;
        }
```

(continues on next page)

(continued from previous page)

```
        std::shared_ptr<T> AddResource(T* _resource)
        {
            std::shared_ptr<T> sp(_resource);
            if (_resource->m_iHandle >= m_vResources.size())
                m_vResources.push_back(sp);
            else
                m_vResources[_resource->m_iHandle] = sp;

            return sp;
        }

    protected:

        virtual bool LoadResource(T** ppOut, uint32_t handle, const std::string&␣
↪filename,
            void* extraParams) = 0;

        std::vector<std::shared_ptr<T>> m_vResources;
        std::stack<uint32_t>            m_sHandles;
        std::vector<ResourceLoader*>    m_vLoaders;

    };
} // ari
```

### Includes

- stack
- string
- vector

### Included By

- *File PluginManager.hpp*
- *File TextureManager.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Template Class ResourceManager*

### File SceneSystem.hpp

*Parent directory* (include/ari/en/3d)

**Contents**



## Definition (include/ari/en/3d/SceneSystem.hpp)

## Program Listing for File SceneSystem.hpp

*Return to documentation for file* (include/ari/en/3d/SceneSystem.hpp)

```
#pragma once
#include "../System.hpp"
#include "../EventSubscriber.hpp"
#include "../../gfx/FrameData.hpp"

namespace ari
{
    class Camera;
    class BoxShape;
    class Node;

    class ARI_API SceneSystem: public System,
        public EventSubscriber<events::OnEntityCreated>,
        public EventSubscriber<events::OnEntityDestroyed>,
        public EventSubscriber<events::OnComponentAssigned<Camera>>,
        public EventSubscriber<events::OnComponentRemoved<Camera>>,
        public EventSubscriber<events::OnComponentAssigned<BoxShape>>,
        public EventSubscriber<events::OnComponentRemoved<BoxShape>>
    {
    public:

        SceneSystem();

        ~SceneSystem();

        void Update(World* p_world, UpdateState state) override;
        void Configure(World* p_world) override;
        void Unconfigure(World* p_world) override;
        Type GetSystemType() override
        {
            return Type::SceneSystem;
        }
        bool NeedUpdateOnState(UpdateState state) override;

        void Receive(World* world, const events::OnEntityCreated& event) override;
        void Receive(World* world, const events::OnEntityDestroyed& event) override;
        void Receive(World* world, const events::OnComponentAssigned<Camera>& event)␣
↪override;
```

(continues on next page)

(continued from previous page)

```
        void Receive(World* world, const events::OnComponentRemoved<Camera>& event)␣
↪override;
        void Receive(World* world, const events::OnComponentAssigned<BoxShape>&␣
↪event) override;
        void Receive(World* world, const events::OnComponentRemoved<BoxShape>& event)␣
↪override;

    protected:

        Camera * m_pActiveCamera;
        FrameData    *    m_FrameDatasUnused,         // This is the unused frame data␣
↪pointers
                     *    m_FrameDatasTransforms,     // This is the transform␣
↪calculated nodes
                     *    m_FrameDatasVisible;        // This is the visible nodes that␣
↪must be rendered.

        void CalcTransform(Node* node, Matrix* parentMat);

    }; // SceneSystem

} // ari
```

### Includes

- ../../gfx/FrameData.hpp
- ../EventSubscriber.hpp
- ../System.hpp

### Included By

- *File Editor.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class SceneSystem*

### File shivadef.hpp

*Parent directory* (include/shiva)

**Contents**

### Definition (include/shiva/shivadef.hpp)

### Program Listing for File shivadef.hpp

*Return to documentation for file* (include/shiva/shivadef.hpp)

```
#pragma once

#if defined( _MSC_VER )
#   pragma warning(disable:4251) // dll interface for std types
#   ifdef SHIVA_EXPORT
#       define SHIVA_API __declspec(dllexport)
#   else
#       define SHIVA_API __declspec(dllimport)
#   endif
#else
#   ifdef SHIVA_EXPORT
#       define SHIVA_API __attribute__((visibility("default")))
#   else
#       define SHIVA_API
#   endif
#endif
```

### Included By

- *File DirectoryTree.hpp*
- *File Editor.hpp*
- *File ProjectBrowser.hpp*
- *File DockWindow.hpp*
- *File EditorWindowManager.hpp*
- *File AssetBrowser.hpp*

### Defines

- *Define SHIVA_API*

### File StringCast.h

*Parent directory* (include/ari)

**Contents**



## Definition (include/ari/StringCast.h)

## Program Listing for File StringCast.h

*Return to documentation for file* (include/ari/StringCast.h)

```
// In JSON map keys can only be strings, so here's a class which makes conversion to/
↪from string easy
#pragma once

#include <string>

template <typename T>
std::string castToString(const T& value);

// template specializations

std::string castToString(const bool& value);
std::string castToString(const int& value);
std::string castToString(const float& value);
std::string castToString(const std::string& value);

template <typename T>
T fromString(const std::string& value);

template <>
bool fromString(const std::string& valueStr);

template <>
int fromString(const std::string& valueStr);

template <>
float fromString(const std::string& valueStr);

template <>
std::string fromString(const std::string& valueStr);


// return empty string if no conversion possible
template <typename T>
std::string castToString(const T& /* value */)
{
    return std::string();
}

template <typename T>
```

(continues on next page)

(continued from previous page)

```
T fromString(const std::string& /* value */)
{
    return T();
}
```

### Includes

- string

### Included By

- *File JsonCast.h*

### Functions

- *Function castToString(const std::string&)*
- *Template Function castToString(const T&)*
- *Function castToString(const bool&)*
- *Function castToString(const int&)*
- *Function castToString(const float&)*
- *Template Function fromString(const std::string&)*
- *Function fromString(const std::string&)*
- *Function fromString(const std::string&)*
- *Function fromString(const std::string&)*
- *Function fromString(const std::string&)*

## File System.hpp

*Parent directory* (include/ari/en)

**Contents**

### Definition (include/ari/en/System.hpp)

### Program Listing for File System.hpp

*Return to documentation for file* (include/ari/en/System.hpp)

```
#pragma once
#include "../aridef.hpp"

namespace ari
{
    class World;

    class ARI_API System
    {
    public:

        enum class Type
        {
            GameplaySystem,
            SceneSystem,
            RenderSystem
        };

        enum class UpdateState
        {
            GameplayState,
            SceneManagerState,
            MainThreadState
        };

        System() = default;

        virtual ~System() = default;

        virtual void Update(World* p_world, UpdateState state) = 0;

        virtual void Configure(World* p_world) = 0;

        virtual void Unconfigure(World* p_world) = 0;

        virtual Type GetSystemType() = 0;

        virtual bool NeedUpdateOnState(UpdateState state) = 0;

    }; // System

} // ari
```

### Includes

- ../aridef.hpp

### Included By

- *File RenderSystem.hpp*
- *File SceneSystem.hpp*
- *File GuiSystem.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class System*

### File TextBox.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/TextBox.hpp`)

### Program Listing for File TextBox.hpp

*Return to documentation for file* (`include/ari/en/gui/TextBox.hpp`)

```
#pragma once
#include "Gui.hpp"

namespace ari
{
    class ARI_API TextBox: public Gui
    {
    public:

        TextBox(size_t maxLength = 128);

        ~TextBox() override;

        bool BeginRender() override;
```

(continues on next page)

(continued from previous page)

```
        void SetText(const char* _text) const;

        char* Text;
        char* Label;

    private:
        size_t m_MaxLength;

    };

} // ari
```

### Includes

- Gui.hpp (*File Gui.hpp*)

### Namespaces

- *Namespace ari*

### Classes

- *Class TextBox*

### File Texture.hpp

*Parent directory* (include/ari/gfx)

**Contents**



### Definition (include/ari/gfx/Texture.hpp)

### Program Listing for File Texture.hpp

*Return to documentation for file* (include/ari/gfx/Texture.hpp)

```
#pragma once
#include "../aridef.hpp"
#include "../Resource.hpp"
#include "bgfx/bgfx.h"
#include "bimg/bimg.h"

namespace ari
{
    struct TextureParams
    {
        uint32_t Flags = BGFX_TEXTURE_NONE;
        bgfx::TextureInfo* Info = nullptr;
        bimg::Orientation::Enum* Orientation = nullptr;
    };

    class ARI_API Texture: public Resource
    {
    public:

        Texture() = default;

        Texture(const uint32_t& _handel, const std::string& _fileName);

        ~Texture() override;

        bgfx::TextureHandle Handle = BGFX_INVALID_HANDLE;

    };

} // ari
```

### Includes

- ../Resource.hpp
- ../aridef.hpp
- bgfx/bgfx.h
- bimg/bimg.h

### Included By

- *File Viewport.hpp*
- *File Image.hpp*
- *File TextureManager.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Struct TextureParams*
- *Class Texture*

### File TextureManager.hpp

*Parent directory* (include/ari/gfx)

**Contents**



### Definition (include/ari/gfx/TextureManager.hpp)

### Program Listing for File TextureManager.hpp

*Return to documentation for file* (include/ari/gfx/TextureManager.hpp)

```
#pragma once
#include "../ResourceManager.hpp"
#include "Texture.hpp"

namespace ari
{
    class ARI_API TextureManager: public ResourceManager<Texture>
    {
    public:
        ~TextureManager() override;

    protected:

        bool LoadResource(Texture** ppOut, uint32_t handle,
            const std::string& filename, void* extraParams) override;

    };

} // ari
```

### Includes

- ../ResourceManager.hpp
- Texture.hpp (*File Texture.hpp*)

### Included By

- *File Engine.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Class TextureManager*

## File Tools.hpp

*Parent directory* (include/shiva/windows)

**Contents**



## Definition (include/shiva/windows/Tools.hpp)

### Program Listing for File Tools.hpp

*Return to documentation for file* (include/shiva/windows/Tools.hpp)

```
#pragma once

namespace shiva
{

} // shiva
```

### Namespaces

- *Namespace shiva*

## File Vector.hpp

*Parent directory* (include/ari/math)

**Contents**

- *Includes*
- *Included By*
- *Namespaces*
- *Classes*

### Definition (`include/ari/math/Vector.hpp`)

### Program Listing for File Vector.hpp

*Return to documentation for file* (`include/ari/math/Vector.hpp`)

```
#pragma once

#include <bx/math.h>
#include "arimath.hpp"

namespace ari
{
    struct Vector3
    {
        Vector3(): x(0.0f), y(0.0f), z(0.0f) { }

        Vector3(const float _x, const float _y, const float _z): x(_x), y(_y), z(_z)
↪{ }

        void Set(const float _x, const float _y, const float _z)
        {
            x = _x;
            y = _y;
            z = _z;
        }

        Vector3 operator-(const Vector3& v) const
        {
            return { x - v.x, y - v.y, z - v.z };
        }

        void Cross(const Vector3& _v1, const Vector3& _v2)
        {
            x = _v1.y*_v2.z - _v1.z*_v2.y;
            y = _v1.z*_v2.x - _v1.x*_v2.z;
            z = _v1.x*_v2.y - _v1.y*_v2.x;
        }

        float GetLength() const
        {
            return sqrtf(x*x + y * y + z * z);
        }

        void Normalize()
        {
            float length = GetLength();
            length = fEpsilon > length ? fEpsilon : length;
```

(continues on next page)

(continued from previous page)

```
            x /= length;
            y /= length;
            z /= length;
        }

        bx::Vec3 ToVec3() const
        {
            return {
                x,
                y,
                z
            };
        }

        union
        {
            float v[3];
            struct {
                float x,
                    y,
                    z;
            };
        };
    }; // Vector3

} // ari
```

### Includes

- `arimath.hpp` (*File arimath.hpp*)
- `bx/math.h`

### Included By

- *File Node3D.hpp*
- *File Matrix.hpp*

### Namespaces

- *Namespace ari*

### Classes

- *Struct Vector3*

### File Vertices.hpp

*Parent directory* (`include/ari/gfx`)

**Contents**



### Definition (include/ari/gfx/Vertices.hpp)

### Program Listing for File Vertices.hpp

*Return to documentation for file* (include/ari/gfx/Vertices.hpp)

```
#pragma once

namespace ari
{
    struct PosVertex
    {
        float x, y, z;
    };

    struct ColorVertex
    {
        uint32_t argb;
    };

} // ari
```

### Namespaces

- *Namespace ari*

### Classes

- *Struct ColorVertex*
- *Struct PosVertex*

### File Viewport.hpp

*Parent directory* (include/ari/en/2d)

**Contents**

### Definition (`include/ari/en/2d/Viewport.hpp`)

### Program Listing for File Viewport.hpp

*Return to documentation for file* (`include/ari/en/2d/Viewport.hpp`)

```
#pragma once
#include "../Component.hpp"
#include "../../math/Rect.hpp"
#include <bgfx/bgfx.h>
#include "../../gfx/Texture.hpp"

namespace ari
{
    class ARI_API Viewport: public Component
    {
    public:

        RectI                       Rect;
        bgfx::TextureFormat::Enum   TextureFormat = bgfx::TextureFormat::Count;
        bool                        CreateDepth = false;
        bool                        UseMSAA = false;

        // internal

        bgfx::FrameBufferHandle     m_frame_buffer_handle = BGFX_INVALID_HANDLE;
        RectI                       m_last_rect;
        bgfx::ViewId                m_view_id = 0;
        Texture                     m_texture,
                                    m_depth_texture;
    };

} // ari
```

### Includes

- `../../gfx/Texture.hpp`
- `../../math/Rect.hpp`
- `../Component.hpp`
- `bgfx/bgfx.h`

### Namespaces

- *Namespace ari*

### Classes

- *Class Viewport*

### File Viewport.hpp

*Parent directory* (include/shiva/windows)

**Contents**



### Definition (include/shiva/windows/Viewport.hpp)

### Program Listing for File Viewport.hpp

*Return to documentation for file* (include/shiva/windows/Viewport.hpp)

```
#pragma once
#include "DockWindow.hpp"

namespace ari
{
    class Camera;
    class Viewport;
    class Image;
}

namespace shiva
{
    class SHIVA_API Viewport : public DockWindow
    {
    public:

        void Init(ari::World* p_world) override;

    protected:

        ari::Camera         *   m_pCamera           = nullptr;
        ari::Viewport       *   m_pViewport         = nullptr;
        ari::Image          *   m_pView             = nullptr;
        ari::PlatformWindow *   m_pPlatformWindow   = nullptr;
        ari::DelegateTwoParam<void, int, int>
                                m_OnMouseMove;

        void OnGui();
        void OnHovered();
```

(continues on next page)

(continued from previous page)

```
    };

} // shiva
```

### Includes

- `DockWindow.hpp` (*File DockWindow.hpp*)

### Namespaces

- *Namespace ari*
- *Namespace shiva*

### Classes

- *Class Viewport*

### File Window.hpp

*Parent directory* (`include/ari/en/gui`)

**Contents**



### Definition (`include/ari/en/gui/Window.hpp`)

### Program Listing for File Window.hpp

*Return to documentation for file* (`include/ari/en/gui/Window.hpp`)

```
#pragma once
#include "Gui.hpp"
#include "dear-imgui/imgui.h"

namespace ari
{
    class ARI_API Window: public Gui
    {
    public:
        Window();
```

(continues on next page)

(continued from previous page)

```
        ~Window() = default;

        bool BeginRender() override;
        void EndRender() override;

        char     *   Name;
        bool         CloseButton,
                     isOpen;
        ImVec2       Pos,
                     Size;
        ImGuiWindowFlags    Flags;

    }; // Window

} // ari
```

### Includes

- `Gui.hpp` (*File Gui.hpp*)
- `dear-imgui/imgui.h`

### Namespaces

- *Namespace ari*

### Classes

- *Class Window*

### File World.hpp

*Parent directory* (`include/ari/en`)

**Contents**

### Definition (include/ari/en/World.hpp)

### Program Listing for File World.hpp

*Return to documentation for file* (include/ari/en/World.hpp)

```
#pragma once
#include "../aridef.hpp"
#include "tinystl/vector.h"
#include "EventSubscriber.hpp"
#include <algorithm>
#include <unordered_map>
#include "bx/spscqueue.h"

namespace ftl
{
    class TaskScheduler;
}

namespace ari
{
    class Node;
    class System;
    class Entity;

    class ARI_API World
    {
    public:

        enum class UpdateType
        {
            Sync,
            Async
        };

        World();

        ~World();

        void SetUpdateType(UpdateType type) { m_UpdateType = type; }

        UpdateType GetUpdateType() const { return m_UpdateType; }

        void AddSystem(System* p_system);

        void RemoveSystem(System* p_system);

        void AddEntity(Entity* p_entity);

        void RemoveEntity(Entity* p_entity);

        void Update(float tick);

        void _AddToDestroyQueue(Node* node);

        template<typename T>
        void subscribe(EventSubscriber<T>* subscriber)
```

(continues on next page)

(continued from previous page)

```
        {
            auto index = getTypeIndex<T>();
            auto found = subscribers.find(index);
            if (found == subscribers.end())
            {
                tinystl::vector<Internal::BaseEventSubscriber*> subList;
                subList.push_back(subscriber);

                subscribers.insert({ index, subList });
            }
            else
            {
                found->second.push_back(subscriber);
            }
        }

        template<typename T>
        void unsubscribe(EventSubscriber<T>* subscriber)
        {
            auto index = getTypeIndex<T>();
            auto found = subscribers.find(index);
            if (found != subscribers.end())
            {
                found->second.erase(std::remove(found->second.begin(), found->second.
↪end(), subscriber), found->second.end());
                if (found->second.size() == 0)
                {
                    subscribers.erase(found);
                }
            }
        }

        void unsubscribeAll(void* subscriber)
        {
            for (auto kv : subscribers)
            {
                kv.second.erase(std::remove(kv.second.begin(), kv.second.end(),␣
↪subscriber), kv.second.end());
                if (kv.second.empty())
                {
                    subscribers.erase(subscribers.find(kv.first));
                }
            }
        }


        template<typename T>
        void emit(const T& event)
        {
            auto found = subscribers.find(getTypeIndex<T>());
            if (found != subscribers.end())
            {
                for (auto* base : found->second)
                {
                    auto* sub = reinterpret_cast<EventSubscriber<T>*>(base);
                    sub->Receive(this, event);
                }
```

(continues on next page)

(continued from previous page)

```
            }
        }

        const tinystl::vector<Entity*>& GetAllEntities() const { return Entities; }

        ftl::TaskScheduler* GetTaskScheduler() const { return m_pTaskScheduler; }

    protected:

        std::unordered_map<TypeIndex,
            tinystl::vector<Internal::BaseEventSubscriber*>> subscribers;
        bx::SpScUnboundedQueueT<Node>   m_qDestroyQueue;
        tinystl::vector<System*> systems;
        tinystl::vector<Entity*> Entities;
        ftl::TaskScheduler  *   m_pTaskScheduler;
        UpdateType              m_UpdateType;

        void CheckDestroyQueue();

    }; // World

} // ari
```

### Includes

- ../aridef.hpp
- EventSubscriber.hpp (*File EventSubscriber.hpp*)
- algorithm
- bx/spscqueue.h
- tinystl/vector.h
- unordered_map

### Included By

- *File Editor.hpp*

### Namespaces

- *Namespace ari*
- *Namespace ftl*

### Classes

- *Class World*

### File WorldManager.hpp

*Parent directory* (include/ari/en)

**Contents**



### Definition (include/ari/en/WorldManager.hpp)

### Program Listing for File WorldManager.hpp

*Return to documentation for file* (include/ari/en/WorldManager.hpp)

```
#pragma once

namespace ari
{

} // ari
```

### Namespaces

- *Namespace ari*

CHAPTER 4

# Indices and tables

- genindex

# Index

## A

## T