cae

0.0.0

Generated by Doxygen 1.11.0

Chapter 1

cae

1.1 Cross-API-Engine

Cross-API-Engine is a fully modular and plugin-oriented engine architecture designed to decouple every subsystem into independently loadable runtime modules. Rather than focusing only on graphics API abstraction, the objective is to create an engine where every component can be swapped, extended, or replaced without modifying core code. This enables rapid experimentation, platform portability, performance benchmarking, and true freedom in engine architecture research.

1.1.1 Supported Platforms

Platform	Compiler	Build system	Status
Windows	MSVC (Visual Studio 2026 / 18.x)	MSBuild / Ninja	
Linux	GCC 13	Ninja / Make	
MacOS	GCC 13 / Clang	Ninja / Make	
iOS			
tvOS			
Android			
Web			

1.1.2 Quick Start

1.1.2.1 Prerequisites

Make sure you have the following dependencies installed on your system:

- CMake 4.0.0
- C++23

1.1.2.2 Build (UNIX: Linux, macOS)

```
cmake -S . -G "Ninja" -B cmake-build-release -DCMAKE_BUILD_TYPE=Release
cmake --build cmake-build-release --config Release
```

1.1.2.3 Build (Windows)

```
cmake -S . -G "Visual Studio 18 2026" -A "x64" -B cmake-build-release -DCMAKE_BUILD_TYPE=Release
cmake --build cmake-build-release --config Release
```

1.1.2.4 Run

```
./cmake-build-release/bin/cae -h
Usage: cae [options]

Options:
  -h, --help              Show this help message
  -v, --version           Show version information
  -c, --config <path>     Specify JSON configuration file
```

1.1.3 Documentation

- Online Doxygen documentation: Full engine architecture, APIs and design rationale.
- Modules: Overview of available modules and their responsibilities.
- Plugins: Plugin system and how to create new runtime modules.

1.1.4 Development

This section targets engine developers and contributors.

1.1.4.1 CMake Build options

- CAE_STRICT_WARNINGS (default: OFF): Enable strict warning level.
- CAE_ENABLE_SANITIZERS (default: OFF): Enable address and undefined sanitizers.
- CAE_ENABLE_LTO (default: OFF): Enable LTO on final targets.
- CAE_BUILD_TESTS (default: OFF): Enable building unit tests.
- CAE_CLANG_TIDY (default: OFF): Enable clang tidy usage.
- CAE_CLANG_FORMAT (default: OFF): Enable clang format usage.
- CAE_BUILD_DOC (default: OFF): Enable building doxygen documentation.

1.1.4.2 Testing

Unit tests are based on Google Test.
Use the CAE_BUILD_TESTS option.

```
./cmake-build-release/bin/cae-tests
```

1.1.4.3 Build Doxygen documentation

documentation is generated using Doxygen.
Use the CAE_BUILD_DOC option.

1.1.5 Third-party libraries

- Doxygen Awesome CSS: A custom CSS theme for Doxygen documentation.
- Google Test: A testing framework for C++.
- nlohmann-json: A JSON library for C++.

1.1.6 Contributing

Want to contribute? See Contributing guidelines.

1.1.7 Security

Please review our Security Policy for more information on reporting security vulnerabilities.

1.1.8 License

This project is licensed under the MIT License - see the License file for details.

Chapter 2

Audio plugin - ALSA

Chapter 3

Audio plugin - Core

Chapter 4

Audio plugin - OpenAL

Chapter 5

Audio Plugins

Platform	Lib
Windows	XAudio2
Linux	Alsa (Advanced Linux Sound Architecture)
MacOS	Core Audio
Cross-Platform	OpenAL
POSIX / Win32	PulseAudio

Chapter 6

Audio plugin - XAudio2

Chapter 7

Input Plugins

Chapter 8

Model plugin - Assimp

Chapter 9

Model plugins

Chapter 10

Network plugins

Chapter 11

Physic plugins

Chapter 12

Plugins

Type	Name	Windows	Linux	MacOS	Status
Audio	ALSA				
Audio	Core				
Audio	OpenAL				
Audio	XAudio2				
Model	Assimp				
Model	fastgltf				
Network	Asio				
Network	Posix				
Network	WinSock				
Physic					
Renderer	DirectX12				
Renderer	Metal				
Renderer	OpenGL				
Renderer	Vulkan				
Shader IR	SPIR-V				
Shader IR	DXC				
Shader Frontend	GLSL				
Shader Frontend	HLSL				
Shader Frontend	MSL				
Shader Frontend	WGSL				
UI	Imgui				
Window	Cocoa				
Window	GLFW				
Window	Win32				
Window	X11				

Legend: fully supported, work in progress / partial, not supported

Chapter 13

Renderer plugin - DirectX12

Chapter 14

Renderer plugin - Metal

Chapter 15

Renderer plugin - OpenGL

15.1 Dependancies

- GLAD - Multi-Language GL/GLES/EGL/GLX/WGL Loader-Generator.
- OpenGL - Cross-language, cross-platform API for rendering 2D and 3D vector graphics.

Warning

OpenGL was deprecated in macOS 10.14.

```
if (Wayland) use EGL;
else if (supports EGL + GL) use EGL;
else fallback to GLX;
```

15.2 Global arch

15.3 EGL

15.4 NSGL

Chapter 16

Renderer plugins

Chapter 17

Renderer plugin - Vulkan

17.1 Dependancies

- Vulkan headers - Official Vulkan API headers.
- Vulkan loader - The Vulkan loader is responsible for loading the Vulkan API and managing multiple Vulkan implementations.

Chapter 18

ShaderFrontend plugin - GLSL

18.1 Dependancies

- GLSLang - The Khronos reference compiler for GLSL and ESSL.

Chapter 19

ShaderFrontend plugin - HLSL

Chapter 20

ShaderFrontend plugin - MSL

Chapter 21

ShaderFrontend plugin - WGSL

Chapter 22

ShaderIR plugin - DXC

Chapter 23

ShaderIR plugin - SPIRV

23.1 Dependancies

- SPIRV-Headers - The official Khronos SPIR-V headers.
- SPIRV-cross - A tool and library for parsing and converting SPIR-V to other shader languages.

Chapter 24

Shader Plugins

frontend -> IR -> Renderer

24.1 Supported Languages & Target APIs

Language	Compatibility (renderer)	Compatibility (IR)
GLSL	OpenGL - Vulkan	SPIR-V → HLSL/MSL/WGSL
HLSL	DirectX	DXC → DXIL / SPIR-V → GLSL/MSL/WGSL
MSL	Metal	SPIR-V → MSL (SPIRV-Cross)
WGSL	WebGPU	SPIR-V → WGSL (via SPIRV-Cross)

24.2 Plugin Types

Plugin Type	Description
Frontend	compile human-readable shader source into SPIR-V
IR	post-process, optimize, or analyze SPIR-V modules, and transform to the target Backend

Chapter 25

UI plugin - Imgui

Chapter 26

UI plugins

Chapter 27

Window plugin - Cocoa

Chapter 28

Window plugin - GLFW

28.1 Dependancies

- GLFW - A cross-platform windowing and input library.

Chapter 29

Window plugins

Chapter 30

Window plugin - Win32

Chapter 31

Window plugin - X11

Chapter 32

Engine module

32.1 Dependencies

- GLM: A header-only C++ mathematics library.

Chapter 33

Modules

Module Name	Description
Engine	Core engine functionalities and systems
Interfaces	Abstractions for various subsystems
Utils	Utility functions and helpers

Chapter 34

Utils module

34.1 Dependencies

- stb: A single-header image loading library.

Chapter 35

Contributing to Cross API Engine

This guide ensures contributions are consistent, readable, and maintainable.

35.1 Contribution workflow

We strictly use GitHub Issues and Pull Requests (PRs). No direct commits to main are allowed.

35.1.1 Create / Find an Issue

Before contributing, make sure there is an open issue for your task. If not, create one using the provided templates:

- Bug Report
- Feature Request

35.1.2 Branching & PR Naming

Each contribution must be developed in a dedicated branch. Branches and Pull Requests must follow this naming convention:
<Type>/<Scope>/<Short-Description>

- Type → Feature | Fix | Refactor | Docs | Test | CI
- Scope → Core | Audio | Input | Network | Renderer | Window
- Short-Description → A concise summary of the change (use hyphens for spaces).

This ensures consistency across branches and PRs, making it easy to identify the purpose of each contribution at a glance.

35.2 Commit Norms

Commit Type	Description
build	Changes that affect the build system or external dependencies (npm, make, etc.)
ci	Changes related to integration files and scripts or configuration (Travis, Ansible, BrowserStack, etc.)
feat	Addition of a new feature
fix	Bug fix
perf	Performance improvements
refactor	Modification that neither adds a new feature nor improves performance
style	Change that does not affect functionality or semantics (indentation, formatting, adding space, renaming a variable, etc.)

docs	Writing or updating documentation
test	Addition or modification of tests

Chapter 36

LICENSE

Chapter 37

Namespace Index

37.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 38

Hierarchical Index

38.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 39

Class Index

39.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 40

File Index

40.1 File List

Here is a list of all files with brief descriptions:

Chapter 41

Namespace Documentation

41.1 cae Namespace Reference

Namespaces

- namespace AUDIO
- namespace CAMERA
- namespace LOG
- namespace MESSAGE
- namespace NETWORK
- namespace PLUGINS
- namespace RENDERER
- namespace USER
- namespace WINDOW

Classes

- interface AAudio

  Abstract class for audio.
- interface AInput

  Abstract class for inputs.
- interface AModel

  Abstract class for model.
- interface ANetwork

  Abstract class for network.
- interface APhysic

  Abstract class for physic.
- struct AppConfig

  Struct for application configuration.
- class Application

  Main class.
- interface ARenderer

  Abstract class for renderer.
- struct ArgsConfig

  Struct for command line arguments configuration.
- class ArgsHandler

  Class to handle command line arguments.
- interface AShaderFrontend

  Abstract class for shader frontend.
- interface AShaderIR

  Abstract class for shader IR.

- interface AUI
  Abstract class for ui.
- class AWindow
- class Camera
  Class for camera.
- struct Color
  Struct for color.
- class Engine
  Engine class.
- struct EngineConfig
  Struct for engine configuration.
- struct EnvConfig
  Struct for environment variables configuration.
- class GLFW
  Class for the GLFW plugin.
- class GLSL
  Class for the GLSL plugin.
- interface IAudio
  Interface for audio.
- interface IContext
  Interface for OpenGL context.
- interface IGamepad
  Interface for gamepad.
- interface IInput
  Interface for inputs.
- interface IKeyboard
  Interface for keyboard.
- interface IModel
  Interface for model.
- interface IMouse
  Interface for mouse.
- interface INetwork
  Interface for network.
- interface IPhysic
  Interface for physics.
- interface IRenderer
  Interface for renderer.
- interface IShaderFrontend
  Interface for shaders frontend.
- interface IShaderIR
  Interface for shaders IR.
- interface IUI
  Interface for ui.
- interface IWindow
  Abstract class for window.
- struct Mesh
- struct NativeWindowHandle
  Struct for native window handle.
- class OPGL
  Class for the OpenGL plugin.
- struct ShaderIRModule

Struct for shader intermediate representation module.

- class ShaderManager

  Class for managing shaders.

- struct ShaderPipelineDesc

  Struct for shader pipeline description.

- struct ShaderSourceDesc

  Struct for shader source description.

- class SPIRV

  Class for the SPIR-V IR plugin.

- class VULKN

  Class for the Vulkan plugin.

- struct WindowEvent

  Struct for window events.

- struct WindowSize

  Struct for window size.

Typedefs

- using ShaderID = std::string

Enumerations

- enum class GamepadButton : uint8_t {
  A = 0 , B , X , Y ,
  Back , Guide , Start , LThumb ,
  RThumb , LShoulder , RShoulder , DPadUp ,
  DPadDown , DPadLeft , DPadRight }
- enum class GamepadAxis : uint8_t {
  LeftX = 0 , LeftY , RightX , RightY ,
  TriggerLeft , TriggerRight }
- enum class KeyState : std::uint8_t { Pressed = 0 , Released = 1 , Held = 2 , Toggled = 3 }
- enum class KeyCode : uint8_t {
  A , B , C , D ,
  E , F , G , H ,
  I , J , K , L ,
  M , N , O , P ,
  Q , R , S , T ,
  U , V , W , X ,
  Y , Z , Num0 , Num1 ,
  Num2 , Num3 , Num4 , Num5 ,
  Num6 , Num7 , Num8 , Num9 ,
  Escape , F1 , F2 , F3 ,
  F4 , F5 , F6 , F7 ,
  F8 , F9 , F10 , F11 ,
  F12 , Left , Right , Up ,
  Down , Home , End , PageUp ,
  PageDown , Insert , Delete , Backspace ,
  Tab , Enter , Space , LShift ,
  RShift , LCtrl , RCtrl , LAlt ,
  RAlt , LSuper , RSuper , Numpad0 ,
  Numpad1 , Numpad2 , Numpad3 , Numpad4 ,
  Numpad5 , Numpad6 , Numpad7 , Numpad8 ,
  Numpad9 , NumpadAdd , NumpadSubtract , NumpadMultiply ,
  NumpadDivide , CapsLock , NumLock , ScrollLock ,
  PrintScreen , Pause , Menu , Count }

- enum class MouseButton : uint8_t { Left = 0 , Right , Middle , XButton1 , XButton2 , WheelUp , WheelDown , Count }
- enum class ShaderSourceType : uint8_t { GLSL = 0 , HLSL = 1 , WGSL = 2 , MSL = 3 , SPIRV = 4 , UNDEFINED = 255 }
- enum class ShaderStage : uint8_t { VERTEX = 0 , FRAGMENT = 1 , GEOMETRY = 2 , COMPUTE = 3 , UNDEFINED = 255 }
- enum class WindowEventType : uint8_t { KeyDown , KeyUp , MouseMove , MouseButtonDown , MouseButtonUp , MouseScroll , Resize , Focus , Close }

  Enum for window event types.

Variables

- constexpr auto VERSION = 450

41.1.1 Typedef Documentation

41.1.1.1 ShaderID

using cae::ShaderID = std::string
Definition at line 16 of file IShaderFrontend.hpp.

41.1.2 Enumeration Type Documentation

41.1.2.1 GamepadAxis

enum class cae::GamepadAxis : uint8_t [strong]

Enumerator

LeftX	
LeftY	
RightX	
RightY	
TriggerLeft	
TriggerRight	

Definition at line 32 of file Gamepad.hpp.

41.1.2.2 GamepadButton

enum class cae::GamepadButton : uint8_t [strong]

Enumerator

A	
B	
X	
Y	
Back	
Guide	
Start	
LThumb	

Enumerator

RThumb	
LShoulder	
RShoulder	
DPadUp	
DPadDown	
DPadLeft	
DPadRight	

Definition at line 13 of file Gamepad.hpp.

41.1.2.3 KeyCode

enum class cae::KeyCode : uint8_t [strong]

Enumerator

A	
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	
M	
N	
O	
P	
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	
Num0	
Num1	
Num2	
Num3	
Num4	
Num5	
Num6	
Num7	
Num8	

Enumerator

Num9	
Escape	
F1	
F2	
F3	
F4	
F5	
F6	
F7	
F8	
F9	
F10	
F11	
F12	
Left	
Right	
Up	
Down	
Home	
End	
PageUp	
PageDown	
Insert	
Delete	
Backspace	
Tab	
Enter	
Space	
LShift	
RShift	
LCtrl	
RCtrl	
LAlt	
RAlt	
LSuper	
RSuper	
Numpad0	
Numpad1	
Numpad2	
Numpad3	
Numpad4	
Numpad5	
Numpad6	
Numpad7	
Numpad8	
Numpad9	
NumpadAdd	

Enumerator

NumpadSubtract	
NumpadMultiply	
NumpadDivide	
CapsLock	
NumLock	
ScrollLock	
PrintScreen	
Pause	
Menu	
Count	

Definition at line 22 of file Keyboard.hpp.

41.1.2.4 KeyState

enum class cae::KeyState : std::uint8_t [strong]

Enumerator

Pressed	
Released	
Held	
Toggled	

Definition at line 14 of file Keyboard.hpp.

41.1.2.5 MouseButton

enum class cae::MouseButton : uint8_t [strong]

Enumerator

Left	
Right	
Middle	
XButton1	
XButton2	
WheelUp	
WheelDown	
Count	

Definition at line 13 of file Mouse.hpp.

41.1.2.6 ShaderSourceType

enum class cae::ShaderSourceType : uint8_t [strong]

Enumerator

GLSL	
HLSL	
WGSL	
MSL	

Enumerator

SPIRV	
UNDEFINED	

Definition at line 18 of file IShaderFrontend.hpp.

41.1.2.7 ShaderStage

enum class cae::ShaderStage : uint8_t [strong]

Enumerator

VERTEX	
FRAGMENT	
GEOMETRY	
COMPUTE	
UNDEFINED	

Definition at line 28 of file IShaderFrontend.hpp.

41.1.2.8 WindowEventType

enum class cae::WindowEventType : uint8_t [strong]
Enum for window event types.

Enumerator

KeyDown	
KeyUp	
MouseMove	
MouseButtonDown	
MouseButtonUp	
MouseScroll	
Resize	
Focus	
Close	

Definition at line 44 of file IWindow.hpp.

41.1.3 Variable Documentation

41.1.3.1 VERSION

auto cae::VERSION = 450 [constexpr]
Definition at line 18 of file GLSL.hpp.
Referenced by cae::GLSL::compileGLSLtoSPIRV().

41.2 cae::AUDIO Namespace Reference

Variables

- constexpr auto VOLUME = 1.F
- constexpr auto MUTED = false

41.2.1 Variable Documentation

41.2.1.1 MUTED

auto cae::AUDIO::MUTED = false [inline], [constexpr]
Definition at line 22 of file Common.hpp.

41.2.1.2 VOLUME

auto cae::AUDIO::VOLUME = 1.F [inline], [constexpr]
Definition at line 21 of file Common.hpp.

41.3 cae::CAMERA Namespace Reference

Variables

- constexpr auto NAME = ”Default name”
- constexpr auto MOVE_SPEED = 2.5F
- constexpr auto LOOK_SPEED = 10.0F
- constexpr auto FOV = 45.F
- constexpr auto NEAR_PLANE = 0.1F
- constexpr auto FAR_PLANE = 100.F

41.3.1 Variable Documentation

41.3.1.1 FAR_PLANE

auto cae::CAMERA::FAR_PLANE = 100.F [inline], [constexpr]
Definition at line 32 of file Common.hpp.

41.3.1.2 FOV

auto cae::CAMERA::FOV = 45.F [inline], [constexpr]
Definition at line 30 of file Common.hpp.

41.3.1.3 LOOK_SPEED

auto cae::CAMERA::LOOK_SPEED = 10.0F [inline], [constexpr]
Definition at line 29 of file Common.hpp.

41.3.1.4 MOVE_SPEED

auto cae::CAMERA::MOVE_SPEED = 2.5F [inline], [constexpr]
Definition at line 28 of file Common.hpp.

41.3.1.5 NAME

auto cae::CAMERA::NAME = ”Default name” [inline], [constexpr]
Definition at line 27 of file Common.hpp.

41.3.1.6 NEAR_PLANE

auto cae::CAMERA::NEAR_PLANE = 0.1F [inline], [constexpr]
Definition at line 31 of file Common.hpp.

41.4 cae::LOG Namespace Reference

Variables

- constexpr auto LOG_FPS = false

41.4.1 Variable Documentation

41.4.1.1 LOG_FPS

auto cae::LOG::LOG_FPS = false [inline], [constexpr]
Definition at line 37 of file Common.hpp.

41.5 cae::MESSAGE Namespace Reference

Variables

- static constexpr std::string_view HELP_MSG
- static constexpr std::string_view VERSION_MSG

41.5.1 Variable Documentation

41.5.1.1 HELP_MSG

std::string_view cae::MESSAGE::HELP_MSG [static], [constexpr]
Initial value:

```
= "Usage: " PROJECT_NAME APP_EXTENSION " [options]\n\n"
                                        "Options:\n"
                                        "  -h, --help              Show this help message\n"
                                        "  -v, --version           Show version information\n"
                                        "  -c, --config <path>     Specify JSON configuration file"
```

Definition at line 17 of file Common.hpp.
Referenced by cae::ArgsHandler::ParseArgs().

41.5.1.2 VERSION_MSG

std::string_view cae::MESSAGE::VERSION_MSG [static], [constexpr]
Initial value:

```
= PROJECT_NAME
          " v" PROJECT_VERSION " " BUILD_TYPE " (" GIT_TAG ", commit " GIT_COMMIT_HASH ") " __DATE__ "
      " __TIME__
```

Definition at line 22 of file Common.hpp.
Referenced by cae::Application::Application(), and cae::ArgsHandler::ParseArgs().

41.6 cae::NETWORK Namespace Reference

Variables

- constexpr auto HOST = "127.0.0.1"
- constexpr auto PORT = 4242

41.6.1 Variable Documentation

41.6.1.1 HOST

auto cae::NETWORK::HOST = "127.0.0.1" [inline], [constexpr]
Definition at line 42 of file Common.hpp.

41.6.1.2 PORT

auto cae::NETWORK::PORT = 4242 [inline], [constexpr]
Definition at line 43 of file Common.hpp.

41.7 cae::PLUGINS Namespace Reference

Namespaces

- namespace NAME

41.8 cae::PLUGINS::NAME Namespace Reference

Namespaces

- namespace RENDERER
- namespace SHADER
- namespace WINDOW

41.9 cae::PLUGINS::NAME::RENDERER Namespace Reference

Variables

- constexpr auto OPENGL = "OpenGL"
- constexpr auto VULKAN = "Vulkan"

41.9.1 Variable Documentation

41.9.1.1 OPENGL

auto cae::PLUGINS::NAME::RENDERER::OPENGL = "OpenGL" [inline], [constexpr]
Definition at line 30 of file Common.hpp.
Referenced by cae::Application::Application().

41.9.1.2 VULKAN

auto cae::PLUGINS::NAME::RENDERER::VULKAN = "Vulkan" [inline], [constexpr]
Definition at line 31 of file Common.hpp.

41.10 cae::PLUGINS::NAME::SHADER Namespace Reference

Namespaces

- namespace FRONTEND
- namespace IR

41.11 cae::PLUGINS::NAME::SHADER::FRONTEND Namespace Reference

Variables

- constexpr auto GLSL = "GLSL"

41.11.1 Variable Documentation

41.11.1.1 GLSL

auto cae::PLUGINS::NAME::SHADER::FRONTEND::GLSL = "GLSL" [inline], [constexpr]
Definition at line 43 of file Common.hpp.
Referenced by cae::Application::Application().

41.12 cae::PLUGINS::NAME::SHADER::IR Namespace Reference

Variables

- constexpr auto SPIRV = "SPIRV"

41.12.1 Variable Documentation

41.12.1.1 SPIRV

auto cae::PLUGINS::NAME::SHADER::IR::SPIRV = "SPIRV" [inline], [constexpr]
Definition at line 38 of file Common.hpp.
Referenced by cae::Application::Application().

41.13 cae::PLUGINS::NAME::WINDOW Namespace Reference

Variables

- constexpr auto COCOA = "Cocoa"
- constexpr auto GLFW = "GLFW"
- constexpr auto WIN32_ = "Win32"
- constexpr auto X11 = "X11"

41.13.1 Variable Documentation

41.13.1.1 COCOA

auto cae::PLUGINS::NAME::WINDOW::COCOA = "Cocoa" [inline], [constexpr]
Definition at line 50 of file Common.hpp.

41.13.1.2 GLFW

auto cae::PLUGINS::NAME::WINDOW::GLFW = "GLFW" [inline], [constexpr]
Definition at line 51 of file Common.hpp.
Referenced by cae::Application::Application().

41.13.1.3 WIN32_

auto cae::PLUGINS::NAME::WINDOW::WIN32_ = "Win32" [inline], [constexpr]
Definition at line 52 of file Common.hpp.

41.13.1.4 X11

auto cae::PLUGINS::NAME::WINDOW::X11 = "X11" [inline], [constexpr]
Definition at line 53 of file Common.hpp.

41.14 cae::RENDERER Namespace Reference

Variables

- constexpr auto VSYNC = false
- constexpr auto FRAME_RATE_LIMIT = 90
- constexpr auto CLEAR_COLOR_R = 1.0F
- constexpr auto CLEAR_COLOR_G = 1.0F
- constexpr auto CLEAR_COLOR_B = 1.0F
- constexpr auto CLEAR_COLOR_A = 1.0F

41.14.1 Variable Documentation

41.14.1.1 CLEAR_COLOR_A

auto cae::RENDERER::CLEAR_COLOR_A = 1.0F [inline], [constexpr]
Definition at line 53 of file Common.hpp.

41.14.1.2 CLEAR_COLOR_B

auto cae::RENDERER::CLEAR_COLOR_B = 1.0F [inline], [constexpr]
Definition at line 52 of file Common.hpp.

41.14.1.3 CLEAR_COLOR_G

auto cae::RENDERER::CLEAR_COLOR_G = 1.0F [inline], [constexpr]
Definition at line 51 of file Common.hpp.

41.14.1.4 CLEAR_COLOR_R

auto cae::RENDERER::CLEAR_COLOR_R = 1.0F [inline], [constexpr]
Definition at line 50 of file Common.hpp.

41.14.1.5 FRAME_RATE_LIMIT

auto cae::RENDERER::FRAME_RATE_LIMIT = 90 [inline], [constexpr]
Definition at line 49 of file Common.hpp.

41.14.1.6 VSYNC

auto cae::RENDERER::VSYNC = false [inline], [constexpr]
Definition at line 48 of file Common.hpp.

41.15 cae::USER Namespace Reference

Variables

- constexpr auto NAME = "User"

41.15.1 Variable Documentation

41.15.1.1 NAME

auto cae::USER::NAME = "User" [inline], [constexpr]
Definition at line 60 of file Common.hpp.

41.16 cae::WINDOW Namespace Reference

Variables

- constexpr uint16_t WIDTH = 960
- constexpr uint16_t HEIGHT = 540
- constexpr auto NAME = "Default name"
- constexpr auto FULLSCREEN = false
- constexpr auto ICON_PATH = ""

41.16.1 Variable Documentation

41.16.1.1 FULLSCREEN

auto cae::WINDOW::FULLSCREEN = false [inline], [constexpr]
Definition at line 61 of file Common.hpp.

41.16.1.2 HEIGHT

uint16_t cae::WINDOW::HEIGHT = 540 [inline], [constexpr]
Definition at line 59 of file Common.hpp.

41.16.1.3 ICON_PATH

auto cae::WINDOW::ICON_PATH = "" [inline], [constexpr]
Definition at line 62 of file Common.hpp.

41.16.1.4 NAME

auto cae::WINDOW::NAME = "Default name" [inline], [constexpr]
Definition at line 60 of file Common.hpp.

41.16.1.5 WIDTH

uint16_t cae::WINDOW::WIDTH = 960 [inline], [constexpr]
Definition at line 58 of file Common.hpp.

41.17 utl Namespace Reference

Classes

- class Clock
  Class for clock.
- struct Image
  Struct for image.
- interface IPlugin
  Interface for plugins.
- class Logger
  Class for logging.
- class Path
  Class for path resolution utilities.
- class PluginLoader
  Modern, type-safe plugin loader.
- struct SharedLib
  RAII wrapper for shared libraries.

Typedefs

- using LibHandle
- using EntryPointFn = IPlugin *(*)()

Enumerations

- enum class PluginType : uint8_t { AUDIO = 0 , NETWORK = 1 , RENDERER = 2 , SHADER_IR = 3 , SHADER_FRONTEND = 4 , WINDOW = 5 , UNDEFINED = 255 }
- enum class PluginPlatform : uint8_t { LINUX = 0 , MACOSX = 1 , WINDOWS = 2 , ALL = 255 }
- enum class LogLevel : uint8_t { INFO , WARNING }

Functions

- std::unordered_map< std::string, std::string > getEnvMap (const char *const *env)
  Get environment variables as a map.
- std::vector< char > fileToVector (const std::filesystem::path &path)
  Read a file and return its contents as a vector of chars.
- std::string fileToString (const std::filesystem::path &path)
  Read a file and return its contents as a string.

41.17.1 Typedef Documentation

41.17.1.1 EntryPointFn

using utl::EntryPointFn = IPlugin *(*)()
Definition at line 79 of file PluginLoader.hpp.

41.17.1.2 LibHandle

using utl::LibHandle
Initial value:

void *
Definition at line 29 of file PluginLoader.hpp.

41.17.2 Enumeration Type Documentation

41.17.2.1 LogLevel

enum class utl::LogLevel : uint8_t [strong]

Enumerator

INFO	
WARNING	

Definition at line 17 of file Logger.hpp.

41.17.2.2 PluginPlatform

enum class utl::PluginPlatform : uint8_t [strong]

Enumerator

LINUX	
MACOSX	
WINDOWS	
ALL	

Definition at line 32 of file IPlugin.hpp.

41.17.2.3 PluginType

enum class utl::PluginType : uint8_t [strong]

Enumerator

AUDIO	
NETWORK	
RENDERER	
SHADER_IR	
SHADER_FRONTEND	
WINDOW	
UNDEFINED	

Definition at line 21 of file IPlugin.hpp.

41.17.3 Function Documentation

41.17.3.1 fileToString()

std::string utl::fileToString (
const std::filesystem::path & path) [nodiscard]
Read a file and return its contents as a string.

Parameters

path	Path to the file

Returns

A string containing the file contents

Definition at line 27 of file file.cpp.
Referenced by cae::Application::start().
Here is the caller graph for this function:

41.17.3.2 fileToVector()

std::vector< char > utl::fileToVector (
const std::filesystem::path & path) [nodiscard]

Read a file and return its contents as a vector of chars.

Parameters

path	Path to the file

Returns

A vector of chars containing the file contents

Definition at line 6 of file file.cpp.

41.17.3.3 getEnvMap()

std::unordered_map< std::string, std::string > utl::getEnvMap (
const char *const * env) [nodiscard]

Get environment variables as a map.

Parameters

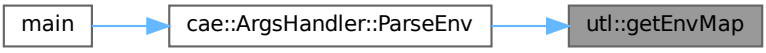
env	Pointer to environment variables

Returns

A map of environment variables

Definition at line 9 of file env.cpp.
Referenced by cae::ArgsHandler::ParseEnv().

Here is the caller graph for this function:

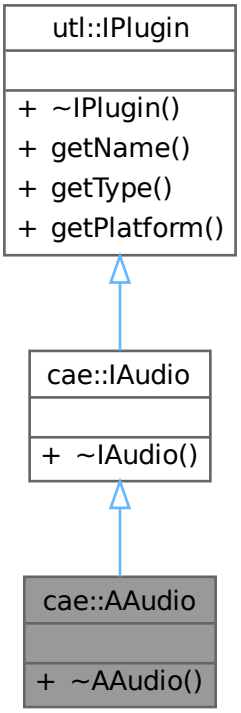


Chapter 42

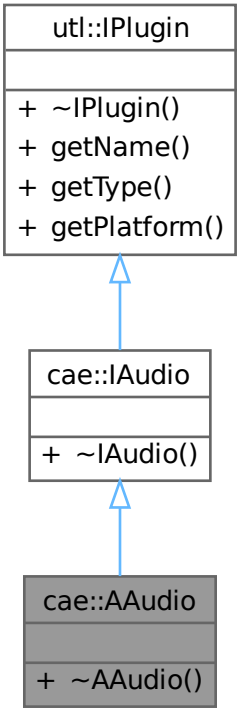
Class Documentation

42.1 cae::AAudio Interface Reference

Abstract class for audio.
#include <AAudio.hpp>
Inheritance diagram for cae::AAudio:



Collaboration diagram for cae::AAudio:



Public Member Functions

- ~AAudio () override=default

Public Member Functions inherited from cae::IAudio

- ~IAudio () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.1.1 Detailed Description

Abstract class for audio.
Definition at line 19 of file AAudio.hpp.

42.1.2 Constructor & Destructor Documentation

42.1.2.1 ~AAudio()

cae::AAudio::~AAudio () [override], [default]

The documentation for this interface was generated from the following file:

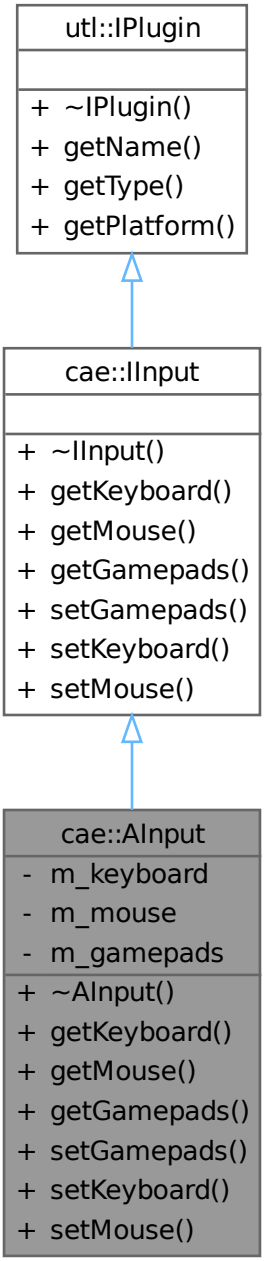
- modules/Interfaces/include/Interfaces/Audio/AAudio.hpp

42.2 cae::AInput Interface Reference

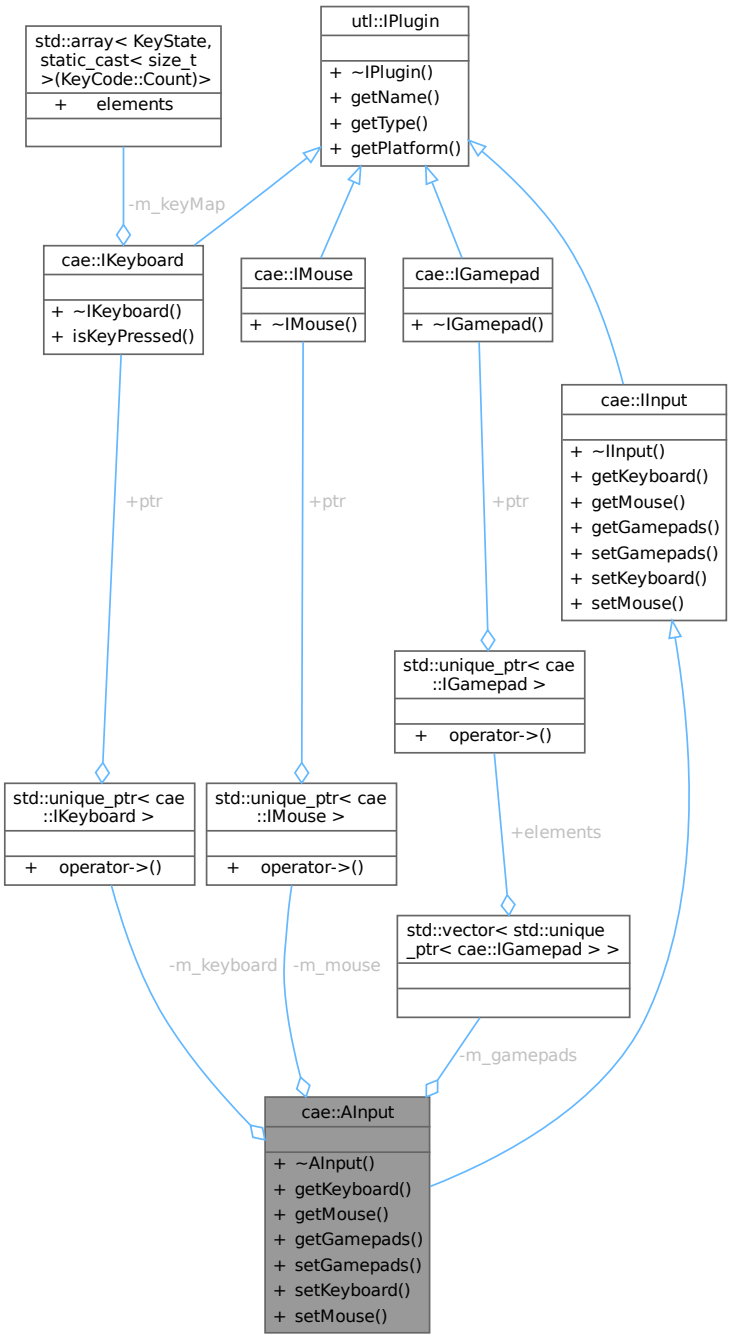
Abstract class for inputs.

#include <AInput.hpp>

Inheritance diagram for cae::AInput:



Collaboration diagram for cae::AInput:



Public Member Functions

- ~AInput () override=default
- const std::unique_ptr< IKeyboard > & getKeyboard () const override
- const std::unique_ptr< IMouse > & getMouse () const override
- const std::vector< std::unique_ptr< IGamepad > > & getGamepads () const override
- void setGamepads (std::vector< std::unique_ptr< IGamepad > > &gamepads) override
- void setKeyboard (std::unique_ptr< IKeyboard > &keyboard) override
- void setMouse (std::unique_ptr< IMouse > &mouse) override

Public Member Functions inherited from cae::IInput

- ~IInput () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

Private Attributes

- std::unique_ptr< IKeyboard > m_keyboard
- std::unique_ptr< IMouse > m_mouse
- std::vector< std::unique_ptr< IGamepad > > m_gamepads

42.2.1 Detailed Description

Abstract class for inputs.
Definition at line 19 of file AInput.hpp.

42.2.2 Constructor & Destructor Documentation

42.2.2.1 ~AInput()

cae::AInput::~AInput () [override], [default]

42.2.3 Member Function Documentation

42.2.3.1 getGamepads()

const std::vector< std::unique_ptr< IGamepad > > & cae::AInput::getGamepads () const [inline], [override], [virtual]
Implements cae::IInput.
Definition at line 27 of file AInput.hpp.
References m_gamepads.

42.2.3.2 getKeyboard()

const std::unique_ptr< IKeyboard > & cae::AInput::getKeyboard () const [inline], [override], [virtual]
Implements cae::IInput.
Definition at line 25 of file AInput.hpp.
References m_keyboard.

42.2.3.3 getMouse()

const std::unique_ptr< IMouse > & cae::AInput::getMouse () const [inline], [override], [virtual]
Implements cae::IInput.
Definition at line 26 of file AInput.hpp.
References m_mouse.

42.2.3.4 setGamepads()

void cae::AInput::setGamepads (
std::vector< std::unique_ptr< IGamepad > > & gamepads) [inline], [override], [virtual]
Implements cae::IInput.
Definition at line 29 of file AInput.hpp.
References m_gamepads.

42.2.3.5 setKeyboard()

void cae::AInput::setKeyboard (
std::unique_ptr< IKeyboard > & keyboard) [inline], [override], [virtual]
Implements cae::IInput.
Definition at line 33 of file AInput.hpp.
References m_keyboard.

42.2.3.6 setMouse()

void cae::AInput::setMouse (
std::unique_ptr< IMouse > & mouse) [inline], [override], [virtual]
Implements cae::IInput.
Definition at line 34 of file AInput.hpp.
References m_mouse.

42.2.4 Member Data Documentation

42.2.4.1 m_gamepads

std::vector<std::unique_ptr<IGamepad> > cae::AInput::m_gamepads [private]
Definition at line 39 of file AInput.hpp.
Referenced by getGamepads(), and setGamepads().

42.2.4.2 m_keyboard

std::unique_ptr<IKeyboard> cae::AInput::m_keyboard [private]
Definition at line 37 of file AInput.hpp.
Referenced by getKeyboard(), and setKeyboard().

42.2.4.3 m_mouse

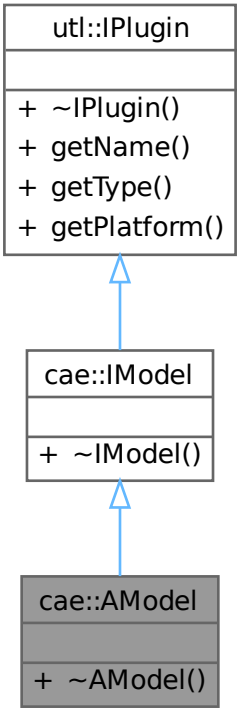
std::unique_ptr<IMouse> cae::AInput::m_mouse [private]
Definition at line 38 of file AInput.hpp.
Referenced by getMouse(), and setMouse().
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Input/AInput.hpp

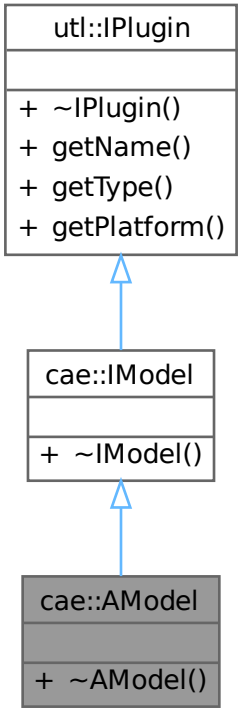
42.3 cae::AModel Interface Reference

Abstract class for model.
#include <AModel.hpp>

Inheritance diagram for cae::AModel:



Collaboration diagram for cae::AModel:



Public Member Functions

- ~AModel () override=default

Public Member Functions inherited from cae::IModel

- ~IModel () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.3.1 Detailed Description

Abstract class for model.
Definition at line 19 of file AModel.hpp.

42.3.2 Constructor & Destructor Documentation

42.3.2.1 ~AModel()

cae::AModel::~AModel () [override], [default]

The documentation for this interface was generated from the following file:

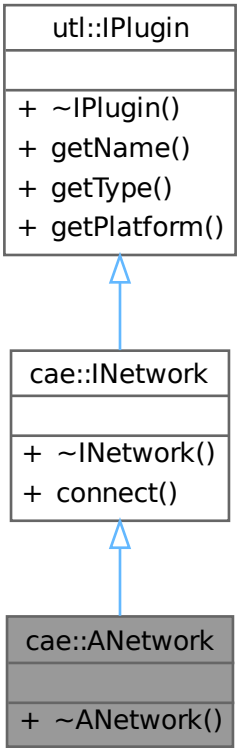
- modules/Interfaces/include/Interfaces/Model/AModel.hpp

42.4 cae::ANetwork Interface Reference

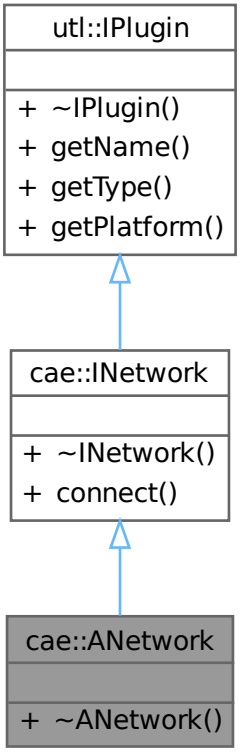
Abstract class for network.

#include <ANetwork.hpp>

Inheritance diagram for cae::ANetwork:



Collaboration diagram for cae::ANetwork:



Public Member Functions

- ~ANetwork () override=default

Public Member Functions inherited from cae::INetwork

- ~INetwork () override=default
- virtual bool connect (const std::string &host, uint16_t port)=0

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.4.1 Detailed Description

Abstract class for network.
Definition at line 19 of file ANetwork.hpp.

42.4.2 Constructor & Destructor Documentation

42.4.2.1 ~ANetwork()

cae::ANetwork::~ANetwork () [override], [default]

The documentation for this interface was generated from the following file:

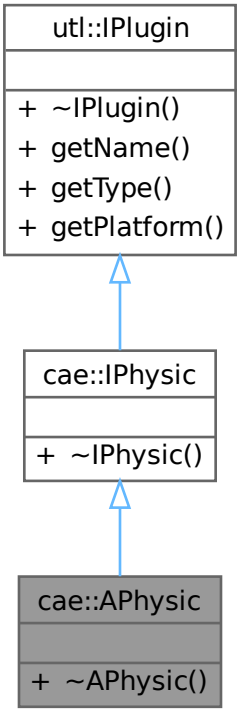
- modules/Interfaces/include/Interfaces/Network/ANetwork.hpp

42.5 cae::APhysic Interface Reference

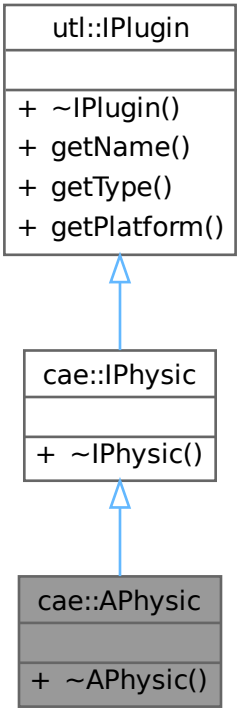
Abstract class for physic.

#include <APhysic.hpp>

Inheritance diagram for cae::APhysic:



Collaboration diagram for cae::APhysic:



Public Member Functions

- ~APhysic () override=default

Public Member Functions inherited from cae::IPhysic

- ~IPhysic () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.5.1 Detailed Description

Abstract class for physic.
Definition at line 19 of file APhysic.hpp.

42.5.2 Constructor & Destructor Documentation

42.5.2.1 ~APhysic()

cae::APhysic::~APhysic () [override], [default]

The documentation for this interface was generated from the following file:

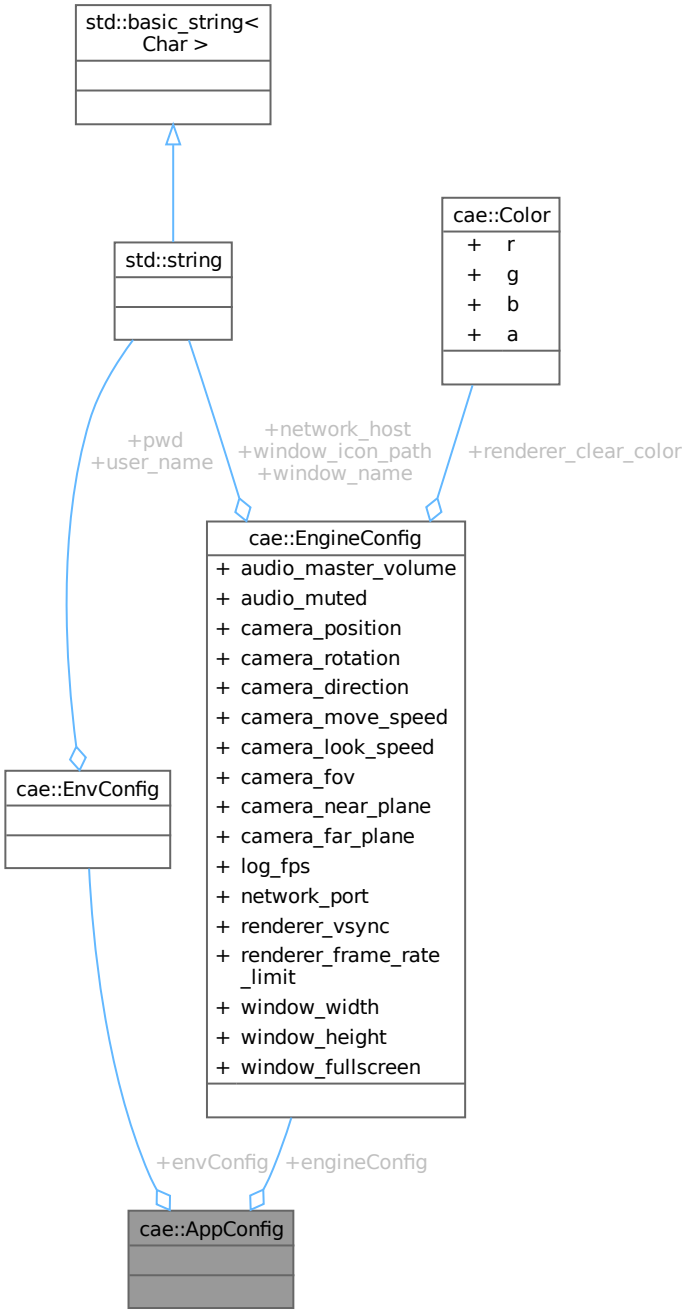
- modules/Interfaces/include/Interfaces/Physic/APhysic.hpp

42.6 cae::AppConfig Struct Reference

Struct for application configuration.

#include <Application.hpp>

Collaboration diagram for cae::AppConfig:



Public Attributes

- EngineConfig engineConfig
- EnvConfig envConfig

42.6.1 Detailed Description

Struct for application configuration.
Definition at line 22 of file Application.hpp.

42.6.2 Member Data Documentation

42.6.2.1 engineConfig

EngineConfig cae::AppConfig::engineConfig
Definition at line 24 of file Application.hpp.
Referenced by cae::Application::Application().

42.6.2.2 envConfig

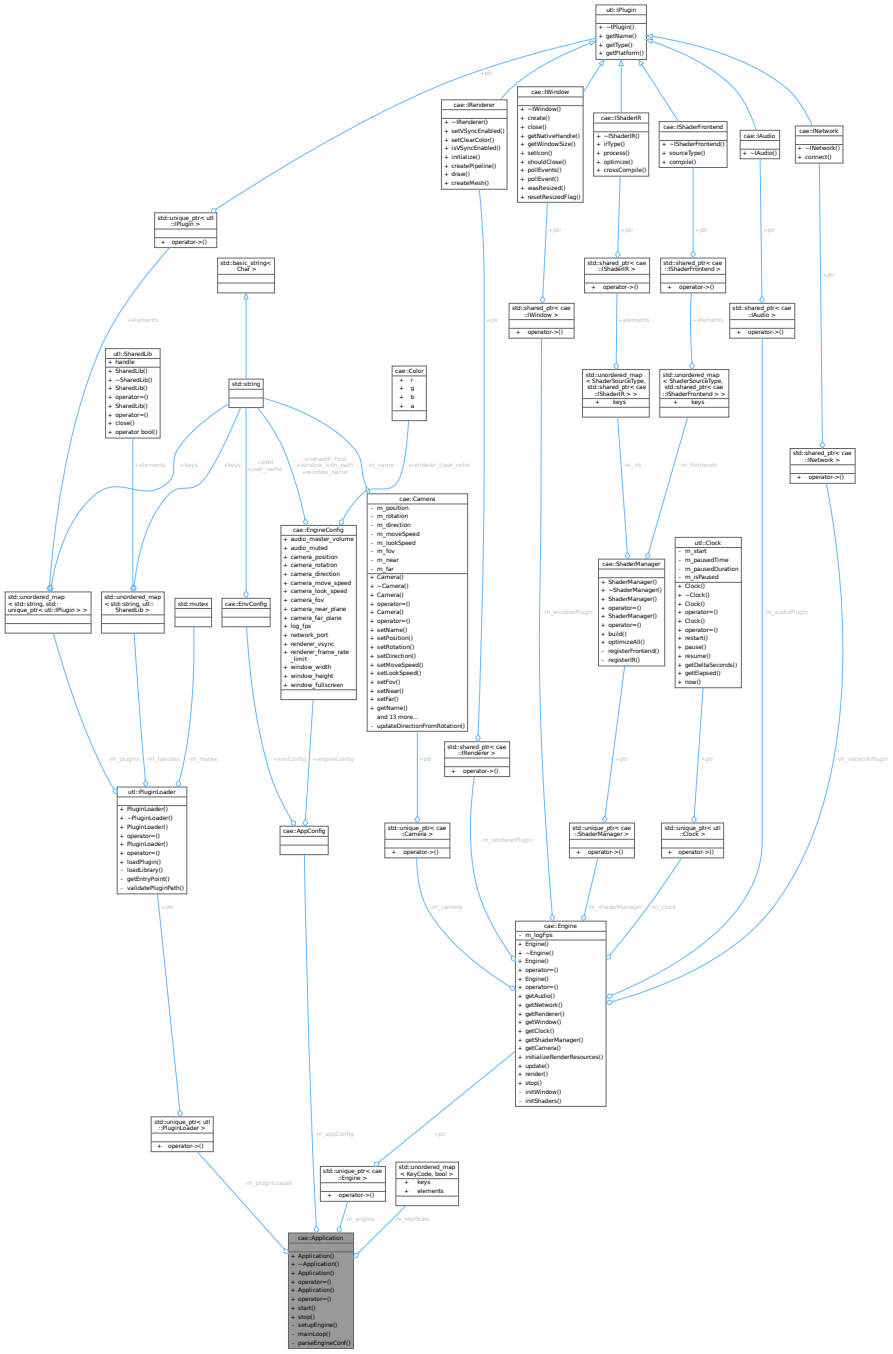
EnvConfig cae::AppConfig::envConfig
Definition at line 25 of file Application.hpp.
Referenced by cae::Application::Application().
The documentation for this struct was generated from the following file:

- include/CAE/Application.hpp

42.7 cae::Application Class Reference

Main class.
#include <Application.hpp>

Collaboration diagram for cae::Application:



Public Member Functions

- Application (const ArgsConfig &argsConfig, const EnvConfig &envConfig)

  Construct the Application with given configurations.
- ~Application ()=default
- Application (const Application &)=delete
- Application & operator= (const Application &)=delete
- Application (Application &&)=delete
- Application & operator= (Application &&)=delete
- void start ()

Start the application.

- void stop ()

  Stop the application.

Private Member Functions

- void setupEngine (const std::string &rendererName, const std::string &windowName, const std::string &shaderFrontendName, const std::string &shaderIRName)

  Setup the engine with the given plugins.

- void mainLoop ()

  main loop

Static Private Member Functions

- static EngineConfig parseEngineConf (const std::string &path)

  Parse the engine configuration file.

Private Attributes

- std::unique_ptr< utl::PluginLoader > m_pluginLoader = nullptr
- std::unique_ptr< Engine > m_engine = nullptr
- AppConfig m_appConfig
- std::unordered_map< KeyCode, bool > m_keyState

42.7.1 Detailed Description

Main class.
Definition at line 33 of file Application.hpp.

42.7.2 Constructor & Destructor Documentation

42.7.2.1 Application() [1/3]

cae::Application::Application (
const ArgsConfig & argsConfig,
const EnvConfig & envConfig)

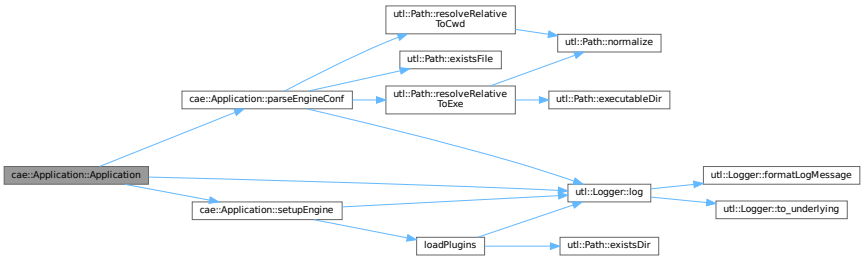
Construct the Application with given configurations.

Parameters

argsConfig	Command line arguments configuration
envConfig	Environment variables configuration

Definition at line 43 of file application.cpp.
References cae::ArgsConfig::config_path, cae::AppConfig::engineConfig, cae::AppConfig::envConfig, cae::PLUGINS::NAME::WINDOW::GLFW, cae::PLUGINS::NAME::SHADER::FRONTEND::GLSL, utl::INFO, utl::Logger::log(), m_appConfig, cae::PLUGINS::NAME::RENDERER::OPENGL, parseEngineConf(), setupEngine(), cae::PLUGINS::NAME::SHADER::IR::SPIRV, cae::MESSAGE::VERSION_MSG, and utl::WARNING.

Here is the call graph for this function:



42.7.2.2 ~Application()

cae::Application::~Application () [default]

42.7.2.3 Application() [2/3]

cae::Application::Application (
const Application &) [delete]

42.7.2.4 Application() [3/3]

cae::Application::Application (
Application &&) [delete]

42.7.3 Member Function Documentation

42.7.3.1 mainLoop()

void cae::Application::mainLoop () [private]
main loop
Definition at line 167 of file application.cpp.
References cae::A, cae::D, cae::Down, cae::KeyDown, cae::KeyUp, cae::LCtrl, cae::Left, cae::Right, cae::S, cae::Space, cae::Up, and cae::W.

42.7.3.2 operator=() [1/2]

Application & cae::Application::operator= (
Application &&) [delete]

42.7.3.3 operator=() [2/2]

Application & cae::Application::operator= (
const Application &) [delete]

42.7.3.4 parseEngineConf()

cae::EngineConfig cae::Application::parseEngineConf (
const std::string & path) [static], [private]
Parse the engine configuration file.

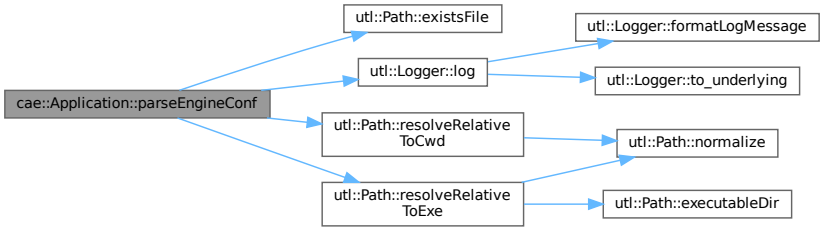
Parameters

path	Path to the engine configuration file

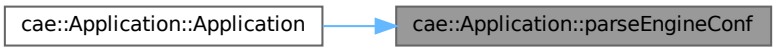
Returns

Parsed EngineConfig

Definition at line 12 of file conf.cpp.
References cae::Color::a, cae::EngineConfig::audio_master_volume, cae::EngineConfig::audio_muted, cae::Color::b, cae::EngineConfig::camera_direction, cae::EngineConfig::camera_far_plane, cae::EngineConfig::camera_fo cae::EngineConfig::camera_look_speed, cae::EngineConfig::camera_move_speed, cae::EngineConfig::camera_near_plan cae::EngineConfig::camera_position, cae::EngineConfig::camera_rotation, utl::Path::existsFile(), cae::Color::g, utl::INFO, utl::Logger::log(), cae::EngineConfig::log_fps, cae::EngineConfig::network_host, cae::EngineConfig::network_port, cae::Color::r, cae::EngineConfig::renderer_clear_color, cae::EngineConfig::renderer_fra cae::EngineConfig::renderer_vsync, utl::Path::resolveRelativeToCwd(), utl::Path::resolveRelativeToExe(), utl::WARNING, cae::EngineConfig::window_fullscreen, cae::EngineConfig::window_height, cae::EngineConfig::window_i cae::EngineConfig::window_name, and cae::EngineConfig::window_width.
Referenced by Application().
Here is the call graph for this function:



Here is the caller graph for this function:



42.7.3.5 setupEngine()

void cae::Application::setupEngine (
const std::string & rendererName,
const std::string & windowName,
const std::string & shaderFrontendName,
const std::string & shaderIRName) [private]

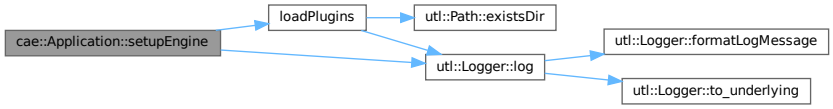
Setup the engine with the given plugins.

Parameters

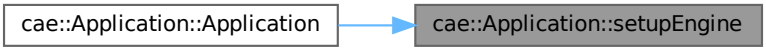
rendererName	renderer plugin name
windowName	window plugin name
shaderFrontendName	shader frontend plugin name
shaderIRName	shader IR plugin name

Definition at line 65 of file application.cpp.
References loadPlugins(), utl::Logger::log(), and utl::WARNING.

Referenced by Application().
Here is the call graph for this function:

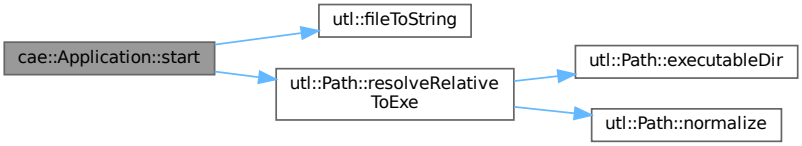


Here is the caller graph for this function:



42.7.3.6 start()

void cae::Application::start ()
Start the application.
Definition at line 142 of file application.cpp.
References cubeVertices, utl::fileToString(), cae::FRAGMENT, cae::GLSL, utl::Path::resolveRelativeToExe(), and cae::VERTEX.
Here is the call graph for this function:



42.7.3.7 stop()

void cae::Application::stop ()
Stop the application.
Definition at line 159 of file application.cpp.

42.7.4 Member Data Documentation

42.7.4.1 m_appConfig

AppConfig cae::Application::m_appConfig [private]
Definition at line 86 of file Application.hpp.
Referenced by Application().

42.7.4.2 m_engine

std::unique_ptr<Engine> cae::Application::m_engine = nullptr [private]
Definition at line 84 of file Application.hpp.

42.7.4.3 m_keyState

std::unordered_map<KeyCode, bool> cae::Application::m_keyState [private]
Definition at line 87 of file Application.hpp.

42.7.4.4 m_pluginLoader

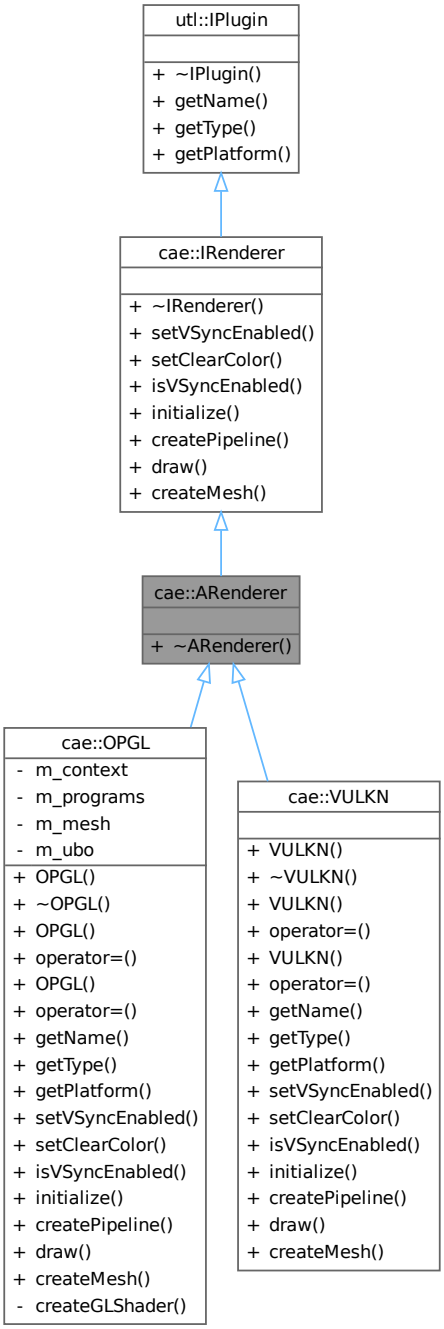
std::unique_ptr<utl::PluginLoader> cae::Application::m_pluginLoader = nullptr [private]
Definition at line 83 of file Application.hpp.
The documentation for this class was generated from the following files:

- include/CAE/Application.hpp
- src/application.cpp
- src/conf.cpp

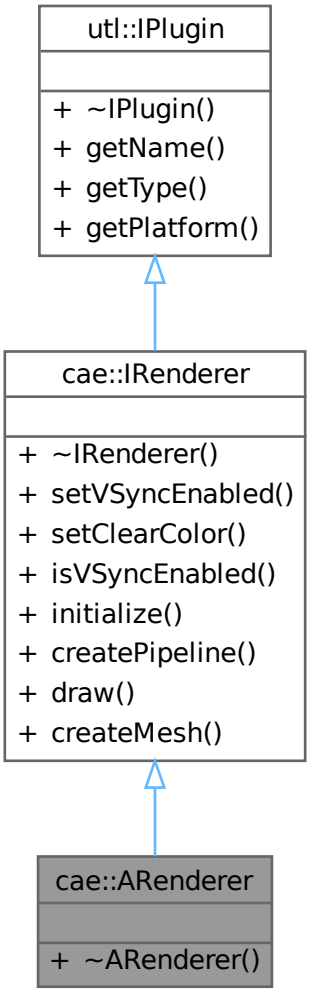
42.8 cae::ARenderer Interface Reference

Abstract class for renderer.
#include <ARenderer.hpp>

Inheritance diagram for cae::ARenderer:



Collaboration diagram for cae::ARenderer:



Public Member Functions

- ~ARenderer () override=default

Public Member Functions inherited from cae::IRenderer

- ~IRenderer () override=default
- virtual void setVSyncEnabled (bool enabled)=0

  Enable or disable VSync.
- virtual void setClearColor (const Color &color)=0

  Set the clear color.
- virtual bool isVSyncEnabled () const =0

  Check if VSync is enabled.
- virtual void initialize (const NativeWindowHandle &nativeWindowHandle, const Color &clear↩ Color={.r=1.F,.g=1.F,.b=1.F,.a=1.F})=0

  Initialize the renderer with a native window handle and clear color.

- virtual void createPipeline (const ShaderID &id, const ShaderIRModule &vertex, const ShaderIRModule &fragment)=0

  Create a rendering pipeline with vertex and fragment shaders.
- virtual void draw (const WindowSize &windowSize, const ShaderID &shaderId, glm::mat4 mvp)=0

  Draw the scene using the specified shader and window size.
- virtual void createMesh (const std::vector< float > &vertices)=0

  Create a mesh with the given vertex data.

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.8.1 Detailed Description

Abstract class for renderer.
Definition at line 19 of file ARenderer.hpp.

42.8.2 Constructor & Destructor Documentation

42.8.2.1 ~ARenderer()

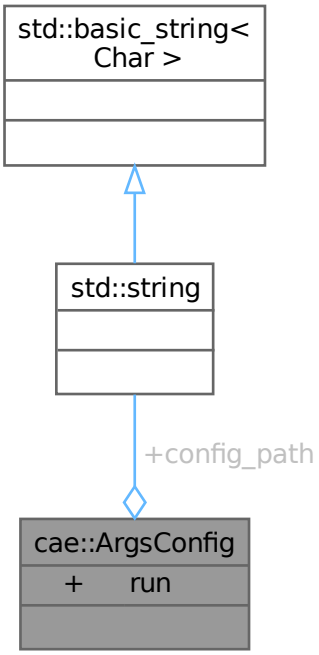
cae::ARenderer::~ARenderer () [override], [default]
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Renderer/ARenderer.hpp

42.9 cae::ArgsConfig Struct Reference

Struct for command line arguments configuration.
#include <ArgsHandler.hpp>

Collaboration diagram for cae::ArgsConfig:



Public Attributes

- bool run = false
- std::string config_path

42.9.1 Detailed Description

Struct for command line arguments configuration.
Definition at line 19 of file ArgsHandler.hpp.

42.9.2 Member Data Documentation

42.9.2.1 config_path

std::string cae::ArgsConfig::config_path
Definition at line 22 of file ArgsHandler.hpp.
Referenced by cae::Application::Application(), and cae::ArgsHandler::ParseArgs().

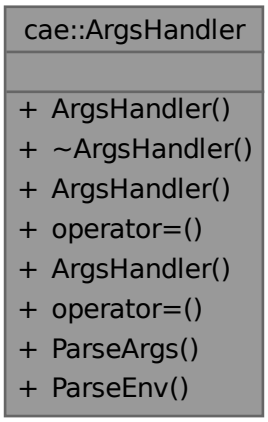
42.9.2.2 run

bool cae::ArgsConfig::run = false
Definition at line 21 of file ArgsHandler.hpp.
Referenced by main(), and cae::ArgsHandler::ParseArgs().
The documentation for this struct was generated from the following file:

- include/CAE/ArgsHandler.hpp

42.10 cae::ArgsHandler Class Reference

Class to handle command line arguments.
#include <ArgsHandler.hpp>
Collaboration diagram for cae::ArgsHandler:



Public Member Functions

- ArgsHandler ()=default
- ~ArgsHandler ()=default
- ArgsHandler (const ArgsHandler &)=delete
- ArgsHandler & operator= (const ArgsHandler &)=delete
- ArgsHandler (ArgsHandler &&)=delete
- ArgsHandler & operator= (ArgsHandler &&)=delete

Static Public Member Functions

- static ArgsConfig ParseArgs (int argc, const char *const *argv)
  Parse command line arguments.
- static EnvConfig ParseEnv (const char *const *envp)
  Parse environment variables.

42.10.1 Detailed Description

Class to handle command line arguments.
Definition at line 41 of file ArgsHandler.hpp.

42.10.2 Constructor & Destructor Documentation

42.10.2.1 ArgsHandler() [1/3]

cae::ArgsHandler::ArgsHandler () [default]

42.10.2.2 ~ArgsHandler()

cae::ArgsHandler::~ArgsHandler () [default]

42.10.2.3 ArgsHandler() [2/3]

cae::ArgsHandler::ArgsHandler (
const ArgsHandler &) [delete]

42.10.2.4 ArgsHandler() [3/3]

cae::ArgsHandler::ArgsHandler (
ArgsHandler &&) [delete]

42.10.3 Member Function Documentation

42.10.3.1 operator=() [1/2]

ArgsHandler & cae::ArgsHandler::operator= (
ArgsHandler &&) [delete]

42.10.3.2 operator=() [2/2]

ArgsHandler & cae::ArgsHandler::operator= (
const ArgsHandler &) [delete]

42.10.3.3 ParseArgs()

cae::ArgsConfig cae::ArgsHandler::ParseArgs (
int argc,
const char *const * argv) [static]

Parse command line arguments.

Parameters

argc	argument count
argv	argument vector

Returns

Parsed ArgsConfig

Definition at line 8 of file argsHandler.cpp.
References cae::ArgsConfig::config_path, cae::MESSAGE::HELP_MSG, cae::ArgsConfig::run, and cae::MESSAGE::VERSION_MSG.
Referenced by main().
Here is the caller graph for this function:

42.10.3.4 ParseEnv()

cae::EnvConfig cae::ArgsHandler::ParseEnv (
const char *const * envp) [static]

Parse environment variables.

Parameters

envp	environment pointer

Returns

Parsed EnvConfig

Definition at line 52 of file argsHandler.cpp.
References utl::getEnvMap(), cae::EnvConfig::pwd, and cae::EnvConfig::user_name.
Referenced by main().
Here is the call graph for this function:

Here is the caller graph for this function:

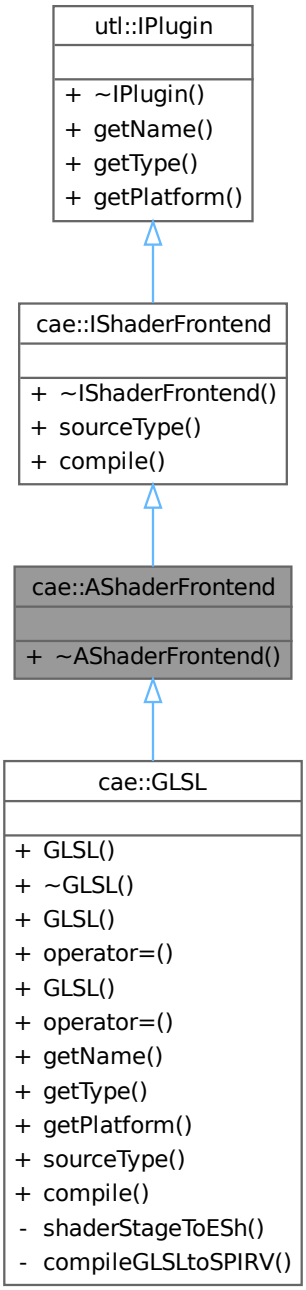
The documentation for this class was generated from the following files:

- include/CAE/ArgsHandler.hpp
- src/argsHandler.cpp

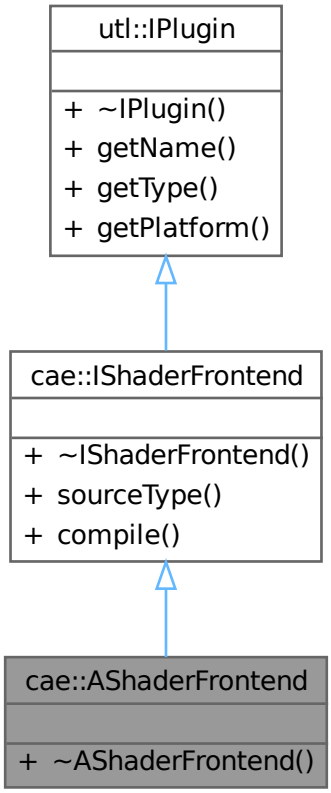
42.11 cae::AShaderFrontend Interface Reference

Abstract class for shader frontend.
#include <AShaderFrontend.hpp>

Inheritance diagram for cae::AShaderFrontend:



Collaboration diagram for cae::AShaderFrontend:



Public Member Functions

- ~AShaderFrontend () override=default

Public Member Functions inherited from cae::IShaderFrontend

- ~IShaderFrontend () override=default
- virtual ShaderSourceType sourceType () const =0

  Get the source type this frontend handles.
- virtual ShaderIRModule compile (const ShaderSourceDesc &desc)=0

  Compile shader source to intermediate representation.

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.11.1 Detailed Description

Abstract class for shader frontend.
Definition at line 19 of file AShaderFrontend.hpp.

42.11.2 Constructor & Destructor Documentation

42.11.2.1 ~AShaderFrontend()

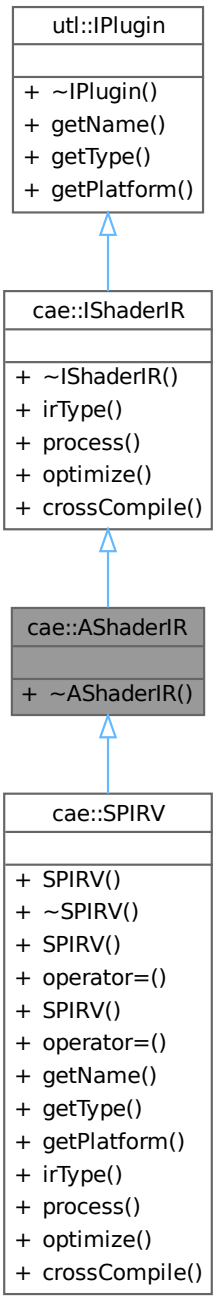
cae::AShaderFrontend::~AShaderFrontend () [override], [default]
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Shader/Frontend/AShaderFrontend.hpp

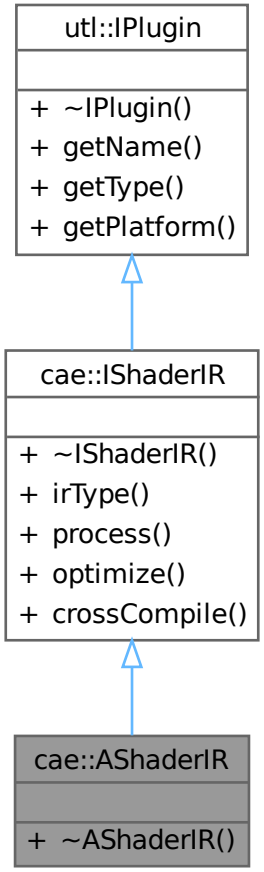
42.12 cae::AShaderIR Interface Reference

Abstract class for shader IR.
#include <AShaderIR.hpp>

Inheritance diagram for cae::AShaderIR:



Collaboration diagram for cae::AShaderIR:



Public Member Functions

- ~AShaderIR () override=default

Public Member Functions inherited from cae::IShaderIR

- ~IShaderIR () override=default
- virtual ShaderSourceType irType () const =0

  Get the IR type this processor handles.
- virtual ShaderIRModule process (const ShaderIRModule &module)=0

  Transform or validate a shader IR module.
- virtual void optimize (std::span< ShaderIRModule > modules)

  Optional: optimize a batch of IR modules.
- virtual ShaderIRModule crossCompile (const ShaderIRModule &module, ShaderSourceType targetType)

  Optional: cross-compile from one IR to another (SPIR-V -> MSL, etc.)

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default

- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.12.1 Detailed Description

Abstract class for shader IR.
Definition at line 19 of file AShaderIR.hpp.

42.12.2 Constructor & Destructor Documentation

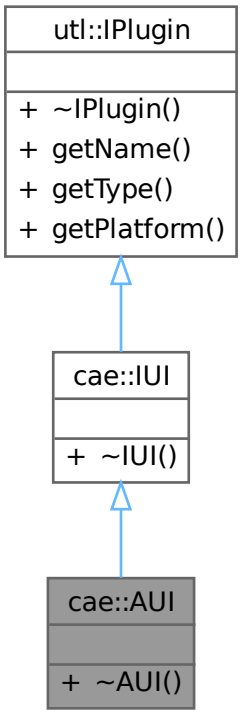
42.12.2.1 ~AShaderIR()

cae::AShaderIR::~AShaderIR () [override], [default]
The documentation for this interface was generated from the following file:

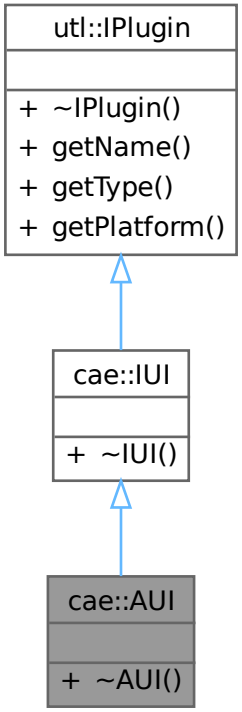
- modules/Interfaces/include/Interfaces/Shader/IR/AShaderIR.hpp

42.13 cae::AUI Interface Reference

Abstract class for ui.
#include <AUI.hpp>
Inheritance diagram for cae::AUI:



Collaboration diagram for cae::AUI:



Public Member Functions

- ~AUI () override=default

Public Member Functions inherited from cae::IUI

- ~IUI () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.13.1 Detailed Description

Abstract class for ui.
Definition at line 19 of file AUI.hpp.

42.13.2 Constructor & Destructor Documentation

42.13.2.1 ~AUI()

cae::AUI::~AUI () [override], [default]

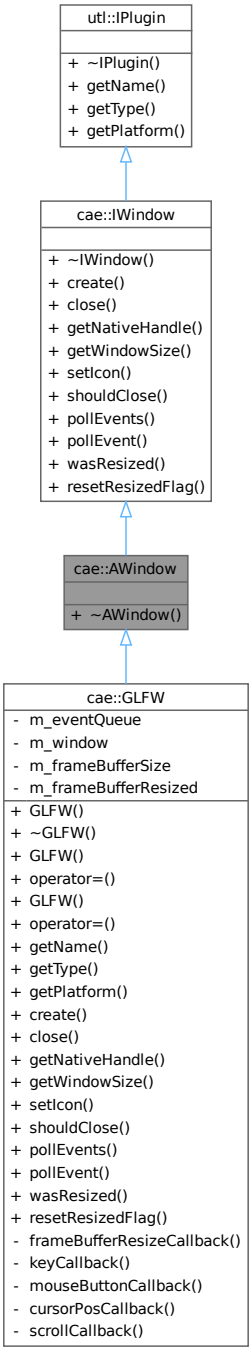
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/UI/AUI.hpp

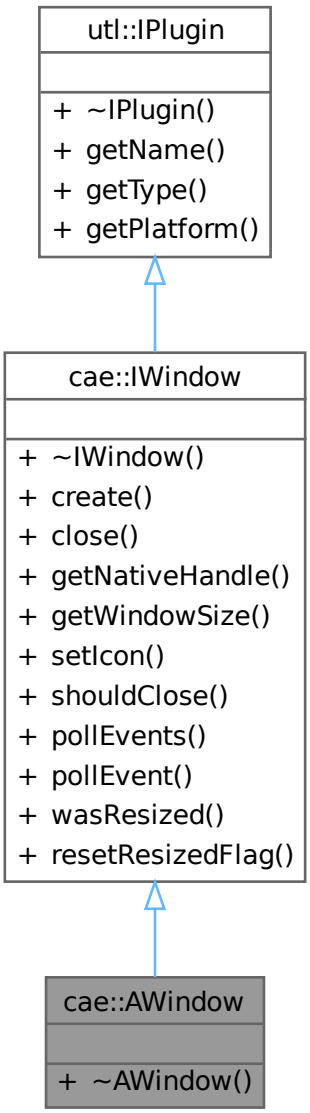
42.14 cae::AWindow Class Reference

#include <AWindow.hpp>

Inheritance diagram for cae::AWindow:



Collaboration diagram for cae::AWindow:



Public Member Functions

- ~AWindow () override=default

Public Member Functions inherited from cae::IWindow

- ~IWindow () override=default
- virtual bool create (const std::string &name, WindowSize size)=0

  Create a window with the given name and size.
- virtual void close ()=0

  Close the window.
- virtual NativeWindowHandle getNativeHandle () const =0

  Get the native window handle.

- virtual WindowSize getWindowSize () const =0

  Get the current window size.
- virtual void setIcon (const std::string &path) const =0

  Set the window icon from the given image path.
- virtual bool shouldClose () const =0

  Check if the window should close.
- virtual void pollEvents ()=0

  Poll window events.
- virtual bool pollEvent (WindowEvent &outEvent)=0

  Poll window events into outEvent.
- virtual bool wasResized () const =0

  Check if the window was resized.
- virtual void resetResizedFlag ()=0

  Reset the resized flag.

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.14.1 Detailed Description

Definition at line 19 of file AWindow.hpp.

42.14.2 Constructor & Destructor Documentation

42.14.2.1 ~AWindow()

cae::AWindow::~AWindow () [override], [default]

The documentation for this class was generated from the following file:

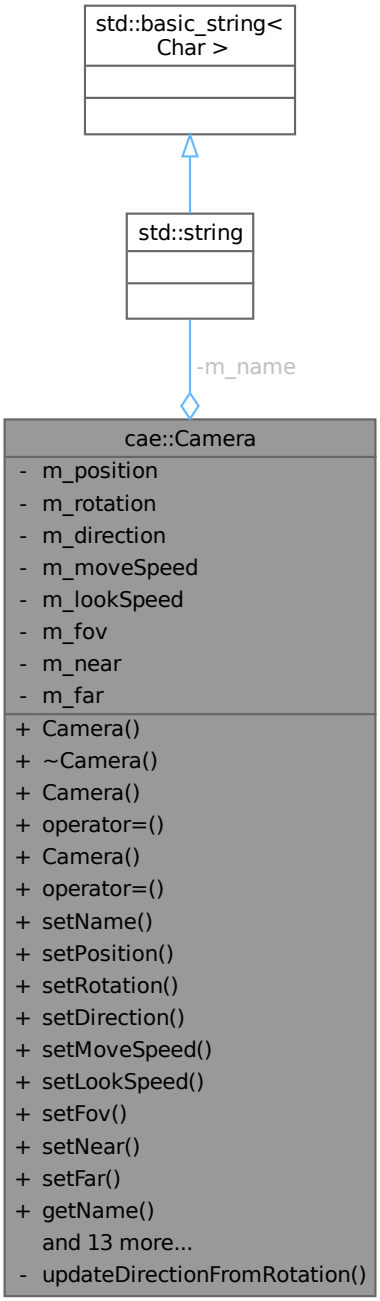
- modules/Interfaces/include/Interfaces/Window/AWindow.hpp

42.15 cae::Camera Class Reference

Class for camera.

#include <Camera.hpp>

Collaboration diagram for cae::Camera:



Public Member Functions

- Camera (const glm::vec3 position, const glm::vec3 rotation, const glm::vec3 direction, const float moveSpeed=CAMERA::MOVE_SPEED, const float lookSpeed=CAMERA::LOOK_SPEED, const float fov=CAMERA::FOV, const float nearPlane=CAMERA::NEAR_PLANE, const float farPlane=CAMERA::FAR_PLANE)
- ~Camera ()=default
- Camera (const Camera &)=delete
- Camera & operator= (const Camera &)=delete

- Camera (Camera &&)=delete
- Camera & operator= (Camera &&)=delete
- void setName (const std::string &name)
- void setPosition (const glm::vec3 &position)
- void setRotation (const glm::vec3 &rotation)
- void setDirection (const glm::vec3 &direction)
- void setMoveSpeed (const float speed)
- void setLookSpeed (const float speed)
- void setFov (const float fov)
- void setNear (const float nearPlane)
- void setFar (const float farPlane)
- const std::string & getName () const
- const glm::vec3 & getPosition () const
- const glm::vec3 & getRotation () const
- const glm::vec3 & getDirection () const
- const float & getMoveSpeed () const
- const float & getLookSpeed () const
- const float & getFov () const
- const float & getNear () const
- const float & getFar () const
- glm::mat4 getViewMatrix () const
- glm::mat4 getProjectionMatrix (const float aspectRatio) const
- glm::mat4 getViewProjection (const float aspectRatio) const
- void move (const glm::vec3 &direction, const float deltaTime)

  Move the camera in a given direction.
- void rotate (const float yawOffset, const float pitchOffset, const float deltaTime)

  Rotate the camera by given yaw and pitch offsets.

Private Member Functions

- void updateDirectionFromRotation ()

Private Attributes

- std::string m_name = CAMERA::NAME
- glm::vec3 m_position = glm::vec3(0.0F, 0.0F, 0.0F)
- glm::vec3 m_rotation = glm::vec3(0.0F, 0.0F, 0.0F)
- glm::vec3 m_direction = glm::vec3(0.0F, 0.0F, -1.0F)
- float m_moveSpeed = CAMERA::MOVE_SPEED
- float m_lookSpeed = CAMERA::LOOK_SPEED
- float m_fov = CAMERA::FOV
- float m_near = CAMERA::NEAR_PLANE
- float m_far = CAMERA::FAR_PLANE

42.15.1 Detailed Description

Class for camera.
Definition at line 23 of file Camera.hpp.

42.15.2 Constructor & Destructor Documentation

42.15.2.1 Camera() [1/3]

```
cae::Camera::Camera (
        const glm::vec3 position,
        const glm::vec3 rotation,
        const glm::vec3 direction,
        const float moveSpeed = CAMERA::MOVE_SPEED,
        const float lookSpeed = CAMERA::LOOK_SPEED,
        const float fov = CAMERA::FOV,
        const float nearPlane = CAMERA::NEAR_PLANE,
        const float farPlane = CAMERA::FAR_PLANE)   [inline]
```

Definition at line 26 of file Camera.hpp.

42.15.2.2 ~Camera()

```
cae::Camera::~Camera ()   [default]
```

42.15.2.3 Camera() [2/3]

```
cae::Camera::Camera (
        const Camera & )   [delete]
```

42.15.2.4 Camera() [3/3]

```
cae::Camera::Camera (
        Camera && )   [delete]
```

42.15.3 Member Function Documentation

42.15.3.1 getDirection()

```
const glm::vec3 & cae::Camera::getDirection () const   [inline], [nodiscard]
```

Definition at line 54 of file Camera.hpp.
References m_direction.

42.15.3.2 getFar()

```
const float & cae::Camera::getFar () const   [inline], [nodiscard]
```

Definition at line 59 of file Camera.hpp.
References m_far.

42.15.3.3 getFov()

```
const float & cae::Camera::getFov () const   [inline], [nodiscard]
```

Definition at line 57 of file Camera.hpp.
References m_fov.

42.15.3.4 getLookSpeed()

```
const float & cae::Camera::getLookSpeed () const   [inline], [nodiscard]
```

Definition at line 56 of file Camera.hpp.
References m_lookSpeed.

42.15.3.5 getMoveSpeed()

```
const float & cae::Camera::getMoveSpeed () const   [inline], [nodiscard]
```

Definition at line 55 of file Camera.hpp.
References m_moveSpeed.

42.15.3.6 getName()

const std::string & cae::Camera::getName () const [inline], [nodiscard]
Definition at line 51 of file Camera.hpp.
References m_name.

42.15.3.7 getNear()

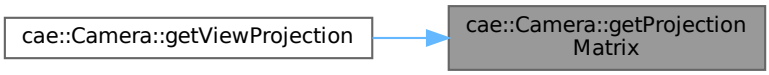
const float & cae::Camera::getNear () const [inline], [nodiscard]
Definition at line 58 of file Camera.hpp.
References m_near.

42.15.3.8 getPosition()

const glm::vec3 & cae::Camera::getPosition () const [inline], [nodiscard]
Definition at line 52 of file Camera.hpp.
References m_position.

42.15.3.9 getProjectionMatrix()

glm::mat4 cae::Camera::getProjectionMatrix (
const float aspectRatio) const [inline], [nodiscard]
Definition at line 65 of file Camera.hpp.
References m_far, m_fov, and m_near.
Referenced by getViewProjection().
Here is the caller graph for this function:



42.15.3.10 getRotation()

const glm::vec3 & cae::Camera::getRotation () const [inline], [nodiscard]
Definition at line 53 of file Camera.hpp.
References m_rotation.

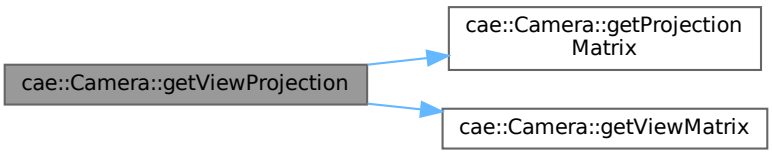
42.15.3.11 getViewMatrix()

glm::mat4 cae::Camera::getViewMatrix () const [inline], [nodiscard]
Definition at line 61 of file Camera.hpp.
References m_direction, and m_position.
Referenced by getViewProjection().
Here is the caller graph for this function:

42.15.3.12 getViewProjection()

glm::mat4 cae::Camera::getViewProjection (
const float aspectRatio) const [inline], [nodiscard]

Definition at line 69 of file Camera.hpp.
References getProjectionMatrix(), and getViewMatrix().
Here is the call graph for this function:



42.15.3.13 move()

void cae::Camera::move (
const glm::vec3 & direction,
const float deltaTime) [inline]

Move the camera in a given direction.

Parameters

direction	Direction to move the camera
deltaTime	Time delta for movement

Definition at line 79 of file Camera.hpp.
References m_moveSpeed, and m_position.

42.15.3.14 operator=() [1/2]

Camera & cae::Camera::operator= (
Camera &&) [delete]

42.15.3.15 operator=() [2/2]

Camera & cae::Camera::operator= (
const Camera &) [delete]

42.15.3.16 rotate()

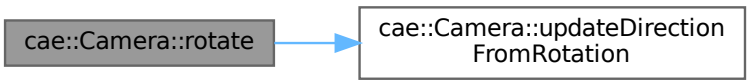
void cae::Camera::rotate (
const float yawOffset,
const float pitchOffset,
const float deltaTime) [inline]

Rotate the camera by given yaw and pitch offsets.

Parameters

yawOffset	Yaw offset to rotate the camera
pitchOffset	Pitch offset to rotate the camera
deltaTime	Time delta for rotation

Definition at line 90 of file Camera.hpp.
References m_lookSpeed, m_rotation, and updateDirectionFromRotation().
Here is the call graph for this function:



42.15.3.17 setDirection()

void cae::Camera::setDirection (
const glm::vec3 & direction) [inline]
Definition at line 44 of file Camera.hpp.
References m_direction.

42.15.3.18 setFar()

void cae::Camera::setFar (
const float farPlane) [inline]
Definition at line 49 of file Camera.hpp.
References m_far.

42.15.3.19 setFov()

void cae::Camera::setFov (
const float fov) [inline]
Definition at line 47 of file Camera.hpp.
References m_fov.

42.15.3.20 setLookSpeed()

void cae::Camera::setLookSpeed (
const float speed) [inline]
Definition at line 46 of file Camera.hpp.
References m_lookSpeed.

42.15.3.21 setMoveSpeed()

void cae::Camera::setMoveSpeed (
const float speed) [inline]
Definition at line 45 of file Camera.hpp.
References m_moveSpeed.

42.15.3.22 setName()

void cae::Camera::setName (
const std::string & name) [inline]
Definition at line 41 of file Camera.hpp.
References m_name.

42.15.3.23 setNear()

void cae::Camera::setNear (
const float nearPlane) [inline]
Definition at line 48 of file Camera.hpp.
References m_near.

42.15.3.24 setPosition()

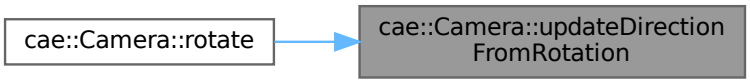
void cae::Camera::setPosition (
const glm::vec3 & position) [inline]
Definition at line 42 of file Camera.hpp.
References m_position.

42.15.3.25 setRotation()

void cae::Camera::setRotation (
const glm::vec3 & rotation) [inline]
Definition at line 43 of file Camera.hpp.
References m_rotation.

42.15.3.26 updateDirectionFromRotation()

void cae::Camera::updateDirectionFromRotation () [inline], [private]
Definition at line 115 of file Camera.hpp.
References m_direction, and m_rotation.
Referenced by rotate().
Here is the caller graph for this function:



42.15.4 Member Data Documentation

42.15.4.1 m_direction

glm::vec3 cae::Camera::m_direction = glm::vec3(0.0F, 0.0F, -1.0F) [private]
Definition at line 106 of file Camera.hpp.
Referenced by getDirection(), getViewMatrix(), setDirection(), and updateDirectionFromRotation().

42.15.4.2 m_far

float cae::Camera::m_far = CAMERA::FAR_PLANE [private]
Definition at line 113 of file Camera.hpp.
Referenced by getFar(), getProjectionMatrix(), and setFar().

42.15.4.3 m_fov

float cae::Camera::m_fov = CAMERA::FOV [private]
Definition at line 111 of file Camera.hpp.
Referenced by getFov(), getProjectionMatrix(), and setFov().

42.15.4.4 m_lookSpeed

float cae::Camera::m_lookSpeed = CAMERA::LOOK_SPEED [private]
Definition at line 109 of file Camera.hpp.
Referenced by getLookSpeed(), rotate(), and setLookSpeed().

42.15.4.5 m_moveSpeed

float cae::Camera::m_moveSpeed = CAMERA::MOVE_SPEED [private]
Definition at line 108 of file Camera.hpp.
Referenced by getMoveSpeed(), move(), and setMoveSpeed().

42.15.4.6 m_name

std::string cae::Camera::m_name = CAMERA::NAME [private]
Definition at line 102 of file Camera.hpp.
Referenced by getName(), and setName().

42.15.4.7 m_near

float cae::Camera::m_near = CAMERA::NEAR_PLANE [private]
Definition at line 112 of file Camera.hpp.
Referenced by getNear(), getProjectionMatrix(), and setNear().

42.15.4.8 m_position

glm::vec3 cae::Camera::m_position = glm::vec3(0.0F, 0.0F, 0.0F) [private]
Definition at line 104 of file Camera.hpp.
Referenced by getPosition(), getViewMatrix(), move(), and setPosition().

42.15.4.9 m_rotation

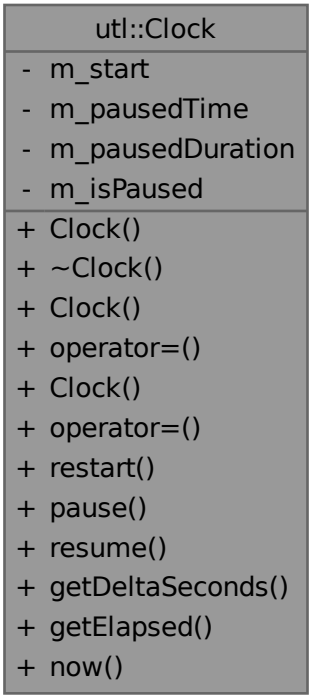
glm::vec3 cae::Camera::m_rotation = glm::vec3(0.0F, 0.0F, 0.0F) [private]
Definition at line 105 of file Camera.hpp.
Referenced by getRotation(), rotate(), setRotation(), and updateDirectionFromRotation().
The documentation for this class was generated from the following file:

- modules/Engine/include/Engine/Camera.hpp

42.16 utl::Clock Class Reference

Class for clock.
#include <Clock.hpp>

Collaboration diagram for utl::Clock:



Public Types

- using TimePoint = std::chrono::time_point<std::chrono::high_resolution_clock>

Public Member Functions

- Clock (const bool startNow=true)
- ~Clock ()=default
- Clock (const Clock &)=delete
- Clock & operator= (const Clock &)=delete
- Clock (Clock &&)=delete
- Clock & operator= (Clock &&)=delete
- void restart ()

  Restart the clock.
- void pause ()

  Pause the clock.
- void resume ()

  Resume the clock.
- float getDeltaSeconds () const

  Get the elapsed time in seconds.
- template<typename Duration = std::chrono::seconds>
  auto getElapsed () const

  Get the elapsed time in specified duration.

Static Public Member Functions

- static TimePoint now ()

  Get the current time point.

Private Types

- using Duration = std::chrono::high_resolution_clock::duration

Private Attributes

- TimePoint m_start
- TimePoint m_pausedTime
- Duration m_pausedDuration
- bool m_isPaused {false}

Friends

- std::ostream & operator<< (std::ostream &os, const Clock &clock)

42.16.1 Detailed Description

Class for clock.
Definition at line 19 of file Clock.hpp.

42.16.2 Member Typedef Documentation

42.16.2.1 Duration

using utl::Clock::Duration = std::chrono::high_resolution_clock::duration [private]
Definition at line 102 of file Clock.hpp.

42.16.2.2 TimePoint

using utl::Clock::TimePoint = std::chrono::time_point<std::chrono::high_resolution_clock>
Definition at line 23 of file Clock.hpp.

42.16.3 Constructor & Destructor Documentation

42.16.3.1 Clock() [1/3]

utl::Clock::Clock (
const bool startNow = true) [inline], [explicit]
Definition at line 25 of file Clock.hpp.

42.16.3.2 ~Clock()

utl::Clock::~Clock () [default]

42.16.3.3 Clock() [2/3]

utl::Clock::Clock (
const Clock &) [delete]

42.16.3.4 Clock() [3/3]

utl::Clock::Clock (
Clock &&) [delete]

42.16.4 Member Function Documentation

42.16.4.1 getDeltaSeconds()

float utl::Clock::getDeltaSeconds () const [inline], [nodiscard]
Get the elapsed time in seconds.

Returns

Elapsed time in seconds

Definition at line 82 of file Clock.hpp.
References m_isPaused, m_pausedDuration, m_pausedTime, m_start, and now().
Here is the call graph for this function:

42.16.4.2 getElapsed()

template<typename Duration = std::chrono::seconds>
auto utl::Clock::getElapsed () const [inline], [nodiscard]
Get the elapsed time in specified duration.

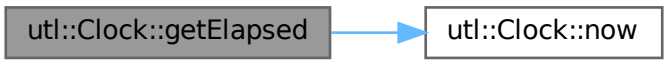
Template Parameters

Duration	Type of duration to return (default: seconds)

Returns

Elapsed time in specified duration

Definition at line 96 of file Clock.hpp.
References m_pausedDuration, m_start, and now().
Here is the call graph for this function:



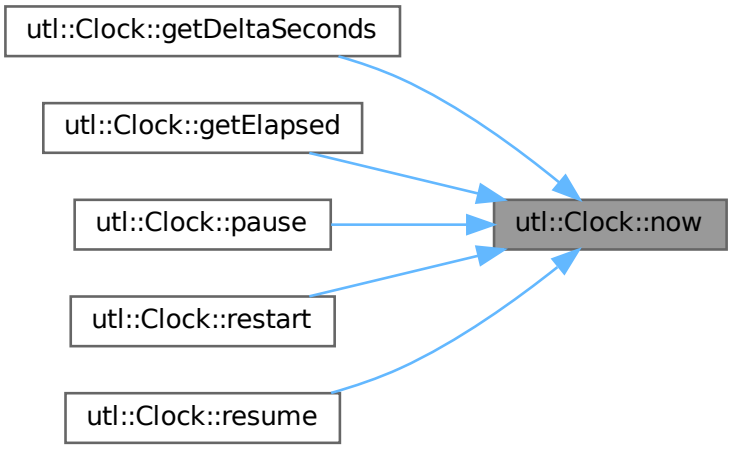
42.16.4.3 now()

static TimePoint utl::Clock::now () [inline], [static]
Get the current time point.

Returns

Current time point

Definition at line 43 of file Clock.hpp.
Referenced by getDeltaSeconds(), getElapsed(), pause(), restart(), and resume().
Here is the caller graph for this function:



42.16.4.4 operator=() [1/2]

Clock & utl::Clock::operator= (
Clock &&) [delete]

42.16.4.5 operator=() [2/2]

Clock & utl::Clock::operator= (
const Clock &) [delete]

42.16.4.6 pause()

void utl::Clock::pause () [inline]
Pause the clock.
Definition at line 57 of file Clock.hpp.
References m_isPaused, m_pausedTime, and now().
Here is the call graph for this function:

42.16.4.7 restart()

void utl::Clock::restart () [inline]
Restart the clock.
Definition at line 48 of file Clock.hpp.
References m_isPaused, m_pausedDuration, m_start, and now().
Here is the call graph for this function:

42.16.4.8 resume()

void utl::Clock::resume () [inline]
Resume the clock.
Definition at line 69 of file Clock.hpp.
References m_isPaused, m_pausedDuration, m_pausedTime, and now().
Here is the call graph for this function:

42.16.5 Friends And Related Symbol Documentation

42.16.5.1 operator<<

std::ostream & operator<< (
std::ostream & os,
const Clock & clock) [friend]
Definition at line 33 of file Clock.hpp.

42.16.6 Member Data Documentation

42.16.6.1 m_isPaused

bool utl::Clock::m_isPaused {false} [private]
Definition at line 107 of file Clock.hpp.
Referenced by getDeltaSeconds(), pause(), restart(), and resume().

42.16.6.2 m_pausedDuration

Duration utl::Clock::m_pausedDuration [private]
Definition at line 106 of file Clock.hpp.
Referenced by getDeltaSeconds(), getElapsed(), restart(), and resume().

42.16.6.3 m_pausedTime

TimePoint utl::Clock::m_pausedTime [private]
Definition at line 105 of file Clock.hpp.
Referenced by getDeltaSeconds(), pause(), and resume().

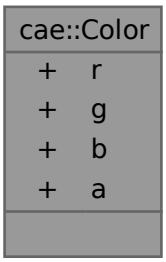
42.16.6.4 m_start

TimePoint utl::Clock::m_start [private]
Definition at line 104 of file Clock.hpp.
Referenced by getDeltaSeconds(), getElapsed(), and restart().
The documentation for this class was generated from the following file:

- modules/Utils/include/Utils/Clock.hpp

42.17 cae::Color Struct Reference

Struct for color.
#include <IRenderer.hpp>
Collaboration diagram for cae::Color:



Public Attributes

- float r
- float g
- float b
- float a

42.17.1 Detailed Description

Struct for color.
Definition at line 22 of file IRenderer.hpp.

42.17.2 Member Data Documentation

42.17.2.1 a

float cae::Color::a
Definition at line 27 of file IRenderer.hpp.
Referenced by cae::Engine::Engine(), cae::OPGL::initialize(), cae::Application::parseEngineConf(), and cae::OPGL::setClearColor().

42.17.2.2 b

float cae::Color::b
Definition at line 26 of file IRenderer.hpp.
Referenced by cae::Engine::Engine(), cae::OPGL::initialize(), cae::Application::parseEngineConf(), and cae::OPGL::setClearColor().

42.17.2.3 g

float cae::Color::g
Definition at line 25 of file IRenderer.hpp.
Referenced by cae::Engine::Engine(), cae::OPGL::initialize(), cae::Application::parseEngineConf(), and cae::OPGL::setClearColor().

42.17.2.4 r

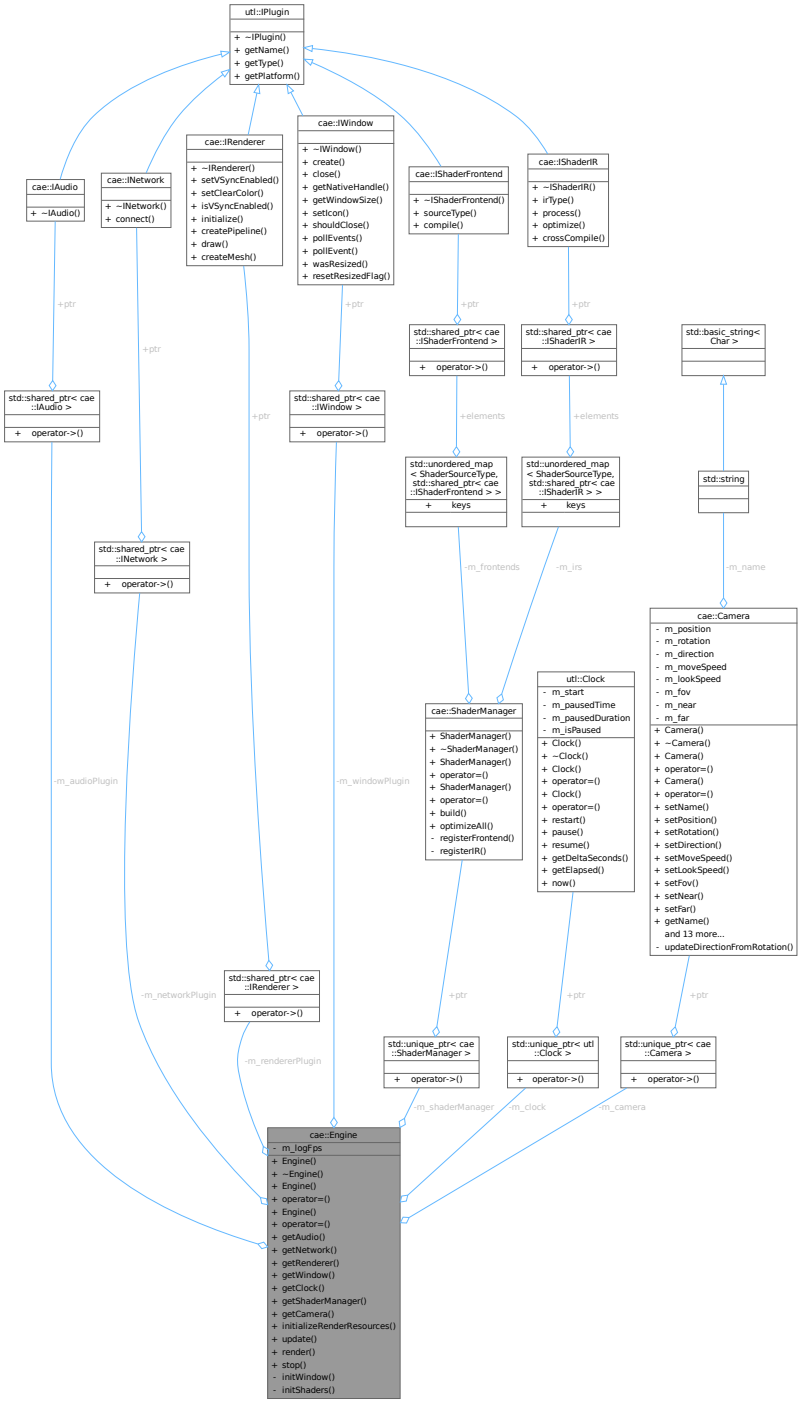
float cae::Color::r
Definition at line 24 of file IRenderer.hpp.
Referenced by cae::Engine::Engine(), cae::IRenderer::initialize(), cae::OPGL::initialize(), cae::Application::parseEngineCon
and cae::OPGL::setClearColor().
The documentation for this struct was generated from the following file:

- modules/Interfaces/include/Interfaces/Renderer/IRenderer.hpp

42.18 cae::Engine Class Reference

Engine class.
#include <Engine.hpp>

Collaboration diagram for cae::Engine:



Public Member Functions

- Engine (const EngineConfig &config, const std::function< std::shared_ptr< IAudio >()> &audio↩ Factory, const std::function< std::shared_ptr< INetwork >()> &networkFactory, const std↩ ::function< std::shared_ptr< IRenderer >()> &rendererFactory, const std::function< std::shared↩ _ptr< IShaderIR >()> &shaderIRFactory, const std::vector< std::function< std::shared_ptr< IShaderFrontend >()> > &shaderFrontendFactories, const std::function< std::shared_ptr< IWindow >()> &windowFactory)
- ~Engine ()=default

- Engine (const Engine &)=delete
- Engine & operator= (const Engine &)=delete
- Engine (Engine &&)=delete
- Engine & operator= (Engine &&)=delete
- const std::shared_ptr< IAudio > & getAudio () const
- const std::shared_ptr< INetwork > & getNetwork () const
- const std::shared_ptr< IRenderer > & getRenderer () const
- const std::shared_ptr< IWindow > & getWindow () const
- const std::unique_ptr< utl::Clock > & getClock ()
- const std::unique_ptr< ShaderManager > & getShaderManager () const
- const std::unique_ptr< Camera > & getCamera () const
- void initializeRenderResources (const std::vector< ShaderSourceDesc > &shaderSources, const std::vector< float > &vertices) const

  Initialize render resources.
- void update (std::array< float, 10 > &fpsBuffer, int &fpsIndex)
- void render ()
- void stop ()

  Stop the engine.

Private Member Functions

- void initWindow (const std::string &windowName, const WindowSize &windowSize, const std::string &iconPath) const

  Initialize the window.
- void initShaders (const std::vector< ShaderSourceDesc > &shaderSources) const

  Initialize shaders.

Private Attributes

- std::shared_ptr< IAudio > m_audioPlugin = nullptr
- std::shared_ptr< INetwork > m_networkPlugin = nullptr
- std::shared_ptr< IRenderer > m_rendererPlugin = nullptr
- std::shared_ptr< IWindow > m_windowPlugin = nullptr
- std::unique_ptr< utl::Clock > m_clock = nullptr
- std::unique_ptr< ShaderManager > m_shaderManager = nullptr
- std::unique_ptr< Camera > m_camera = nullptr
- bool m_logFps = false

42.18.1 Detailed Description

Engine class.
Definition at line 65 of file Engine.hpp.

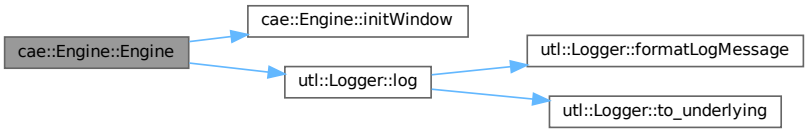
42.18.2 Constructor & Destructor Documentation

42.18.2.1 Engine() [1/3]

```
cae::Engine::Engine (
        const EngineConfig & config,
        const std::function< std::shared_ptr< IAudio >()> & audioFactory,
        const std::function< std::shared_ptr< INetwork >()> & networkFactory,
        const std::function< std::shared_ptr< IRenderer >()> & rendererFactory,
        const std::function< std::shared_ptr< IShaderIR >()> & shaderIRFactory,
        const std::vector< std::function< std::shared_ptr< IShaderFrontend >()> > & shaderFrontendFactories,
        const std::function< std::shared_ptr< IWindow >()> & windowFactory)
```

Definition at line 18 of file engine.cpp.

References cae::Color::a, cae::EngineConfig::audio_master_volume, cae::EngineConfig::audio_muted, cae::Color::b, cae::Color::g, utl::INFO, initWindow(), utl::Logger::log(), cae::EngineConfig::log_fps, m_rendererPlugin, m_windowPlugin, cae::EngineConfig::network_host, cae::EngineConfig::network_port, cae::Color::r, cae::EngineConfig::renderer_clear_color, cae::EngineConfig::renderer_frame_rate_limit, cae::EngineConfig::renderer_vsync, cae::EngineConfig::window_fullscreen, cae::EngineConfig::window_height, cae::EngineConfig::window_icon_path, cae::EngineConfig::window_name, and cae::EngineConfig::window_width.
Here is the call graph for this function:



42.18.2.2 ~Engine()

cae::Engine::~Engine () [default]

42.18.2.3 Engine() [2/3]

cae::Engine::Engine (
const Engine &) [delete]

42.18.2.4 Engine() [3/3]

cae::Engine::Engine (
Engine &&) [delete]

42.18.3 Member Function Documentation

42.18.3.1 getAudio()

const std::shared_ptr< IAudio > & cae::Engine::getAudio () const [inline], [nodiscard]
Definition at line 82 of file Engine.hpp.
References m_audioPlugin.

42.18.3.2 getCamera()

const std::unique_ptr< Camera > & cae::Engine::getCamera () const [inline], [nodiscard]
Definition at line 89 of file Engine.hpp.
References m_camera.

42.18.3.3 getClock()

const std::unique_ptr< utl::Clock > & cae::Engine::getClock () [inline], [nodiscard]
Definition at line 87 of file Engine.hpp.
References m_clock.

42.18.3.4 getNetwork()

const std::shared_ptr< INetwork > & cae::Engine::getNetwork () const [inline], [nodiscard]
Definition at line 83 of file Engine.hpp.
References m_networkPlugin.

42.18.3.5 getRenderer()

const std::shared_ptr< IRenderer > & cae::Engine::getRenderer () const [inline], [nodiscard]
Definition at line 84 of file Engine.hpp.
References m_rendererPlugin.

42.18.3.6 getShaderManager()

const std::unique_ptr< ShaderManager > & cae::Engine::getShaderManager () const [inline], [nodiscard]
Definition at line 88 of file Engine.hpp.
References m_shaderManager.

42.18.3.7 getWindow()

const std::shared_ptr< IWindow > & cae::Engine::getWindow () const [inline], [nodiscard]
Definition at line 85 of file Engine.hpp.
References m_windowPlugin.

42.18.3.8 initializeRenderResources()

void cae::Engine::initializeRenderResources (
const std::vector< ShaderSourceDesc > & shaderSources,
const std::vector< float > & vertices) const
Initialize render resources.

Parameters

shaderSources	Shader sources to initialize
vertices	Vertex data to initialize

Definition at line 55 of file engine.cpp.

42.18.3.9 initShaders()

void cae::Engine::initShaders (
const std::vector< ShaderSourceDesc > & shaderSources) const [private]
Initialize shaders.

Parameters

shaderSources	Shader sources to initialize

Definition at line 106 of file engine.cpp.
References cae::SPIRV.

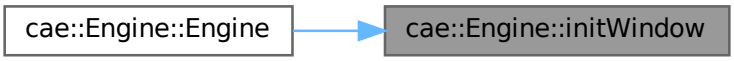
42.18.3.10 initWindow()

void cae::Engine::initWindow (
const std::string & windowName,
const WindowSize & windowSize,
const std::string & iconPath) const [private]
Initialize the window.

Parameters

windowName	window name
windowSize	window size
iconPath	path to window icon

Definition at line 96 of file engine.cpp.
Referenced by Engine().
Here is the caller graph for this function:



42.18.3.11 operator=() [1/2]

Engine & cae::Engine::operator= (
const Engine &) [delete]

42.18.3.12 operator=() [2/2]

Engine & cae::Engine::operator= (
Engine &&) [delete]

42.18.3.13 render()

void cae::Engine::render ()
Definition at line 62 of file engine.cpp.

42.18.3.14 stop()

void cae::Engine::stop ()
Stop the engine.
Definition at line 81 of file engine.cpp.
References utl::INFO, and utl::Logger::log().
Here is the call graph for this function:

42.18.3.15 update()

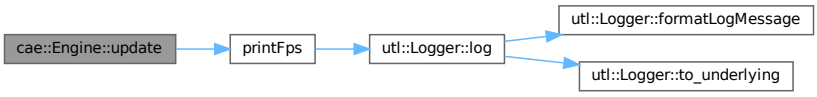
void cae::Engine::update (
std::array< float, 10 > & fpsBuffer,
int & fpsIndex)

Parameters

fpsBuffer	

fpsIndex	

Definition at line 72 of file engine.cpp.
References printFps().
Here is the call graph for this function:



42.18.4 Member Data Documentation

42.18.4.1 m_audioPlugin

std::shared_ptr<IAudio> cae::Engine::m_audioPlugin = nullptr [private]
Definition at line 117 of file Engine.hpp.
Referenced by getAudio().

42.18.4.2 m_camera

std::unique_ptr<Camera> cae::Engine::m_camera = nullptr [private]
Definition at line 124 of file Engine.hpp.
Referenced by getCamera().

42.18.4.3 m_clock

std::unique_ptr<utl::Clock> cae::Engine::m_clock = nullptr [private]
Definition at line 122 of file Engine.hpp.
Referenced by getClock().

42.18.4.4 m_logFps

bool cae::Engine::m_logFps = false [private]
Definition at line 126 of file Engine.hpp.

42.18.4.5 m_networkPlugin

std::shared_ptr<INetwork> cae::Engine::m_networkPlugin = nullptr [private]
Definition at line 118 of file Engine.hpp.
Referenced by getNetwork().

42.18.4.6 m_rendererPlugin

std::shared_ptr<IRenderer> cae::Engine::m_rendererPlugin = nullptr [private]
Definition at line 119 of file Engine.hpp.
Referenced by Engine(), and getRenderer().

42.18.4.7 m_shaderManager

std::unique_ptr<ShaderManager> cae::Engine::m_shaderManager = nullptr [private]
Definition at line 123 of file Engine.hpp.
Referenced by getShaderManager().

42.18.4.8 m_windowPlugin

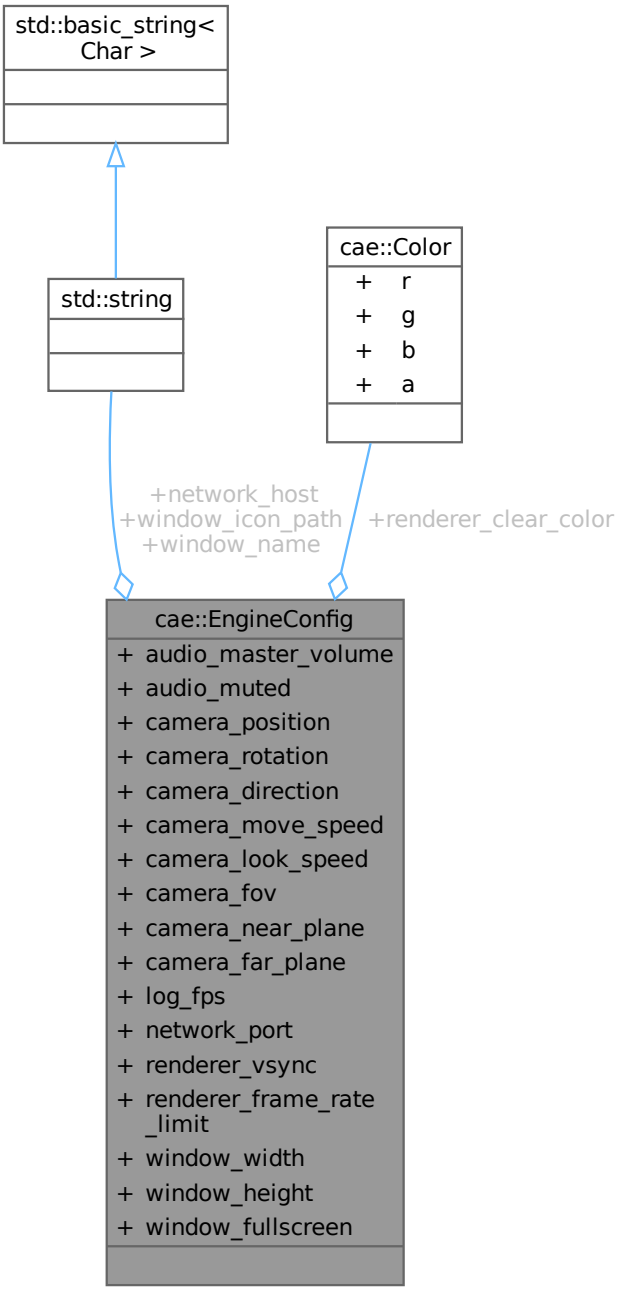
std::shared_ptr<IWindow> cae::Engine::m_windowPlugin = nullptr [private]
Definition at line 120 of file Engine.hpp.
Referenced by Engine(), and getWindow().
The documentation for this class was generated from the following files:

- modules/Engine/include/Engine/Engine.hpp
- modules/Engine/src/engine.cpp

42.19 cae::EngineConfig Struct Reference

Struct for engine configuration.
#include <Engine.hpp>

Collaboration diagram for cae::EngineConfig:



Public Attributes

- float audio_master_volume = AUDIO::VOLUME
- bool audio_muted = AUDIO::MUTED
- glm::vec3 camera_position = glm::vec3(0.0F, 0.0F, 0.0F)
- glm::vec3 camera_rotation = glm::vec3(0.0F, 0.0F, 0.0F)
- glm::vec3 camera_direction = glm::vec3(0.0F, 0.0F, -1.0F)
- float camera_move_speed = CAMERA::MOVE_SPEED

- float camera_look_speed = CAMERA::LOOK_SPEED
- float camera_fov = CAMERA::FOV
- float camera_near_plane = CAMERA::NEAR_PLANE
- float camera_far_plane = CAMERA::FAR_PLANE
- bool log_fps = LOG::LOG_FPS
- std::string network_host = NETWORK::HOST
- uint16_t network_port = NETWORK::PORT
- bool renderer_vsync = RENDERER::VSYNC
- uint16_t renderer_frame_rate_limit = RENDERER::FRAME_RATE_LIMIT
- Color renderer_clear_color
- uint16_t window_width = WINDOW::WIDTH
- uint16_t window_height = WINDOW::HEIGHT
- bool window_fullscreen = WINDOW::FULLSCREEN
- std::string window_name = WINDOW::NAME
- std::string window_icon_path = WINDOW::ICON_PATH

42.19.1 Detailed Description

Struct for engine configuration.
Definition at line 27 of file Engine.hpp.

42.19.2 Member Data Documentation

42.19.2.1 audio_master_volume

float cae::EngineConfig::audio_master_volume = AUDIO::VOLUME
Definition at line 29 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.2 audio_muted

bool cae::EngineConfig::audio_muted = AUDIO::MUTED
Definition at line 30 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.3 camera_direction

glm::vec3 cae::EngineConfig::camera_direction = glm::vec3(0.0F, 0.0F, -1.0F)
Definition at line 34 of file Engine.hpp.
Referenced by cae::Application::parseEngineConf().

42.19.2.4 camera_far_plane

float cae::EngineConfig::camera_far_plane = CAMERA::FAR_PLANE
Definition at line 39 of file Engine.hpp.
Referenced by cae::Application::parseEngineConf().

42.19.2.5 camera_fov

float cae::EngineConfig::camera_fov = CAMERA::FOV
Definition at line 37 of file Engine.hpp.
Referenced by cae::Application::parseEngineConf().

42.19.2.6 camera_look_speed

float cae::EngineConfig::camera_look_speed = CAMERA::LOOK_SPEED
Definition at line 36 of file Engine.hpp.
Referenced by cae::Application::parseEngineConf().

42.19.2.7 camera_move_speed

float cae::EngineConfig::camera_move_speed = CAMERA::MOVE_SPEED
Definition at line 35 of file Engine.hpp.
Referenced by cae::Application::parseEngineConf().

42.19.2.8 camera_near_plane

float cae::EngineConfig::camera_near_plane = CAMERA::NEAR_PLANE
Definition at line 38 of file Engine.hpp.
Referenced by cae::Application::parseEngineConf().

42.19.2.9 camera_position

glm::vec3 cae::EngineConfig::camera_position = glm::vec3(0.0F, 0.0F, 0.0F)
Definition at line 32 of file Engine.hpp.
Referenced by cae::Application::parseEngineConf().

42.19.2.10 camera_rotation

glm::vec3 cae::EngineConfig::camera_rotation = glm::vec3(0.0F, 0.0F, 0.0F)
Definition at line 33 of file Engine.hpp.
Referenced by cae::Application::parseEngineConf().

42.19.2.11 log_fps

bool cae::EngineConfig::log_fps = LOG::LOG_FPS
Definition at line 41 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.12 network_host

std::string cae::EngineConfig::network_host = NETWORK::HOST
Definition at line 43 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.13 network_port

uint16_t cae::EngineConfig::network_port = NETWORK::PORT
Definition at line 44 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.14 renderer_clear_color

Color cae::EngineConfig::renderer_clear_color
Initial value:
= {.r = RENDERER::CLEAR_COLOR_R,
.g = RENDERER::CLEAR_COLOR_G,
.b = RENDERER::CLEAR_COLOR_B,
.a = RENDERER::CLEAR_COLOR_A}
Definition at line 48 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.15 renderer_frame_rate_limit

uint16_t cae::EngineConfig::renderer_frame_rate_limit = RENDERER::FRAME_RATE_LIMIT
Definition at line 47 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.16 renderer_vsync

bool cae::EngineConfig::renderer_vsync = RENDERER::VSYNC
Definition at line 46 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.17 window_fullscreen

bool cae::EngineConfig::window_fullscreen = WINDOW::FULLSCREEN
Definition at line 55 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.18 window_height

uint16_t cae::EngineConfig::window_height = WINDOW::HEIGHT
Definition at line 54 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.19 window_icon_path

std::string cae::EngineConfig::window_icon_path = WINDOW::ICON_PATH
Definition at line 57 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.20 window_name

std::string cae::EngineConfig::window_name = WINDOW::NAME
Definition at line 56 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

42.19.2.21 window_width

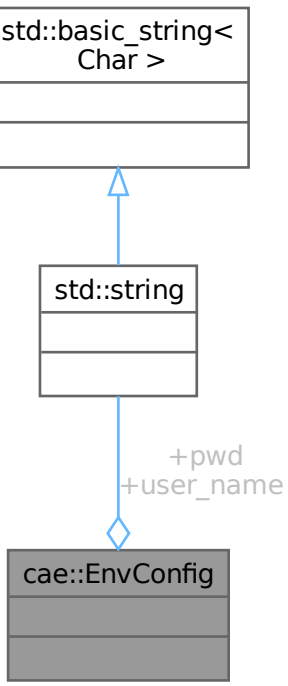
uint16_t cae::EngineConfig::window_width = WINDOW::WIDTH
Definition at line 53 of file Engine.hpp.
Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().
The documentation for this struct was generated from the following file:

- modules/Engine/include/Engine/Engine.hpp

42.20 cae::EnvConfig Struct Reference

Struct for environment variables configuration.
#include <ArgsHandler.hpp>

Collaboration diagram for cae::EnvConfig:



Public Attributes

- std::string user_name
- std::string pwd

42.20.1 Detailed Description

Struct for environment variables configuration.
Definition at line 30 of file ArgsHandler.hpp.

42.20.2 Member Data Documentation

42.20.2.1 pwd

std::string cae::EnvConfig::pwd
Definition at line 33 of file ArgsHandler.hpp.
Referenced by cae::ArgsHandler::ParseEnv().

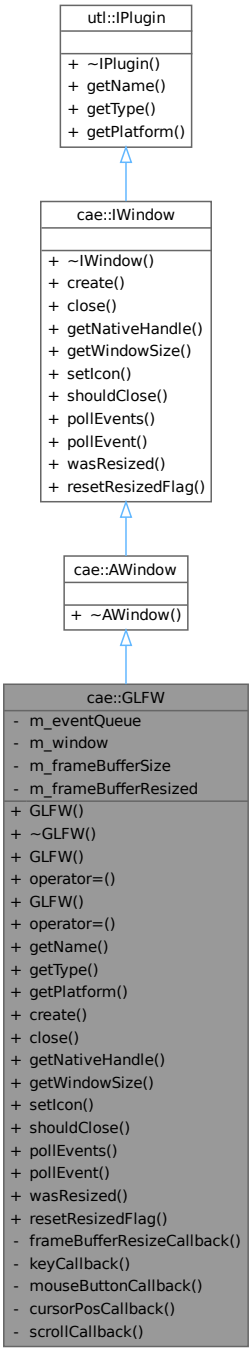
42.20.2.2 user_name

std::string cae::EnvConfig::user_name
Definition at line 32 of file ArgsHandler.hpp.
Referenced by cae::ArgsHandler::ParseEnv().
The documentation for this struct was generated from the following file:

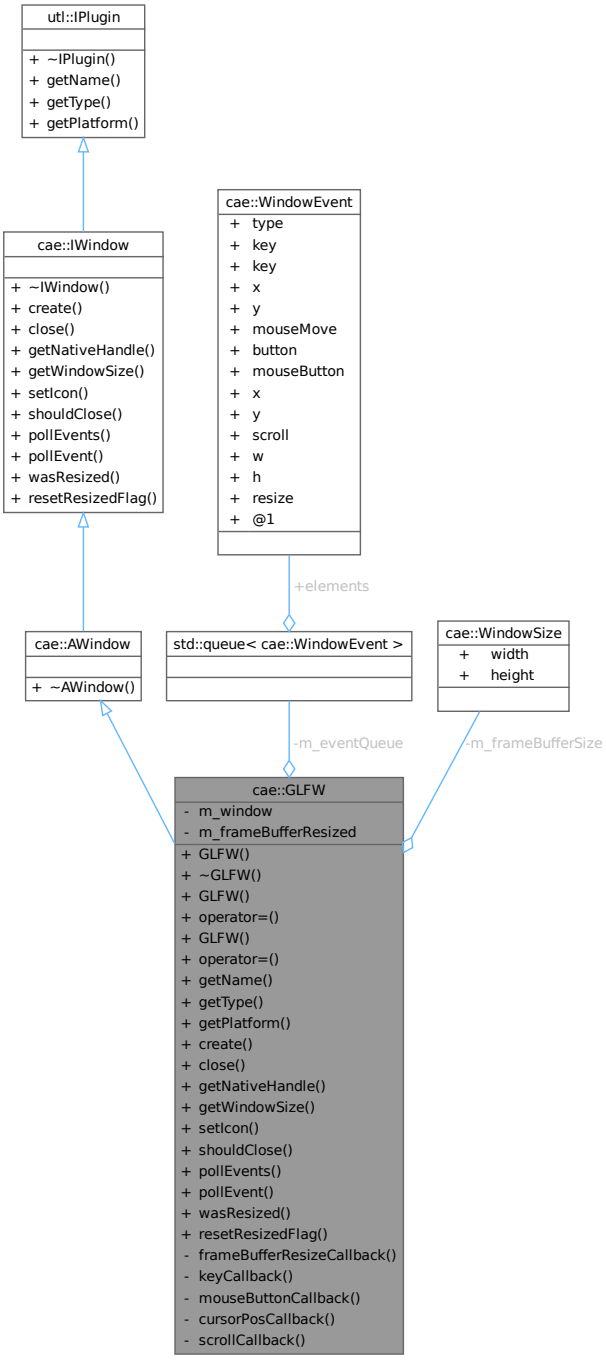
- include/CAE/ArgsHandler.hpp

42.21 cae::GLFW Class Reference

Class for the GLFW plugin.
#include <GLFW.hpp>
Inheritance diagram for cae::GLFW:



Collaboration diagram for cae::GLFW:



Public Member Functions

- GLFW ()=default
- ~GLFW () override=default
- GLFW (const GLFW &)=delete
- GLFW & operator= (const GLFW &)=delete
- GLFW (GLFW &&)=delete
- GLFW & operator= (GLFW &&)=delete

- std::string getName () const override

  Get the name of the plugin.
- utl::PluginType getType () const override

  Get the type of the plugin.
- utl::PluginPlatform getPlatform () const override

  Get the handled platform of the plugin.
- bool create (const std::string &name, WindowSize size) override

  Create a window with the given name and size.
- void close () override

  Close the window.
- NativeWindowHandle getNativeHandle () const override

  Get the native window handle.
- WindowSize getWindowSize () const override

  Get the current window size.
- void setIcon (const std::string &path) const override

  Set the window icon from the given image path.
- bool shouldClose () const override

  Check if the window should close.
- void pollEvents () override

  Poll window events.
- bool pollEvent (WindowEvent &event) override

  Poll window events into outEvent.
- bool wasResized () const override

  Check if the window was resized.
- void resetResizedFlag () override

  Reset the resized flag.

Public Member Functions inherited from cae::AWindow

- ~AWindow () override=default

Public Member Functions inherited from cae::IWindow

- ~IWindow () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default

Static Private Member Functions

- static void frameBufferResizeCallback (GLFWwindow *window, int width, int height)
- static void keyCallback (GLFWwindow *window, int key, int scancode, int action, int mods)
- static void mouseButtonCallback (GLFWwindow *window, int button, int action, int mods)
- static void cursorPosCallback (GLFWwindow *window, double x, double y)
- static void scrollCallback (GLFWwindow *window, double xoffset, double yoffset)

Private Attributes

- std::queue< WindowEvent > m_eventQueue
- GLFWwindow * m_window = nullptr
- WindowSize m_frameBufferSize {}
- bool m_frameBufferResized = false

42.21.1 Detailed Description

Class for the GLFW plugin.
Definition at line 30 of file GLFW.hpp.

42.21.2 Constructor & Destructor Documentation

42.21.2.1 GLFW() [1/3]

cae::GLFW::GLFW () [default]

42.21.2.2 ~GLFW()

cae::GLFW::~GLFW () [override], [default]

42.21.2.3 GLFW() [2/3]

cae::GLFW::GLFW (
const GLFW &) [delete]

42.21.2.4 GLFW() [3/3]

cae::GLFW::GLFW (
GLFW &&) [delete]

42.21.3 Member Function Documentation

42.21.3.1 close()

void cae::GLFW::close () [override], [virtual]
Close the window.
Implements cae::IWindow.
Definition at line 302 of file glfw.cpp.

42.21.3.2 create()

bool cae::GLFW::create (
const std::string & name,
WindowSize size) [override], [virtual]
Create a window with the given name and size.

Parameters

name	Window name
size	Window size

Returns

True if the window was created successfully

Implements cae::IWindow.
Definition at line 268 of file glfw.cpp.
References cae::WindowSize::height, utl::Logger::log(), utl::WARNING, and cae::WindowSize::width.
Here is the call graph for this function:

42.21.3.3 cursorPosCallback()

void cae::GLFW::cursorPosCallback (
GLFWwindow * window,
double x,
double y) [static], [private]
Definition at line 217 of file glfw.cpp.
References cae::MouseMove, and cae::WindowEvent::type.

42.21.3.4 frameBufferResizeCallback()

void cae::GLFW::frameBufferResizeCallback (
GLFWwindow * window,
int width,
int height) [static], [private]
Definition at line 249 of file glfw.cpp.
References m_frameBufferResized, cae::Resize, and cae::WindowEvent::type.

42.21.3.5 getName()

std::string cae::GLFW::getName () const [inline], [nodiscard], [override], [virtual]
Get the name of the plugin.

Returns

The name of the plugin

Implements utl::IPlugin.
Definition at line 42 of file GLFW.hpp.

42.21.3.6 getNativeHandle()

cae::NativeWindowHandle cae::GLFW::getNativeHandle () const [nodiscard], [override], [virtual]
Get the native window handle.

Returns

Native window handle

Implements cae::IWindow.
Definition at line 320 of file glfw.cpp.
References cae::NativeWindowHandle::window.

42.21.3.7 getPlatform()

utl::PluginPlatform cae::GLFW::getPlatform () const [inline], [nodiscard], [override], [virtual]
Get the handled platform of the plugin.

Returns

The handled platform of the plugin

Implements utl::IPlugin.
Definition at line 44 of file GLFW.hpp.
References utl::ALL.

42.21.3.8 getType()

utl::PluginType cae::GLFW::getType () const [inline], [nodiscard], [override], [virtual]
Get the type of the plugin.

Returns

The type of the plugin

Implements utl::IPlugin.
Definition at line 43 of file GLFW.hpp.
References utl::WINDOW.

42.21.3.9 getWindowSize()

cae::WindowSize cae::GLFW::getWindowSize () const [nodiscard], [override], [virtual]
Get the current window size.

Returns

Current window size

Implements cae::IWindow.
Definition at line 312 of file glfw.cpp.

42.21.3.10 keyCallback()

void cae::GLFW::keyCallback (
GLFWwindow * window,
int key,
int scancode,
int action,
int mods) [static], [private]
Definition at line 176 of file glfw.cpp.
References cae::KeyDown, cae::KeyUp, translateKey(), and cae::WindowEvent::type.
Here is the call graph for this function:

42.21.3.11 mouseButtonCallback()

void cae::GLFW::mouseButtonCallback (
GLFWwindow * window,
int button,
int action,
int mods) [static], [private]

Definition at line 202 of file glfw.cpp.
References cae::MouseButtonDown, cae::MouseButtonUp, and cae::WindowEvent::type.

42.21.3.12 operator=() [1/2]

GLFW & cae::GLFW::operator= (
const GLFW &) [delete]

42.21.3.13 operator=() [2/2]

GLFW & cae::GLFW::operator= (
GLFW &&) [delete]

42.21.3.14 pollEvent()

bool cae::GLFW::pollEvent (
WindowEvent & outEvent) [override], [virtual]

Poll window events into outEvent.

Parameters

outEvent	Event to be filled

Returns

True if an event was polled

Implements cae::IWindow.
Definition at line 348 of file glfw.cpp.

42.21.3.15 pollEvents()

void cae::GLFW::pollEvents () [inline], [override], [virtual]

Poll window events.
Implements cae::IWindow.
Definition at line 55 of file GLFW.hpp.

42.21.3.16 resetResizedFlag()

void cae::GLFW::resetResizedFlag () [inline], [override], [virtual]

Reset the resized flag.
Implements cae::IWindow.
Definition at line 59 of file GLFW.hpp.
References m_frameBufferResized.

42.21.3.17 scrollCallback()

void cae::GLFW::scrollCallback (
GLFWwindow * window,
double xoffset,
double yoffset) [static], [private]

Definition at line 233 of file glfw.cpp.
References cae::MouseScroll, and cae::WindowEvent::type.

42.21.3.18 setIcon()

void cae::GLFW::setIcon (
const std::string & path) const [override], [virtual]
Set the window icon from the given image path.

Parameters

path	Path to the icon image

Returns

True if the icon was set successfully

Implements cae::IWindow.
Definition at line 336 of file glfw.cpp.
References utl::Image::height, utl::Logger::log(), utl::Image::pixels, utl::WARNING, and utl::Image::width.
Here is the call graph for this function:

42.21.3.19 shouldClose()

bool cae::GLFW::shouldClose () const [inline], [nodiscard], [override], [virtual]
Check if the window should close.

Returns

True if the window should close

Implements cae::IWindow.
Definition at line 54 of file GLFW.hpp.
References m_window.

42.21.3.20 wasResized()

bool cae::GLFW::wasResized () const [inline], [nodiscard], [override], [virtual]
Check if the window was resized.

Returns

True if the window was resized

Implements cae::IWindow.
Definition at line 58 of file GLFW.hpp.
References m_frameBufferResized.

42.21.4 Member Data Documentation

42.21.4.1 m_eventQueue

std::queue<WindowEvent> cae::GLFW::m_eventQueue [private]
Definition at line 68 of file GLFW.hpp.

42.21.4.2 m_frameBufferResized

bool cae::GLFW::m_frameBufferResized = false [private]
Definition at line 72 of file GLFW.hpp.
Referenced by frameBufferResizeCallback(), resetResizedFlag(), and wasResized().

42.21.4.3 m_frameBufferSize

WindowSize cae::GLFW::m_frameBufferSize {} [private]
Definition at line 71 of file GLFW.hpp.

42.21.4.4 m_window

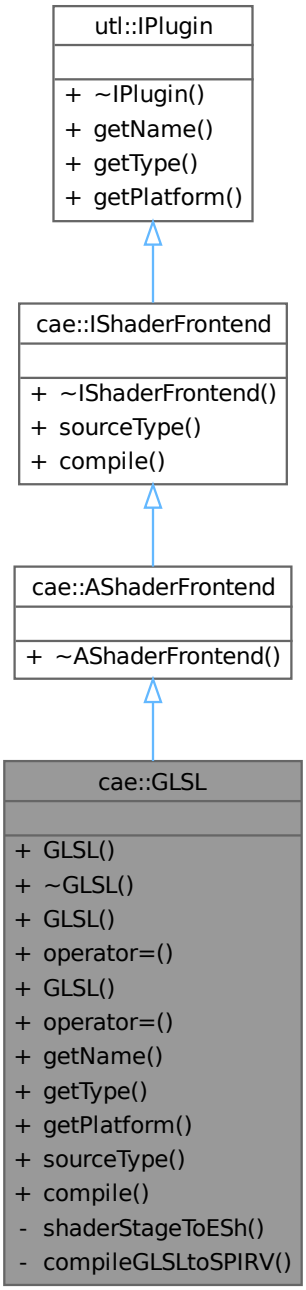
GLFWwindow* cae::GLFW::m_window = nullptr [private]
Definition at line 70 of file GLFW.hpp.
Referenced by shouldClose().
The documentation for this class was generated from the following files:

- plugins/Window/GLFW/include/GLFW/GLFW.hpp
- plugins/Window/GLFW/src/glfw.cpp

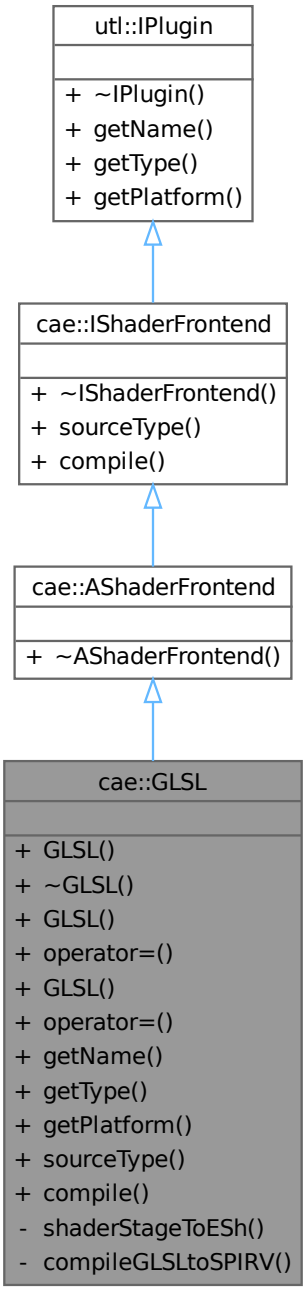
42.22 cae::GLSL Class Reference

Class for the GLSL plugin.
#include <GLSL.hpp>

Inheritance diagram for cae::GLSL:



Collaboration diagram for cae::GLSL:



Public Member Functions

- GLSL ()=default
- ~GLSL () override=default
- GLSL (const GLSL &)=delete
- GLSL & operator= (const GLSL &)=delete
- GLSL (GLSL &&)=delete
- GLSL & operator= (GLSL &&)=delete

- std::string getName () const override
  Get the name of the plugin.
- utl::PluginType getType () const override
  Get the type of the plugin.
- utl::PluginPlatform getPlatform () const override
  Get the handled platform of the plugin.
- ShaderSourceType sourceType () const override
  Get the source type this frontend handles.
- ShaderIRModule compile (const ShaderSourceDesc &desc) override
  Compile shader source to intermediate representation.

Public Member Functions inherited from cae::AShaderFrontend

- ~AShaderFrontend () override=default

Public Member Functions inherited from cae::IShaderFrontend

- ~IShaderFrontend () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default

Static Private Member Functions

- static EShLanguage shaderStageToESh (const ShaderStage stage)
- static std::vector< uint32_t > compileGLSLtoSPIRV (const std::string &src, ShaderStage stage)

42.22.1 Detailed Description

Class for the GLSL plugin.
Definition at line 25 of file GLSL.hpp.

42.22.2 Constructor & Destructor Documentation

42.22.2.1 GLSL() [1/3]

cae::GLSL::GLSL () [default]

42.22.2.2 ~GLSL()

cae::GLSL::~GLSL () [override], [default]

42.22.2.3 GLSL() [2/3]

cae::GLSL::GLSL (
const GLSL &) [delete]

42.22.2.4 GLSL() [3/3]

cae::GLSL::GLSL (
GLSL &&) [delete]

42.22.3 Member Function Documentation

42.22.3.1 compile()

ShaderIRModule cae::GLSL::compile (
const ShaderSourceDesc & desc) [inline], [override], [virtual]
Compile shader source to intermediate representation.

Parameters

desc	Shader source description

Returns

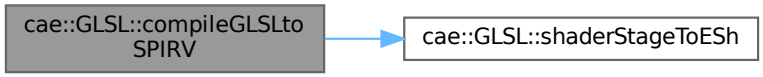
Compiled shader intermediate representation module

Implements cae::IShaderFrontend.
Definition at line 43 of file GLSL.hpp.
References compileGLSLtoSPIRV(), cae::ShaderIRModule::entryPoint, cae::ShaderIRModule::id, cae::ShaderSourceDesc::id, cae::ShaderSourceDesc::source, cae::ShaderIRModule::spirv, cae::ShaderIRModule::stage, and cae::ShaderSourceDesc::stage.
Here is the call graph for this function:

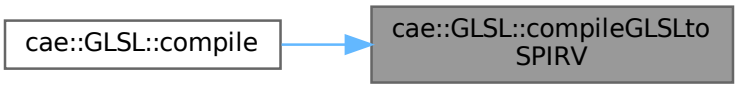
42.22.3.2 compileGLSLtoSPIRV()

std::vector< uint32_t > cae::GLSL::compileGLSLtoSPIRV (
const std::string & src,
ShaderStage stage) [static], [private]

Definition at line 5 of file glsl.cpp.
References shaderStageToESh(), and cae::VERSION.
Referenced by compile().
Here is the call graph for this function:



Here is the caller graph for this function:



42.22.3.3 getName()

std::string cae::GLSL::getName () const [inline], [nodiscard], [override], [virtual]
Get the name of the plugin.

Returns

The name of the plugin

Implements utl::IPlugin.
Definition at line 37 of file GLSL.hpp.

42.22.3.4 getPlatform()

utl::PluginPlatform cae::GLSL::getPlatform () const [inline], [nodiscard], [override], [virtual]
Get the handled platform of the plugin.

Returns

The handled platform of the plugin

Implements utl::IPlugin.
Definition at line 39 of file GLSL.hpp.
References utl::ALL.

42.22.3.5 getType()

utl::PluginType cae::GLSL::getType () const [inline], [nodiscard], [override], [virtual]
Get the type of the plugin.

Returns

The type of the plugin

Implements utl::IPlugin.
Definition at line 38 of file GLSL.hpp.
References utl::SHADER_FRONTEND.

42.22.3.6 operator=() [1/2]

GLSL & cae::GLSL::operator= (
const GLSL &) [delete]

42.22.3.7 operator=() [2/2]

GLSL & cae::GLSL::operator= (
GLSL &&) [delete]

42.22.3.8 shaderStageToESh()

static EShLanguage cae::GLSL::shaderStageToESh (
const ShaderStage stage) [inline], [static], [private]
Definition at line 54 of file GLSL.hpp.
References cae::COMPUTE, cae::FRAGMENT, cae::GEOMETRY, and cae::VERTEX.
Referenced by compileGLSLtoSPIRV().
Here is the caller graph for this function:

42.22.3.9 sourceType()

ShaderSourceType cae::GLSL::sourceType () const [inline], [nodiscard], [override], [virtual]
Get the source type this frontend handles.

Returns

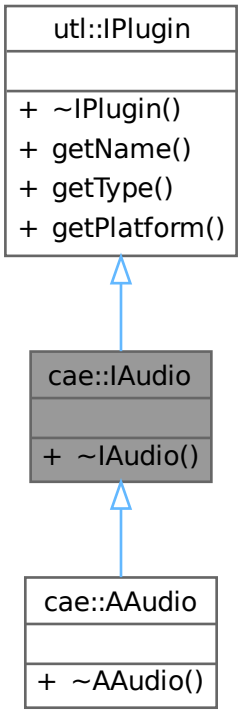
The source type this frontend handles

Implements cae::IShaderFrontend.
Definition at line 41 of file GLSL.hpp.
References cae::GLSL.
The documentation for this class was generated from the following files:

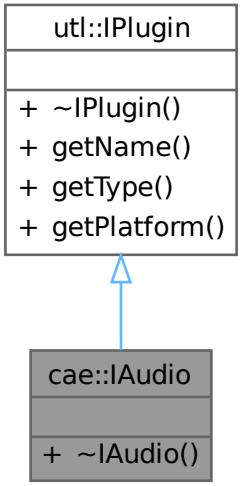
- plugins/Shader/Frontend/GLSL/include/GLSL/GLSL.hpp
- plugins/Shader/Frontend/GLSL/src/glsl.cpp

42.23 cae::IAudio Interface Reference

Interface for audio.
#include <IAudio.hpp>
Inheritance diagram for cae::IAudio:



Collaboration diagram for cae::IAudio:



Public Member Functions

- ~IAudio () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.23.1 Detailed Description

Interface for audio.
Definition at line 19 of file IAudio.hpp.

42.23.2 Constructor & Destructor Documentation

42.23.2.1 ~IAudio()

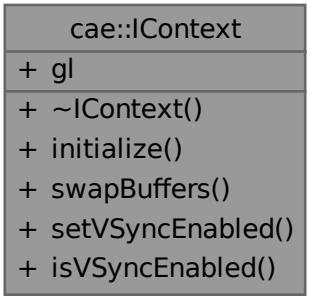
cae::IAudio::~IAudio () [override], [default]
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Audio/IAudio.hpp

42.24 cae::IContext Interface Reference

Interface for OpenGL context.
#include <IContext.hpp>

Collaboration diagram for cae::IContext:



Public Member Functions

- virtual ~IContext ()=default
- virtual void initialize (const NativeWindowHandle &window)=0

  Initialize the OpenGL context with the given window.
- virtual void swapBuffers ()=0

  Swap the front and back buffers.
- virtual void setVSyncEnabled (bool enabled)=0

  Enable or disable VSync.
- virtual bool isVSyncEnabled () const =0

  Check if VSync is enabled.

Public Attributes

- GladGLContext gl {nullptr}

42.24.1 Detailed Description

Interface for OpenGL context.
Definition at line 20 of file IContext.hpp.

42.24.2 Constructor & Destructor Documentation

42.24.2.1 ~IContext()

virtual cae::IContext::~IContext () [virtual], [default]

42.24.3 Member Function Documentation

42.24.3.1 initialize()

virtual void cae::IContext::initialize (
const NativeWindowHandle & window) [pure virtual]

Initialize the OpenGL context with the given window.

Parameters

window	The native window handle

42.24.3.2 isVSyncEnabled()

virtual bool cae::IContext::isVSyncEnabled () const [nodiscard], [pure virtual]
Check if VSync is enabled.

Returns

Whether VSync is enabled

42.24.3.3 setVSyncEnabled()

virtual void cae::IContext::setVSyncEnabled (
bool enabled) [pure virtual]
Enable or disable VSync.

Parameters

enabled	Whether VSync should be enabled

42.24.3.4 swapBuffers()

virtual void cae::IContext::swapBuffers () [pure virtual]
Swap the front and back buffers.

42.24.4 Member Data Documentation

42.24.4.1 gl

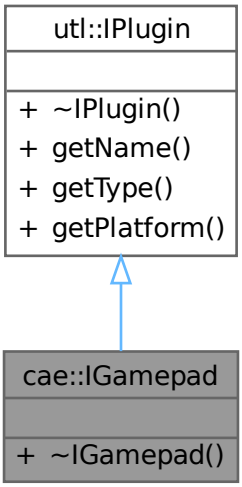
GladGLContext cae::IContext::gl {nullptr}
Definition at line 48 of file IContext.hpp.
The documentation for this interface was generated from the following file:

- plugins/Renderer/OpenGL/include/OPGL/Context/IContext.hpp

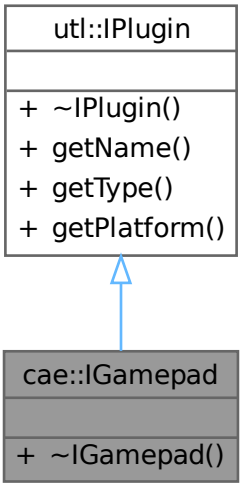
42.25 cae::IGamepad Interface Reference

Interface for gamepad.
#include <IGamepad.hpp>

Inheritance diagram for cae::IGamepad:



Collaboration diagram for cae::IGamepad:



Public Member Functions

- ~IGamepad () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

Get the name of the plugin.

- virtual PluginType getType () const =0

  Get the type of the plugin.

- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.25.1 Detailed Description

Interface for gamepad.
Definition at line 19 of file IGamepad.hpp.

42.25.2 Constructor & Destructor Documentation

42.25.2.1 ~IGamepad()

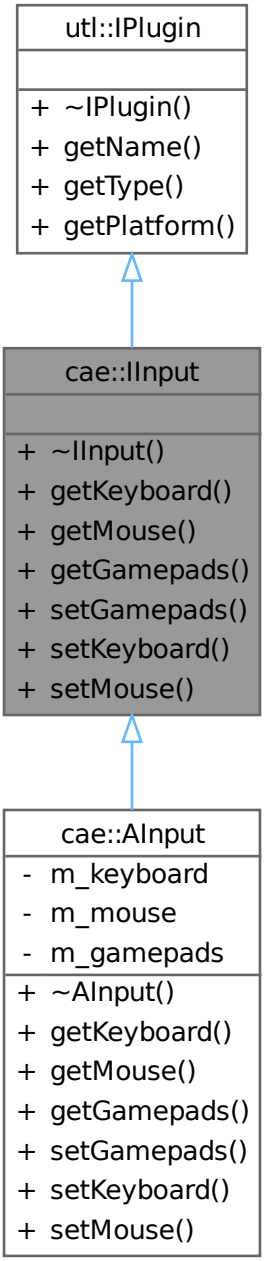
cae::IGamepad::~IGamepad () [override], [default]
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Input/IGamepad.hpp

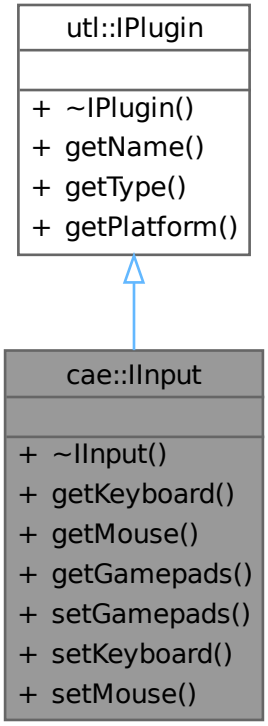
42.26 cae::IInput Interface Reference

Interface for inputs.
#include <IInput.hpp>

Inheritance diagram for cae::IInput:



Collaboration diagram for cae::IInput:



Public Member Functions

- ~IInput () override=default
- virtual const std::unique_ptr< IKeyboard > & getKeyboard () const =0
- virtual const std::unique_ptr< IMouse > & getMouse () const =0
- virtual const std::vector< std::unique_ptr< IGamepad > > & getGamepads () const =0
- virtual void setGamepads (std::vector< std::unique_ptr< IGamepad > > &gamepads)=0
- virtual void setKeyboard (std::unique_ptr< IKeyboard > &keyboard)=0
- virtual void setMouse (std::unique_ptr< IMouse > &mouse)=0

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.26.1 Detailed Description

Interface for inputs.
Definition at line 21 of file IInput.hpp.

42.26.2 Constructor & Destructor Documentation

42.26.2.1 ~IInput()

cae::IInput::~IInput () [override], [default]

42.26.3 Member Function Documentation

42.26.3.1 getGamepads()

virtual const std::vector< std::unique_ptr< IGamepad > > & cae::IInput::getGamepads () const [pure virtual]
Implemented in cae::AInput.

42.26.3.2 getKeyboard()

virtual const std::unique_ptr< IKeyboard > & cae::IInput::getKeyboard () const [pure virtual]
Implemented in cae::AInput.

42.26.3.3 getMouse()

virtual const std::unique_ptr< IMouse > & cae::IInput::getMouse () const [pure virtual]
Implemented in cae::AInput.

42.26.3.4 setGamepads()

virtual void cae::IInput::setGamepads (
std::vector< std::unique_ptr< IGamepad > > & gamepads) [pure virtual]
Implemented in cae::AInput.

42.26.3.5 setKeyboard()

virtual void cae::IInput::setKeyboard (
std::unique_ptr< IKeyboard > & keyboard) [pure virtual]
Implemented in cae::AInput.

42.26.3.6 setMouse()

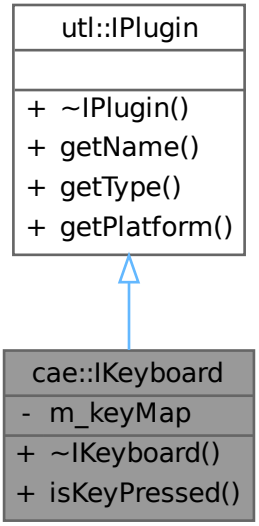
virtual void cae::IInput::setMouse (
std::unique_ptr< IMouse > & mouse) [pure virtual]
Implemented in cae::AInput.
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Input/IInput.hpp

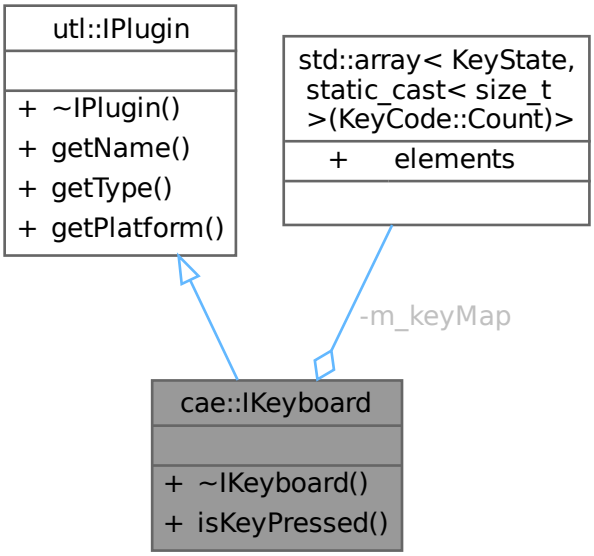
42.27 cae::IKeyboard Interface Reference

Interface for keyboard.
#include <IKeyboard.hpp>

Inheritance diagram for cae::IKeyboard:



Collaboration diagram for cae::IKeyboard:



Public Member Functions

- ~IKeyboard () override=default
- virtual bool isKeyPressed (KeyCode keyCode) const =0

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

Private Attributes

- std::array< KeyState, static_cast< size_t >(KeyCode::Count)> m_keyMap {}

42.27.1 Detailed Description

Interface for keyboard.
Definition at line 23 of file IKeyboard.hpp.

42.27.2 Constructor & Destructor Documentation

42.27.2.1 ~IKeyboard()

cae::IKeyboard::~IKeyboard () [override], [default]

42.27.3 Member Function Documentation

42.27.3.1 isKeyPressed()

virtual bool cae::IKeyboard::isKeyPressed (
KeyCode keyCode) const [pure virtual]

42.27.4 Member Data Documentation

42.27.4.1 m_keyMap

std::array<KeyState, static_cast<size_t>(KeyCode::Count)> cae::IKeyboard::m_keyMap {} [private]
Definition at line 32 of file IKeyboard.hpp.
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Input/IKeyboard.hpp

42.28 utl::Image Struct Reference

Struct for image.
#include <Image.hpp>

Collaboration diagram for utl::Image:

Public Types

- using pixel = unsigned char *

Public Member Functions

- Image (const std::filesystem::path &path, bool flip=false)
  Load an image from a file.
- ~Image ()

Public Attributes

- pixel pixels = nullptr
- int width = 0
- int height = 0
- int channels = 0

42.28.1 Detailed Description

Struct for image.
Definition at line 19 of file Image.hpp.

42.28.2 Member Typedef Documentation

42.28.2.1 pixel

using utl::Image::pixel = unsigned char *
Definition at line 22 of file Image.hpp.

42.28.3 Constructor & Destructor Documentation

42.28.3.1 Image()

utl::Image::Image (
const std::filesystem::path & path,
bool flip = false) [explicit]
Load an image from a file.

Parameters

path	Path to the image file

Parameters

flip	Whether to flip the image vertically

Definition at line 8 of file image.cpp.
References channels, height, pixels, and width.

42.28.3.2 ~Image()

utl::Image::~Image ()
Definition at line 21 of file image.cpp.

42.28.4 Member Data Documentation

42.28.4.1 channels

int utl::Image::channels = 0
Definition at line 35 of file Image.hpp.
Referenced by Image().

42.28.4.2 height

int utl::Image::height = 0
Definition at line 34 of file Image.hpp.
Referenced by Image(), and cae::GLFW::setIcon().

42.28.4.3 pixels

pixel utl::Image::pixels = nullptr
Definition at line 32 of file Image.hpp.
Referenced by Image(), and cae::GLFW::setIcon().

42.28.4.4 width

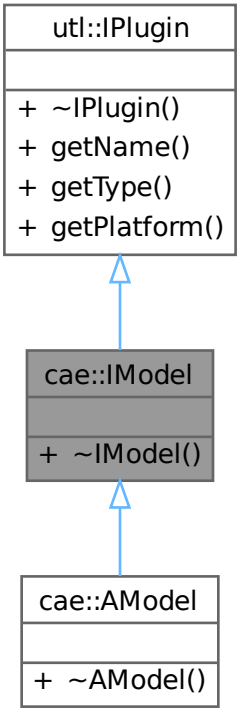
int utl::Image::width = 0
Definition at line 33 of file Image.hpp.
Referenced by Image(), and cae::GLFW::setIcon().
The documentation for this struct was generated from the following files:

- modules/Utils/include/Utils/Image.hpp
- modules/Utils/src/image.cpp

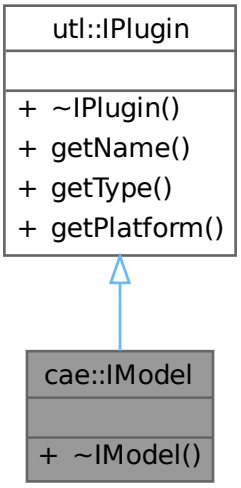
42.29 cae::IModel Interface Reference

Interface for model.
#include <IModel.hpp>

Inheritance diagram for cae::IModel:



Collaboration diagram for cae::IModel:



Public Member Functions

- ~IModel () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.29.1 Detailed Description

Interface for model.
Definition at line 19 of file IModel.hpp.

42.29.2 Constructor & Destructor Documentation

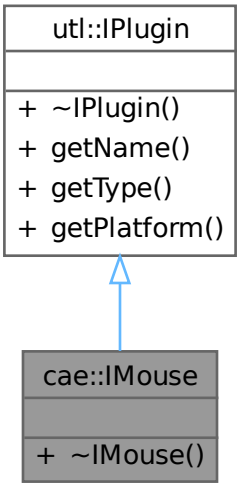
42.29.2.1 ~IModel()

cae::IModel::~IModel () [override], [default]
The documentation for this interface was generated from the following file:

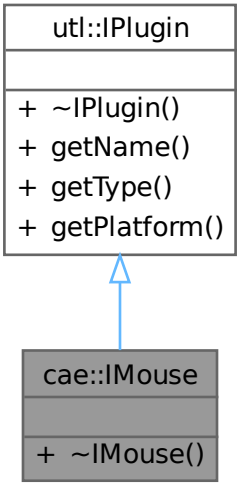
- modules/Interfaces/include/Interfaces/Model/IModel.hpp

42.30 cae::IMouse Interface Reference

Interface for mouse.
#include <IMouse.hpp>
Inheritance diagram for cae::IMouse:



Collaboration diagram for cae::IMouse:



Public Member Functions

- ~IMouse () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.30.1 Detailed Description

Interface for mouse.
Definition at line 19 of file IMouse.hpp.

42.30.2 Constructor & Destructor Documentation

42.30.2.1 ~IMouse()

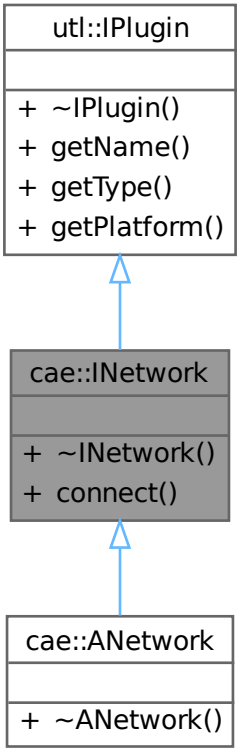
cae::IMouse::~IMouse () [override], [default]
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Input/IMouse.hpp

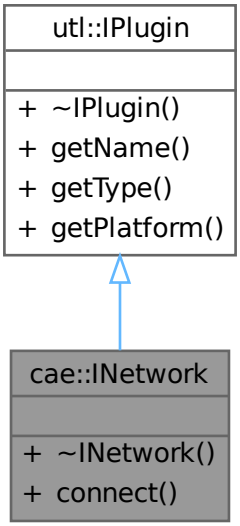
42.31 cae::INetwork Interface Reference

Interface for network.
#include <INetwork.hpp>

Inheritance diagram for cae::INetwork:



Collaboration diagram for cae::INetwork:



Public Member Functions

- ~INetwork () override=default
- virtual bool connect (const std::string &host, uint16_t port)=0

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0
  Get the name of the plugin.
- virtual PluginType getType () const =0
  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0
  Get the handled platform of the plugin.

42.31.1 Detailed Description

Interface for network.
Definition at line 19 of file INetwork.hpp.

42.31.2 Constructor & Destructor Documentation

42.31.2.1 ~INetwork()

cae::INetwork::~INetwork () [override], [default]

42.31.3 Member Function Documentation

42.31.3.1 connect()

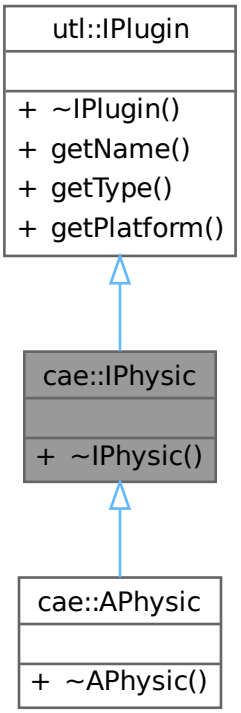
virtual bool cae::INetwork::connect (
const std::string & host,
uint16_t port) [pure virtual]

The documentation for this interface was generated from the following file:

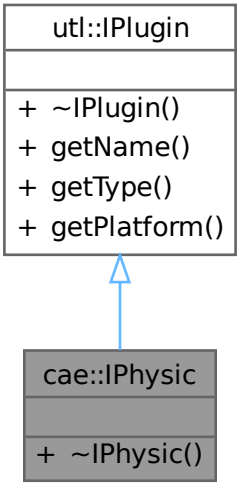
- modules/Interfaces/include/Interfaces/Network/INetwork.hpp

42.32 cae::IPhysic Interface Reference

Interface for physics.
#include <IPhysic.hpp>
Inheritance diagram for cae::IPhysic:



Collaboration diagram for cae::IPhysic:



Public Member Functions

- ~IPhysic () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0
  Get the name of the plugin.
- virtual PluginType getType () const =0
  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0
  Get the handled platform of the plugin.

42.32.1 Detailed Description

Interface for physics.
Definition at line 19 of file IPhysic.hpp.

42.32.2 Constructor & Destructor Documentation

42.32.2.1 ~IPhysic()

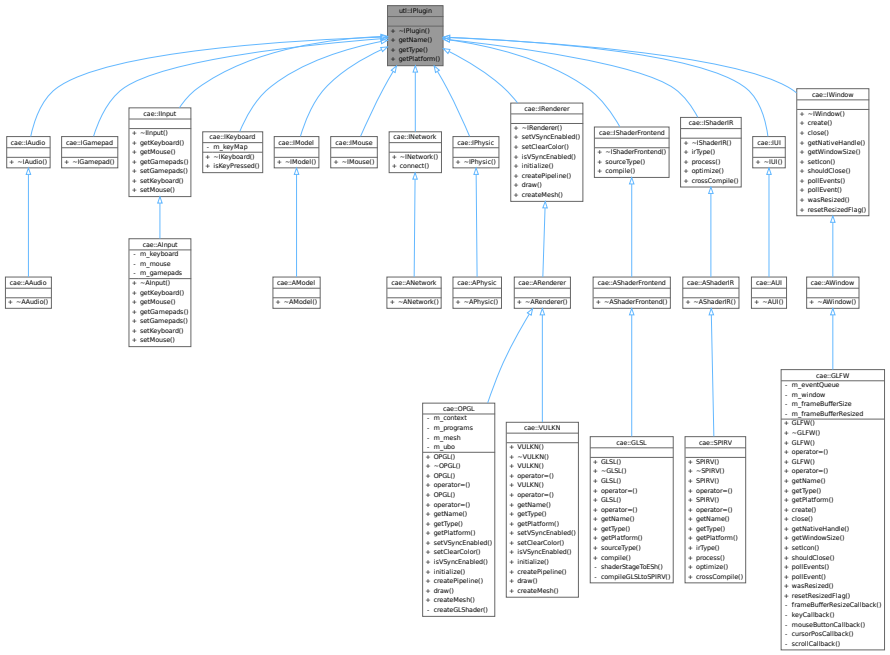
cae::IPhysic::~IPhysic () [override], [default]
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Physic/IPhysic.hpp

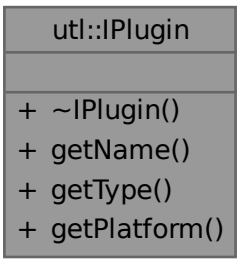
42.33 utl::IPlugin Interface Reference

Interface for plugins.
#include <IPlugin.hpp>

Inheritance diagram for utl::IPlugin:



Collaboration diagram for utl::IPlugin:



Public Member Functions

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.33.1 Detailed Description

Interface for plugins.
Definition at line 45 of file IPlugin.hpp.

42.33.2 Constructor & Destructor Documentation

42.33.2.1 ~IPlugin()

virtual utl::IPlugin::~IPlugin () [virtual], [default]

42.33.3 Member Function Documentation

42.33.3.1 getName()

virtual std::string utl::IPlugin::getName () const [nodiscard], [pure virtual]
Get the name of the plugin.

Returns

The name of the plugin

Implemented in cae::GLFW, cae::GLSL, cae::OPGL, cae::SPIRV, and cae::VULKN.

42.33.3.2 getPlatform()

virtual PluginPlatform utl::IPlugin::getPlatform () const [nodiscard], [pure virtual]
Get the handled platform of the plugin.

Returns

The handled platform of the plugin

Implemented in cae::GLFW, cae::GLSL, cae::OPGL, cae::SPIRV, and cae::VULKN.

42.33.3.3 getType()

virtual PluginType utl::IPlugin::getType () const [nodiscard], [pure virtual]
Get the type of the plugin.

Returns

The type of the plugin

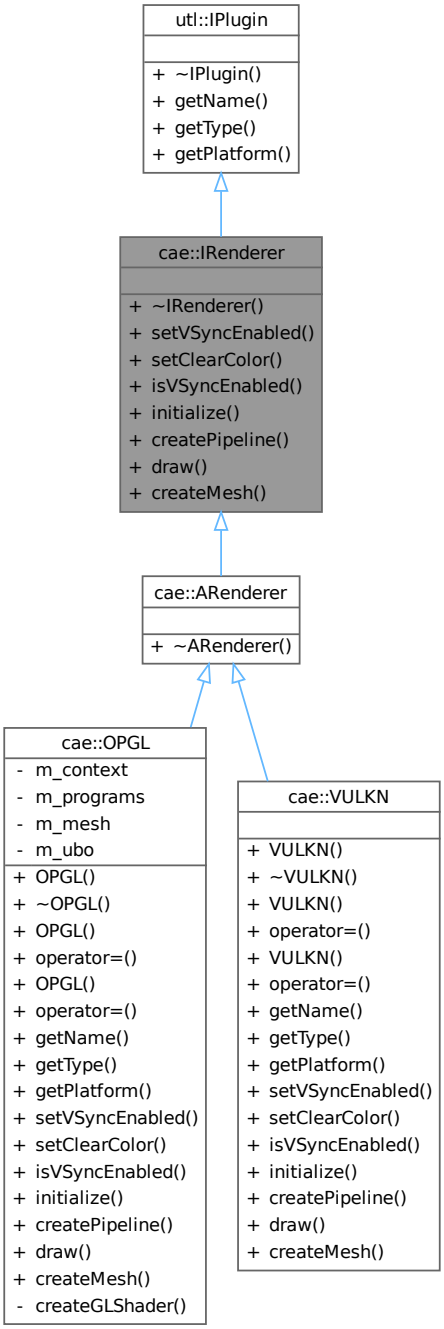
Implemented in cae::GLFW, cae::GLSL, cae::OPGL, cae::SPIRV, and cae::VULKN.
The documentation for this interface was generated from the following file:

- modules/Utils/include/Utils/Interfaces/IPlugin.hpp

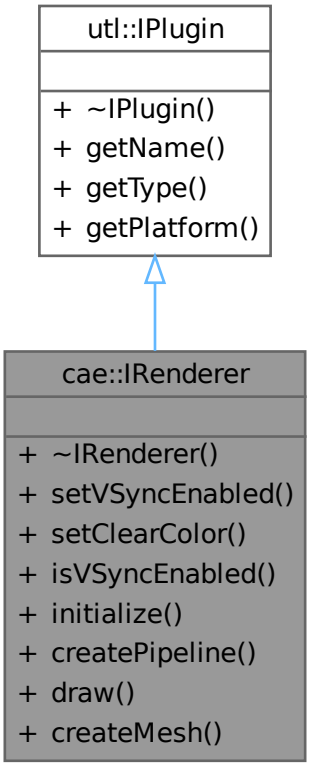
42.34 cae::IRenderer Interface Reference

Interface for renderer.
#include <IRenderer.hpp>

Inheritance diagram for cae::IRenderer:



Collaboration diagram for cae::IRenderer:



Public Member Functions

- ~IRenderer () override=default
- virtual void setVSyncEnabled (bool enabled)=0

  Enable or disable VSync.
- virtual void setClearColor (const Color &color)=0

  Set the clear color.
- virtual bool isVSyncEnabled () const =0

  Check if VSync is enabled.
- virtual void initialize (const NativeWindowHandle &nativeWindowHandle, const Color &clear↩ Color={.r=1.F,.g=1.F,.b=1.F,.a=1.F})=0

  Initialize the renderer with a native window handle and clear color.
- virtual void createPipeline (const ShaderID &id, const ShaderIRModule &vertex, const ShaderIRModule &fragment)=0

  Create a rendering pipeline with vertex and fragment shaders.
- virtual void draw (const WindowSize &windowSize, const ShaderID &shaderId, glm::mat4 mvp)=0

  Draw the scene using the specified shader and window size.
- virtual void createMesh (const std::vector< float > &vertices)=0

  Create a mesh with the given vertex data.

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.34.1 Detailed Description

Interface for renderer.
Definition at line 35 of file IRenderer.hpp.

42.34.2 Constructor & Destructor Documentation

42.34.2.1 ~IRenderer()

cae::IRenderer::~IRenderer () [override], [default]

42.34.3 Member Function Documentation

42.34.3.1 createMesh()

virtual void cae::IRenderer::createMesh (
const std::vector< float > & vertices) [pure virtual]

Create a mesh with the given vertex data.

Parameters

vertices	Vertex data to create the mesh

Implemented in cae::OPGL, and cae::VULKN.

42.34.3.2 createPipeline()

virtual void cae::IRenderer::createPipeline (
const ShaderID & id,
const ShaderIRModule & vertex,
const ShaderIRModule & fragment) [pure virtual]

Create a rendering pipeline with vertex and fragment shaders.

Parameters

id	Shader ID
vertex	Vertex shader IR module
fragment	Fragment shader IR module

Implemented in cae::OPGL, and cae::VULKN.

42.34.3.3 draw()

virtual void cae::IRenderer::draw (
const WindowSize & windowSize,
const ShaderID & shaderId,
glm::mat4 mvp) [pure virtual]

Draw the scene using the specified shader and window size.

Parameters

windowSize	Current window size
shaderId	Shader ID to use for drawing
mvp	Model-View-Projection matrix

Implemented in cae::OPGL, and cae::VULKN.

42.34.3.4 initialize()

virtual void cae::IRenderer::initialize (
const NativeWindowHandle & nativeWindowHandle,
const Color & clearColor = {.r=1.F,.g=1.F,.b=1.F,.a=1.F}) [pure virtual]

Initialize the renderer with a native window handle and clear color.

Parameters

nativeWindowHandle	Native window handle
clearColor	Clear color (default: white)

Implemented in cae::OPGL, and cae::VULKN.
References cae::Color::r.

42.34.3.5 isVSyncEnabled()

virtual bool cae::IRenderer::isVSyncEnabled () const [nodiscard], [pure virtual]

Check if VSync is enabled.

Returns

Whether VSync is enabled

Implemented in cae::OPGL, and cae::VULKN.

42.34.3.6 setClearColor()

virtual void cae::IRenderer::setClearColor (
const Color & color) [pure virtual]

Set the clear color.

Parameters

color	Clear color to set

Implemented in cae::OPGL, and cae::VULKN.

42.34.3.7 setVSyncEnabled()

virtual void cae::IRenderer::setVSyncEnabled (
bool enabled) [pure virtual]

Enable or disable VSync.

Parameters

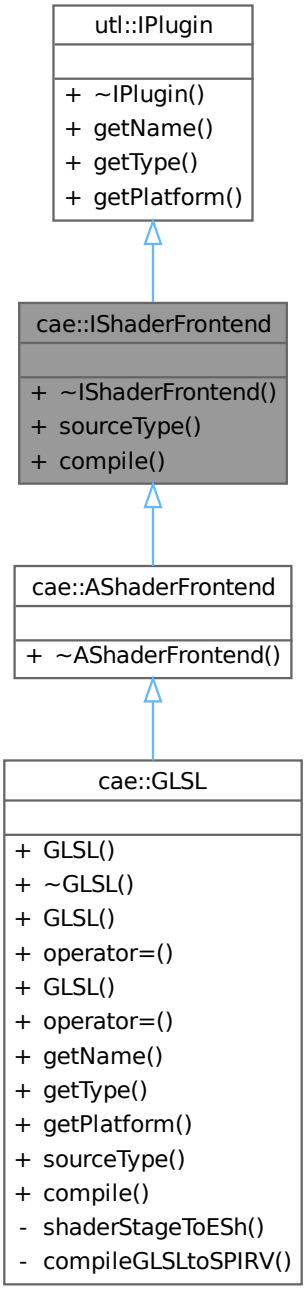
enabled	Whether VSync is enabled

Implemented in cae::OPGL, and cae::VULKN.
The documentation for this interface was generated from the following file:

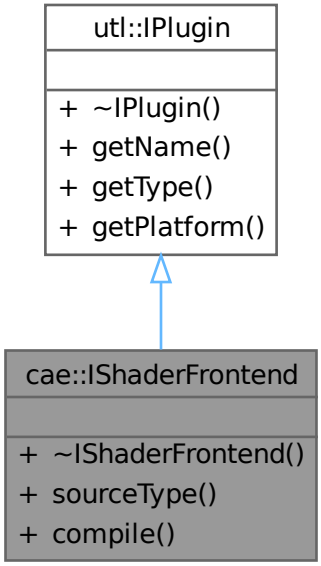
- modules/Interfaces/include/Interfaces/Renderer/IRenderer.hpp

42.35 cae::IShaderFrontend Interface Reference

Interface for shaders frontend.
#include <IShaderFrontend.hpp>
Inheritance diagram for cae::IShaderFrontend:



Collaboration diagram for cae::IShaderFrontend:



Public Member Functions

- ~IShaderFrontend () override=default
- virtual ShaderSourceType sourceType () const =0

  Get the source type this frontend handles.
- virtual ShaderIRModule compile (const ShaderSourceDesc &desc)=0

  Compile shader source to intermediate representation.

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.35.1 Detailed Description

Interface for shaders frontend.
Definition at line 80 of file IShaderFrontend.hpp.

42.35.2 Constructor & Destructor Documentation

42.35.2.1 ~IShaderFrontend()

cae::IShaderFrontend::~IShaderFrontend () [override], [default]

42.35.3 Member Function Documentation

42.35.3.1 compile()

virtual ShaderIRModule cae::IShaderFrontend::compile (
const ShaderSourceDesc & desc) [pure virtual]
Compile shader source to intermediate representation.

Parameters

desc	Shader source description

Returns

Compiled shader intermediate representation module

Implemented in cae::GLSL.

42.35.3.2 sourceType()

virtual ShaderSourceType cae::IShaderFrontend::sourceType () const [pure virtual]
Get the source type this frontend handles.

Returns

The source type this frontend handles

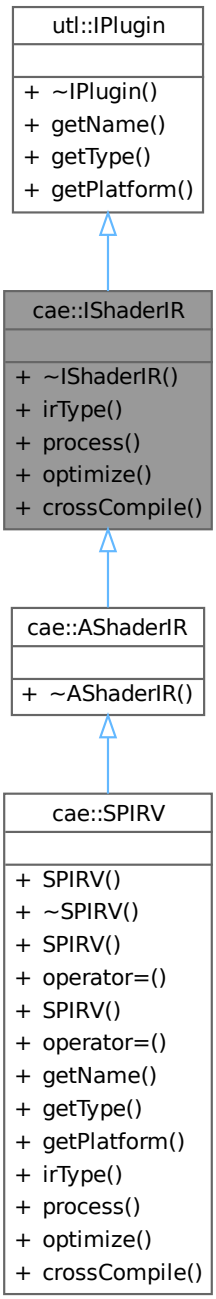
Implemented in cae::GLSL.
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Shader/Frontend/IShaderFrontend.hpp

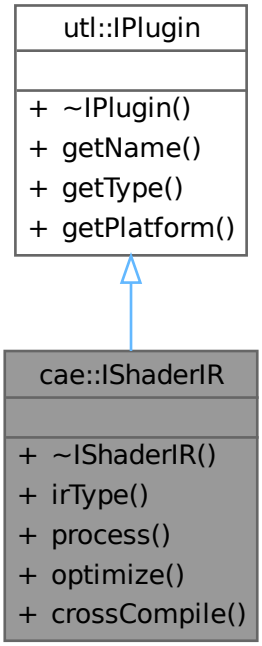
42.36 cae::IShaderIR Interface Reference

Interface for shaders IR.
#include <IShaderIR.hpp>

Inheritance diagram for cae::IShaderIR:



Collaboration diagram for cae::IShaderIR:



Public Member Functions

- ~IShaderIR () override=default
- virtual ShaderSourceType irType () const =0

  Get the IR type this processor handles.
- virtual ShaderIRModule process (const ShaderIRModule &module)=0

  Transform or validate a shader IR module.
- virtual void optimize (std::span< ShaderIRModule > modules)

  Optional: optimize a batch of IR modules.
- virtual ShaderIRModule crossCompile (const ShaderIRModule &module, ShaderSourceType targetType)

  Optional: cross-compile from one IR to another (SPIR-V -> MSL, etc.)

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.36.1 Detailed Description

Interface for shaders IR.
Definition at line 22 of file IShaderIR.hpp.

42.36.2 Constructor & Destructor Documentation

42.36.2.1 ~IShaderIR()

cae::IShaderIR::~IShaderIR () [override], [default]

42.36.3 Member Function Documentation

42.36.3.1 crossCompile()

virtual ShaderIRModule cae::IShaderIR::crossCompile (
const ShaderIRModule & module,
ShaderSourceType targetType) [inline], [virtual]
Optional: cross-compile from one IR to another (SPIR-V -> MSL, etc.)
Implemented in cae::SPIRV.
Definition at line 49 of file IShaderIR.hpp.

42.36.3.2 irType()

virtual ShaderSourceType cae::IShaderIR::irType () const [pure virtual]
Get the IR type this processor handles.

Returns

The IR type this processor handles

Implemented in cae::SPIRV.

42.36.3.3 optimize()

virtual void cae::IShaderIR::optimize (
std::span< ShaderIRModule > modules) [inline], [virtual]
Optional: optimize a batch of IR modules.
Implemented in cae::SPIRV.
Definition at line 44 of file IShaderIR.hpp.

42.36.3.4 process()

virtual ShaderIRModule cae::IShaderIR::process (
const ShaderIRModule & module) [pure virtual]
Transform or validate a shader IR module.

Parameters

module	The input IR module

Returns

Transformed IR module ready for the backend

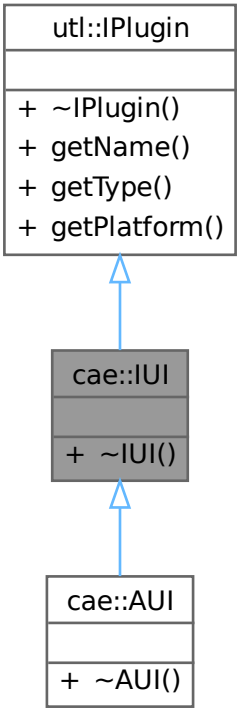
Implemented in cae::SPIRV.
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Shader/IR/IShaderIR.hpp

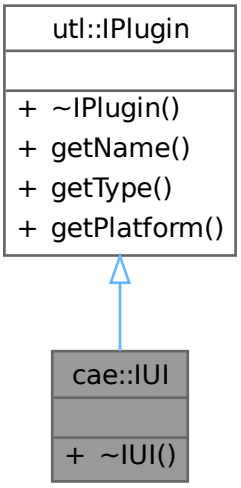
42.37 cae::IUI Interface Reference

Interface for ui.
#include <IUI.hpp>

Inheritance diagram for cae::IUI:



Collaboration diagram for cae::IUI:



Public Member Functions

- ~IUI () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.
- virtual PluginType getType () const =0

  Get the type of the plugin.
- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.37.1 Detailed Description

Interface for ui.
Definition at line 19 of file IUI.hpp.

42.37.2 Constructor & Destructor Documentation

42.37.2.1 ~IUI()

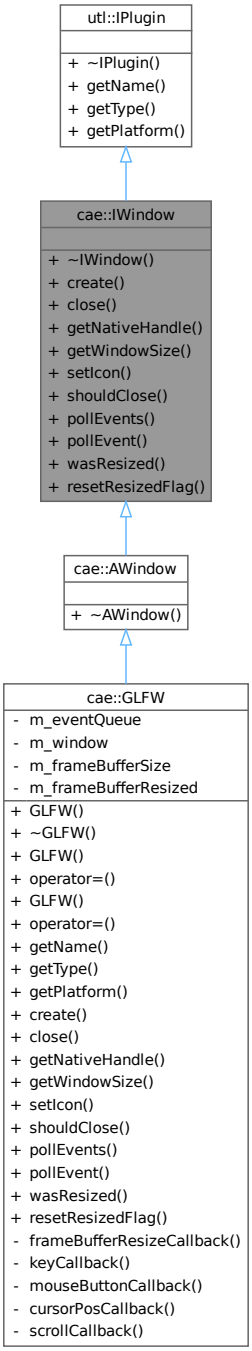
cae::IUI::~IUI () [override], [default]
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/UI/IUI.hpp

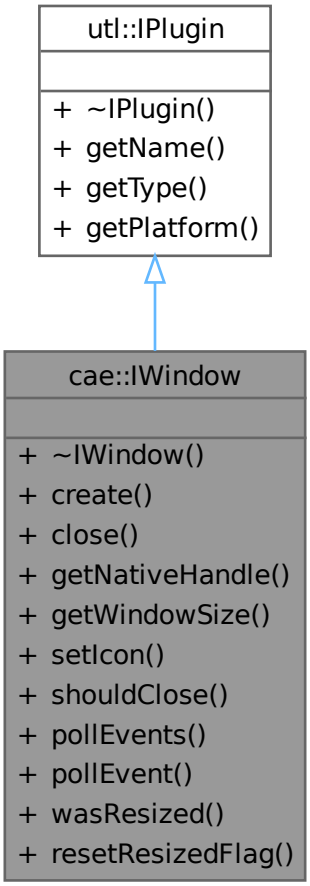
42.38 cae::IWindow Interface Reference

Abstract class for window.
#include <IWindow.hpp>

Inheritance diagram for cae::IWindow:



Collaboration diagram for cae::IWindow:



Public Member Functions

- ~IWindow () override=default
- virtual bool create (const std::string &name, WindowSize size)=0

  Create a window with the given name and size.
- virtual void close ()=0

  Close the window.
- virtual NativeWindowHandle getNativeHandle () const =0

  Get the native window handle.
- virtual WindowSize getWindowSize () const =0

  Get the current window size.
- virtual void setIcon (const std::string &path) const =0

  Set the window icon from the given image path.
- virtual bool shouldClose () const =0

  Check if the window should close.
- virtual void pollEvents ()=0

  Poll window events.
- virtual bool pollEvent (WindowEvent &outEvent)=0

Poll window events into outEvent.

- virtual bool wasResized () const =0

  Check if the window was resized.

- virtual void resetResizedFlag ()=0

  Reset the resized flag.

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default
- virtual std::string getName () const =0

  Get the name of the plugin.

- virtual PluginType getType () const =0

  Get the type of the plugin.

- virtual PluginPlatform getPlatform () const =0

  Get the handled platform of the plugin.

42.38.1 Detailed Description

Abstract class for window.
Interface for window.
Definition at line 96 of file IWindow.hpp.

42.38.2 Constructor & Destructor Documentation

42.38.2.1 ~IWindow()

cae::IWindow::~IWindow () [override], [default]

42.38.3 Member Function Documentation

42.38.3.1 close()

virtual void cae::IWindow::close () [pure virtual]
Close the window.
Implemented in cae::GLFW.

42.38.3.2 create()

virtual bool cae::IWindow::create (
const std::string & name,
WindowSize size) [pure virtual]
Create a window with the given name and size.

Parameters

name	Window name
size	Window size

Returns

True if the window was created successfully

Implemented in cae::GLFW.

42.38.3.3 getNativeHandle()

virtual NativeWindowHandle cae::IWindow::getNativeHandle () const [pure virtual]
Get the native window handle.

Returns

Native window handle

Implemented in cae::GLFW.

42.38.3.4 getWindowSize()

virtual WindowSize cae::IWindow::getWindowSize () const [pure virtual]
Get the current window size.

Returns

Current window size

Implemented in cae::GLFW.

42.38.3.5 pollEvent()

virtual bool cae::IWindow::pollEvent (
WindowEvent & outEvent) [pure virtual]
Poll window events into outEvent.

Parameters

outEvent	Event to be filled

Returns

True if an event was polled

Implemented in cae::GLFW.

42.38.3.6 pollEvents()

virtual void cae::IWindow::pollEvents () [pure virtual]
Poll window events.
Implemented in cae::GLFW.

42.38.3.7 resetResizedFlag()

virtual void cae::IWindow::resetResizedFlag () [pure virtual]
Reset the resized flag.
Implemented in cae::GLFW.

42.38.3.8 setIcon()

virtual void cae::IWindow::setIcon (
const std::string & path) const [pure virtual]
Set the window icon from the given image path.

Parameters

path	Path to the icon image

Returns

True if the icon was set successfully

Implemented in cae::GLFW.

42.38.3.9 shouldClose()

virtual bool cae::IWindow::shouldClose () const [pure virtual]
Check if the window should close.

Returns

True if the window should close

Implemented in cae::GLFW.

42.38.3.10 wasResized()

virtual bool cae::IWindow::wasResized () const [pure virtual]
Check if the window was resized.

Returns

True if the window was resized

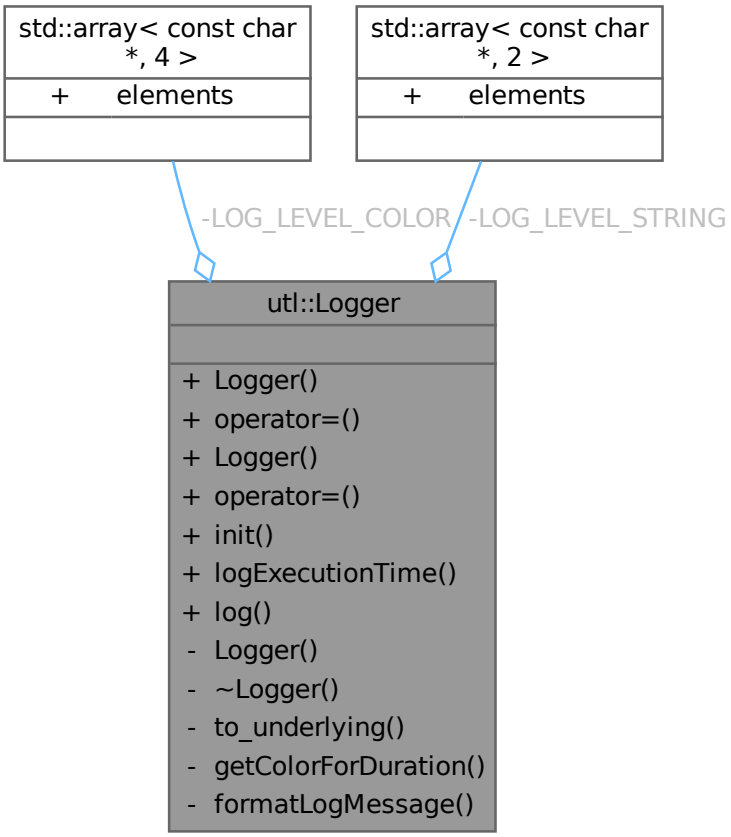
Implemented in cae::GLFW.
The documentation for this interface was generated from the following file:

- modules/Interfaces/include/Interfaces/Window/IWindow.hpp

42.39 utl::Logger Class Reference

Class for logging.
#include <Logger.hpp>

Collaboration diagram for utl::Logger:



Public Member Functions

- Logger (const Logger &)=delete
- Logger & operator= (const Logger &)=delete
- Logger (Logger &&)=delete
- Logger & operator= (Logger &&)=delete

Static Public Member Functions

- static void init ()

  Initialize the logger.
- template<typename Func >
  static void logExecutionTime (const std::string &message, Func &&func)

  Log the execution time of a function.
- static void log (const std::string &message, const LogLevel &logLevel)

  Log a message with a specific log level.

Private Types

- enum class ColorIndex : uint8_t { COLOR_ERROR = 0 , COLOR_INFO = 1 , COLOR_WARNING = 2 , COLOR_RESET = 3 }

Private Member Functions

- Logger ()=default
- ~Logger ()=default

Static Private Member Functions

- template<typename E >
  static constexpr auto to_underlying (E e) noexcept
- static const char * getColorForDuration (const float duration)
- static std::string formatLogMessage (LogLevel level, const std::string &message)

Static Private Attributes

- static constexpr std::array< const char *, 4 > LOG_LEVEL_COLOR
- static constexpr std::array< const char *, 2 > LOG_LEVEL_STRING = {"INFO", "WARNING"}

42.39.1 Detailed Description

Class for logging.
Definition at line 28 of file Logger.hpp.

42.39.2 Member Enumeration Documentation

42.39.2.1 ColorIndex

enum class utl::Logger::ColorIndex : uint8_t [strong], [private]

Enumerator

COLOR_ERROR	
COLOR_INFO	
COLOR_WARNING	
COLOR_RESET	

Definition at line 80 of file Logger.hpp.

42.39.3 Constructor & Destructor Documentation

42.39.3.1 Logger() [1/3]

utl::Logger::Logger (
const Logger &) [delete]

42.39.3.2 Logger() [2/3]

utl::Logger::Logger (
Logger &&) [delete]

42.39.3.3 Logger() [3/3]

utl::Logger::Logger () [private], [default]

42.39.3.4 ~Logger()

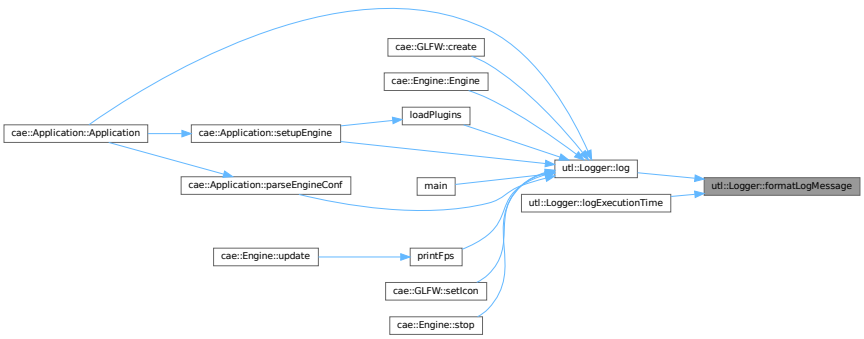
utl::Logger::~Logger () [private], [default]

42.39.4 Member Function Documentation

42.39.4.1 formatLogMessage()

static std::string utl::Logger::formatLogMessage (
LogLevel level,
const std::string & message) [inline], [static], [nodiscard], [private]

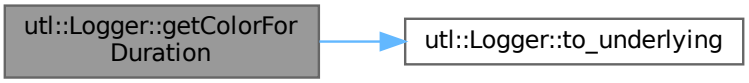
Definition at line 107 of file Logger.hpp.
References LOG_LEVEL_STRING.
Referenced by log(), and logExecutionTime().
Here is the caller graph for this function:



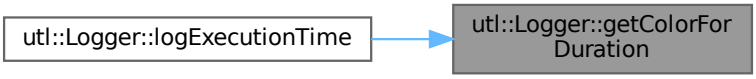
42.39.4.2 getColorForDuration()

static const char * utl::Logger::getColorForDuration (
const float duration) [inline], [static], [nodiscard], [private]

Definition at line 100 of file Logger.hpp.
References COLOR_ERROR, COLOR_INFO, COLOR_WARNING, LOG_LEVEL_COLOR, and to_underlying().
Referenced by logExecutionTime().
Here is the call graph for this function:



Here is the caller graph for this function:



42.39.4.3 init()

void utl::Logger::init () [static]
Initialize the logger.
Definition at line 7 of file logger.cpp.
Referenced by main().
Here is the caller graph for this function:

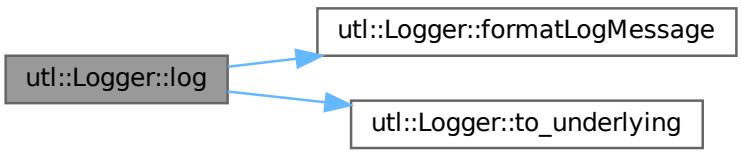
42.39.4.4 log()

static void utl::Logger::log (
const std::string & message,
const LogLevel & logLevel) [inline], [static]
Log a message with a specific log level.

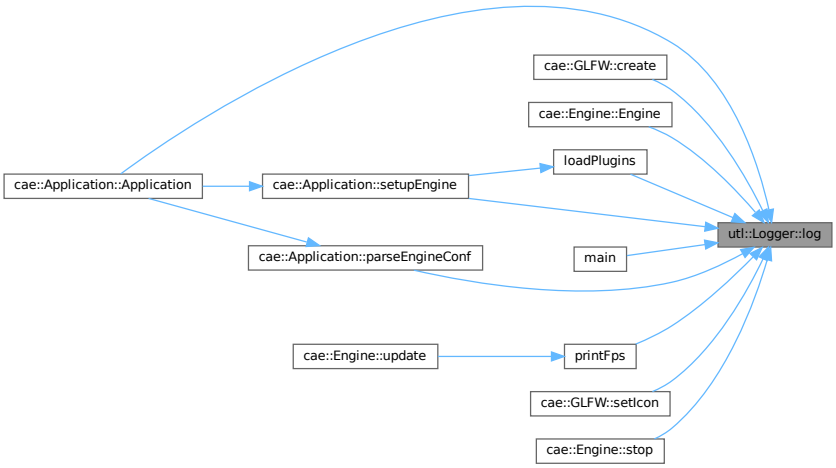
Parameters

message	Message to be logged
logLevel	Log level of the message

Definition at line 71 of file Logger.hpp.
References COLOR_INFO, COLOR_RESET, COLOR_WARNING, formatLogMessage(), utl::INFO, LOG_LEVEL_COLOR, and to_underlying().
Referenced by cae::Application::Application(), cae::GLFW::create(), cae::Engine::Engine(), loadPlugins(), main(), cae::Application::parseEngineConf(), printFps(), cae::GLFW::setIcon(), cae::Application::setupEngine(), and cae::Engine::stop().
Here is the call graph for this function:



Here is the caller graph for this function:



42.39.4.5 logExecutionTime()

template<typename Func >
static void utl::Logger::logExecutionTime (
const std::string & message,
Func && func) [inline], [static]

Log the execution time of a function.

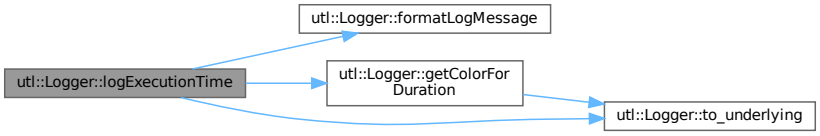
Template Parameters

Func	Function to be measured

Parameters

message	Message to be logged
func	Function to be measured

Definition at line 54 of file Logger.hpp.
References COLOR_RESET, formatLogMessage(), getColorForDuration(), utl::INFO, LOG_LEVEL_COLOR, and to_underlying().
Here is the call graph for this function:



42.39.4.6 operator=() [1/2]

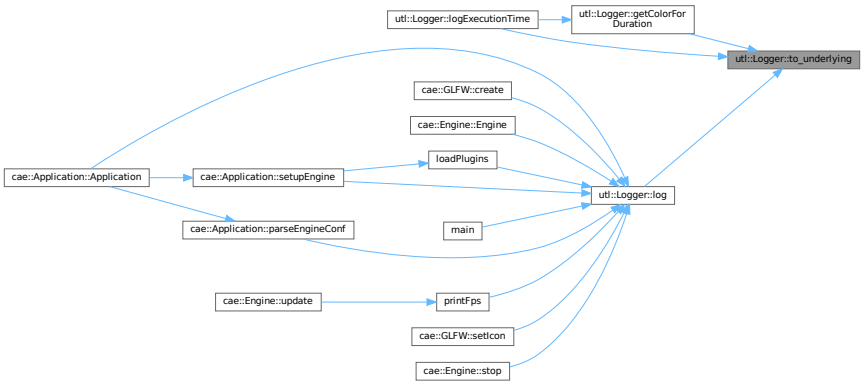
Logger & utl::Logger::operator= (
const Logger &) [delete]

42.39.4.7 operator=() [2/2]

Logger & utl::Logger::operator= (
Logger &&) [delete]

42.39.4.8 to_underlying()

template<typename E >
static constexpr auto utl::Logger::to_underlying (
E e) [inline], [static], [constexpr], [private], [noexcept]
Definition at line 32 of file Logger.hpp.
Referenced by getColorForDuration(), log(), and logExecutionTime().
Here is the caller graph for this function:



42.39.5 Member Data Documentation

42.39.5.1 LOG_LEVEL_COLOR

std::array<const char *, 4> utl::Logger::LOG_LEVEL_COLOR [static], [constexpr], [private]
Initial value:
= {
"\033[31m",
"\033[32m",
"\033[33m",
"\033[0m\n"
}
Definition at line 88 of file Logger.hpp.
Referenced by getColorForDuration(), log(), and logExecutionTime().

42.39.5.2 LOG_LEVEL_STRING

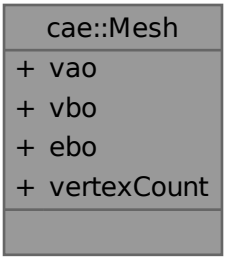
std::array<const char *, 2> utl::Logger::LOG_LEVEL_STRING = {"INFO", "WARNING"} [static], [constexpr], [private]
Definition at line 95 of file Logger.hpp.
Referenced by formatLogMessage().
The documentation for this class was generated from the following files:

- modules/Utils/include/Utils/Logger.hpp
- modules/Utils/src/logger.cpp

42.40 cae::Mesh Struct Reference

#include <OPGL.hpp>

Collaboration diagram for cae::Mesh:



Public Attributes

- GLuint vao = 0
- GLuint vbo = 0
- GLuint ebo = 0
- GLsizei vertexCount = 0

42.40.1 Detailed Description

Definition at line 19 of file OPGL.hpp.

42.40.2 Member Data Documentation

42.40.2.1 ebo

GLuint cae::Mesh::ebo = 0
Definition at line 23 of file OPGL.hpp.

42.40.2.2 vao

GLuint cae::Mesh::vao = 0
Definition at line 21 of file OPGL.hpp.

42.40.2.3 vbo

GLuint cae::Mesh::vbo = 0
Definition at line 22 of file OPGL.hpp.

42.40.2.4 vertexCount

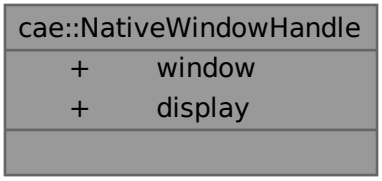
GLsizei cae::Mesh::vertexCount = 0
Definition at line 24 of file OPGL.hpp.
Referenced by cae::OPGL::createMesh().
The documentation for this struct was generated from the following file:

- plugins/Renderer/OpenGL/include/OPGL/OPGL.hpp

42.41 cae::NativeWindowHandle Struct Reference

Struct for native window handle.
#include <IWindow.hpp>

Collaboration diagram for cae::NativeWindowHandle:



Public Attributes

- void * window
- void * display

42.41.1 Detailed Description

Struct for native window handle.
Definition at line 33 of file IWindow.hpp.

42.41.2 Member Data Documentation

42.41.2.1 display

void* cae::NativeWindowHandle::display
Definition at line 36 of file IWindow.hpp.

42.41.2.2 window

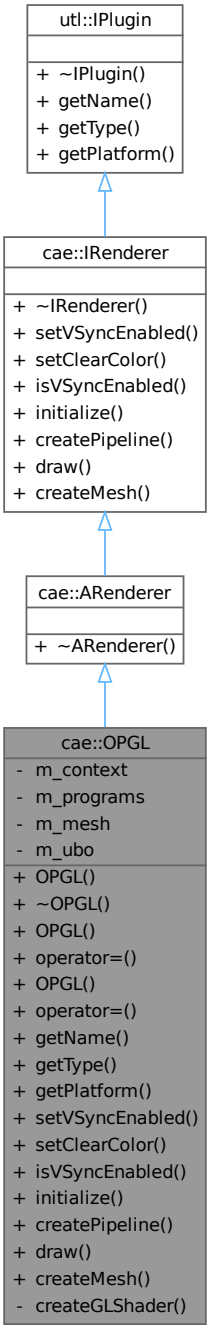
void* cae::NativeWindowHandle::window
Definition at line 35 of file IWindow.hpp.
Referenced by cae::GLFW::getNativeHandle().
The documentation for this struct was generated from the following file:

- modules/Interfaces/include/Interfaces/Window/IWindow.hpp

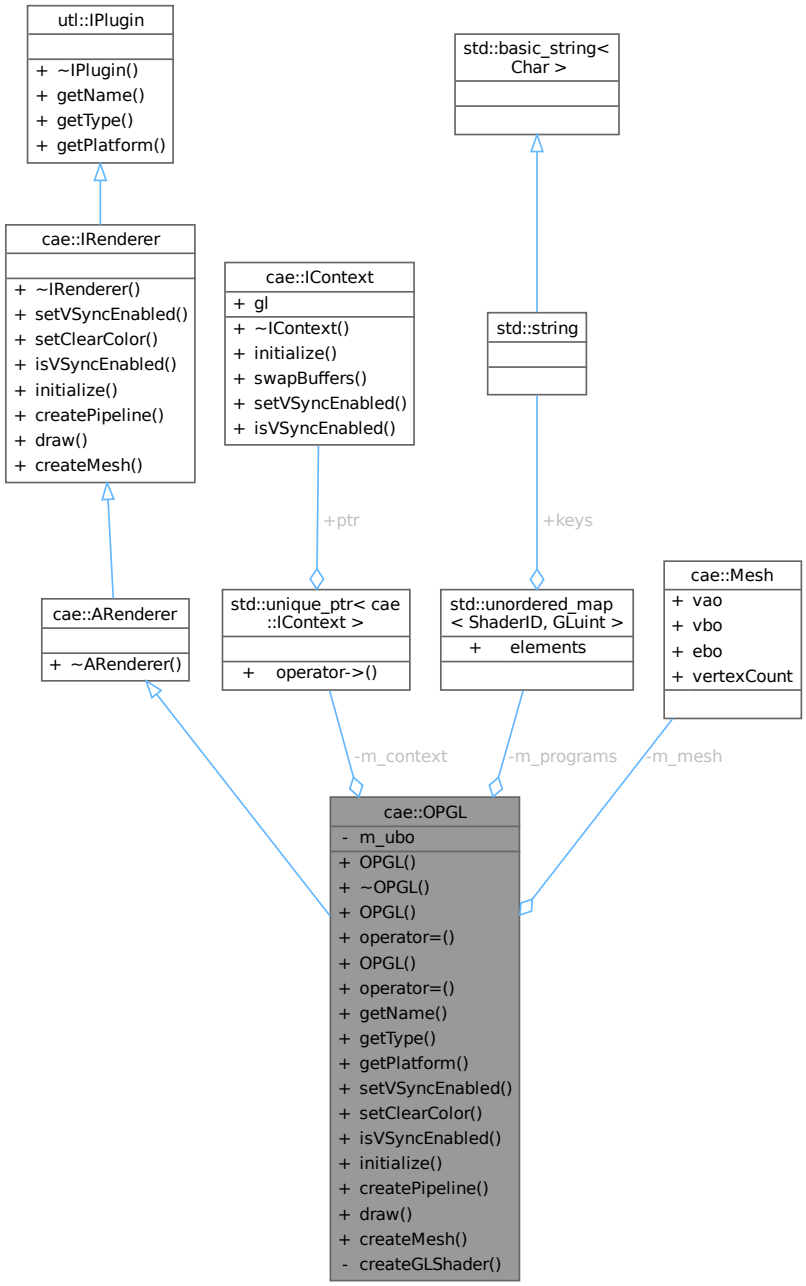
42.42 cae::OPGL Class Reference

Class for the OpenGL plugin.
#include <OPGL.hpp>

Inheritance diagram for cae::OPGL:



Collaboration diagram for cae::OPGL:



Public Member Functions

- OPGL ()=default
- ~OPGL () override=default
- OPGL (const OPGL &)=delete
- OPGL & operator= (const OPGL &)=delete
- OPGL (OPGL &&)=delete
- OPGL & operator= (OPGL &&)=delete
- std::string getName () const override

  Get the name of the plugin.

- utl::PluginType getType () const override

  Get the type of the plugin.
- utl::PluginPlatform getPlatform () const override

  Get the handled platform of the plugin.
- void setVSyncEnabled (const bool enabled) override

  Enable or disable VSync.
- void setClearColor (const Color &color) override

  Set the clear color.
- bool isVSyncEnabled () const override

  Check if VSync is enabled.
- void initialize (const NativeWindowHandle &nativeWindowHandle, const Color &clearColor) override

  Initialize the renderer with a native window handle and clear color.
- void createPipeline (const ShaderID &id, const ShaderIRModule &vertex, const ShaderIRModule &fragment) override

  Create a rendering pipeline with vertex and fragment shaders.
- void draw (const WindowSize &windowSize, const ShaderID &shaderId, glm::mat4 mvp) override

  Draw the scene using the specified shader and window size.
- void createMesh (const std::vector< float > &vertices) override

  Create a mesh with the given vertex data.

Public Member Functions inherited from cae::ARenderer

- ~ARenderer () override=default

Public Member Functions inherited from cae::IRenderer

- ~IRenderer () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default

Static Private Member Functions

- static GLuint createGLShader (GLenum type, const ShaderIRModule &data, const GladGLContext &gl)

Private Attributes

- std::unique_ptr< IContext > m_context
- std::unordered_map< ShaderID, GLuint > m_programs
- Mesh m_mesh
- GLuint m_ubo

42.42.1 Detailed Description

Class for the OpenGL plugin.
Definition at line 32 of file OPGL.hpp.

42.42.2 Constructor & Destructor Documentation

42.42.2.1 OPGL() [1/3]

cae::OPGL::OPGL () [default]

42.42.2.2 ~OPGL()

cae::OPGL::~OPGL () [override], [default]

42.42.2.3 OPGL() [2/3]

cae::OPGL::OPGL (
const OPGL &) [delete]

42.42.2.4 OPGL() [3/3]

cae::OPGL::OPGL (
OPGL &&) [delete]

42.42.3 Member Function Documentation

42.42.3.1 createGLShader()

GLuint cae::OPGL::createGLShader (
GLenum type,
const ShaderIRModule & data,
const GladGLContext & gl) [static], [private]

Definition at line 108 of file opgl.cpp.
References cae::ShaderIRModule::entryPoint, and cae::ShaderIRModule::spirv.

42.42.3.2 createMesh()

void cae::OPGL::createMesh (
const std::vector< float > & vertices) [override], [virtual]

Create a mesh with the given vertex data.

Parameters

vertices	Vertex data to create the mesh

Implements cae::IRenderer.
Definition at line 83 of file opgl.cpp.
References cae::Mesh::vertexCount.

42.42.3.3 createPipeline()

void cae::OPGL::createPipeline (
const ShaderID & id,
const ShaderIRModule & vertex,
const ShaderIRModule & fragment) [override], [virtual]

Create a rendering pipeline with vertex and fragment shaders.

Parameters

id	Shader ID
vertex	Vertex shader IR module
fragment	Fragment shader IR module

Implements cae::IRenderer.
Definition at line 56 of file opgl.cpp.

42.42.3.4 draw()

void cae::OPGL::draw (
const WindowSize & windowSize,
const ShaderID & shaderId,
glm::mat4 mvp) [override], [virtual]

Draw the scene using the specified shader and window size.

Parameters

windowSize	Current window size
shaderId	Shader ID to use for drawing
mvp	Model-View-Projection matrix

Implements cae::IRenderer.
Definition at line 38 of file opgl.cpp.
References cae::WindowSize::height, and cae::WindowSize::width.

42.42.3.5 getName()

std::string cae::OPGL::getName () const [inline], [nodiscard], [override], [virtual]

Get the name of the plugin.

Returns

The name of the plugin

Implements utl::IPlugin.
Definition at line 43 of file OPGL.hpp.

42.42.3.6 getPlatform()

utl::PluginPlatform cae::OPGL::getPlatform () const [inline], [nodiscard], [override], [virtual]

Get the handled platform of the plugin.

Returns

The handled platform of the plugin

Implements utl::IPlugin.
Definition at line 45 of file OPGL.hpp.
References utl::ALL.

42.42.3.7 getType()

utl::PluginType cae::OPGL::getType () const [inline], [nodiscard], [override], [virtual]

Get the type of the plugin.

Returns

The type of the plugin

Implements utl::IPlugin.
Definition at line 44 of file OPGL.hpp.
References utl::RENDERER.

42.42.3.8 initialize()

void cae::OPGL::initialize (
const NativeWindowHandle & nativeWindowHandle,
const Color & clearColor) [override], [virtual]

Initialize the renderer with a native window handle and clear color.

Parameters

nativeWindowHandle	Native window handle
clearColor	Clear color (default: white)

Implements cae::IRenderer.
Definition at line 15 of file opgl.cpp.
References cae::Color::a, cae::Color::b, cae::Color::g, m_context, m_ubo, and cae::Color::r.

42.42.3.9 isVSyncEnabled()

bool cae::OPGL::isVSyncEnabled () const [inline], [nodiscard], [override], [virtual]
Check if VSync is enabled.

Returns

Whether VSync is enabled

Implements cae::IRenderer.
Definition at line 54 of file OPGL.hpp.
References m_context.

42.42.3.10 operator=() [1/2]

OPGL & cae::OPGL::operator= (
const OPGL &) [delete]

42.42.3.11 operator=() [2/2]

OPGL & cae::OPGL::operator= (
OPGL &&) [delete]

42.42.3.12 setClearColor()

void cae::OPGL::setClearColor (
const Color & color) [inline], [override], [virtual]
Set the clear color.

Parameters

color	Clear color to set

Implements cae::IRenderer.
Definition at line 48 of file OPGL.hpp.
References cae::Color::a, cae::Color::b, cae::Color::g, m_context, and cae::Color::r.

42.42.3.13 setVSyncEnabled()

void cae::OPGL::setVSyncEnabled (
const bool enabled) [inline], [override], [virtual]
Enable or disable VSync.

Parameters

enabled	Whether VSync is enabled

Implements cae::IRenderer.
Definition at line 47 of file OPGL.hpp.
References m_context.

42.42.4 Member Data Documentation

42.42.4.1 m_context

std::unique_ptr<IContext> cae::OPGL::m_context [private]
Definition at line 63 of file OPGL.hpp.
Referenced by initialize(), isVSyncEnabled(), setClearColor(), and setVSyncEnabled().

42.42.4.2 m_mesh

Mesh cae::OPGL::m_mesh [private]
Definition at line 65 of file OPGL.hpp.

42.42.4.3 m_programs

std::unordered_map<ShaderID, GLuint> cae::OPGL::m_programs [private]
Definition at line 64 of file OPGL.hpp.

42.42.4.4 m_ubo

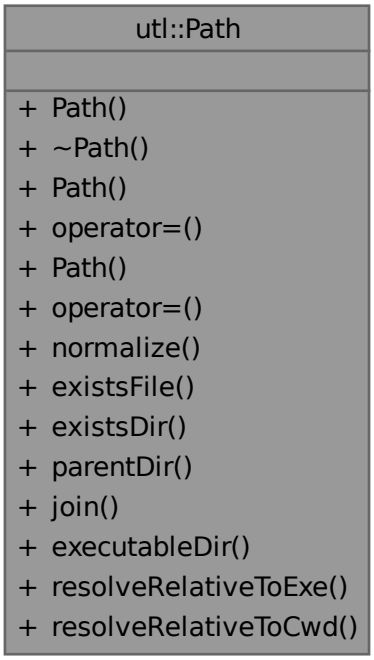
GLuint cae::OPGL::m_ubo [private]
Definition at line 67 of file OPGL.hpp.
Referenced by initialize().
The documentation for this class was generated from the following files:

- plugins/Renderer/OpenGL/include/OPGL/OPGL.hpp
- plugins/Renderer/OpenGL/src/opgl.cpp

42.43 utl::Path Class Reference

Class for path resolution utilities.
#include <Path.hpp>

Collaboration diagram for utl::Path:



Public Member Functions

- Path ()=default
- ~Path ()=default
- Path (const Path &)=delete
- Path & operator= (const Path &)=delete
- Path (Path &&)=delete
- Path & operator= (Path &&)=delete

Static Public Member Functions

- static fs::path normalize (const fs::path &path)

  Normalize a path (resolve symlinks, relative paths, etc.)
- static bool existsFile (const fs::path &path)

  Check if a file exists.
- static bool existsDir (const fs::path &path)

  Check if a directory exists.
- static fs::path parentDir (const fs::path &path)

  Get the parent directory of a path.
- template<typename... Paths>
  static fs::path join (const Paths &...paths)

  Join multiple paths.
- static fs::path executableDir ()

  Get the directory of the executable.
- static fs::path resolveRelativeToExe (const fs::path &relativePath)

  Resolve a relative path to the executable directory.
- static fs::path resolveRelativeToCwd (const fs::path &relativePath)

42.43.1 Detailed Description

Class for path resolution utilities.
Definition at line 29 of file Path.hpp.

42.43.2 Constructor & Destructor Documentation

42.43.2.1 Path() [1/3]

utl::Path::Path () [default]

42.43.2.2 ~Path()

utl::Path::~Path () [default]

42.43.2.3 Path() [2/3]

utl::Path::Path (
const Path &) [delete]

42.43.2.4 Path() [3/3]

utl::Path::Path (
Path &&) [delete]

42.43.3 Member Function Documentation

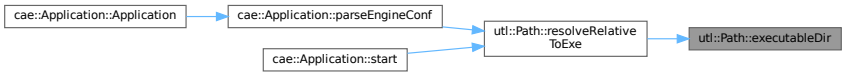
42.43.3.1 executableDir()

static fs::path utl::Path::executableDir () [inline], [static]
Get the directory of the executable.

Returns

Directory of the executable

Definition at line 92 of file Path.hpp.
Referenced by resolveRelativeToExe().
Here is the caller graph for this function:



42.43.3.2 existsDir()

static bool utl::Path::existsDir (
const fs::path & path) [inline], [static]
Check if a directory exists.

Parameters

path	Path to be checked

Returns

True if the directory exists

Definition at line 68 of file Path.hpp.
Referenced by loadPlugins().
Here is the caller graph for this function:

42.43.3.3 existsFile()

static bool utl::Path::existsFile (
const fs::path & path) [inline], [static]
Check if a file exists.

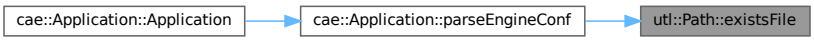
Parameters

path	Path to be checked

Returns

True if the file exists

Definition at line 61 of file Path.hpp.
Referenced by cae::Application::parseEngineConf().
Here is the caller graph for this function:



42.43.3.4 join()

template<typename... Paths>
static fs::path utl::Path::join (
const Paths &... paths) [inline], [static]
Join multiple paths.

Template Parameters

Paths	Variadic template for paths

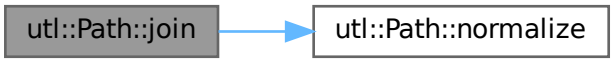
Parameters

paths	Paths to be joined

Returns

Joined path

Definition at line 83 of file Path.hpp.
References normalize().
Here is the call graph for this function:



42.43.3.5 normalize()

static fs::path utl::Path::normalize (
const fs::path & path) [inline], [static]
Normalize a path (resolve symlinks, relative paths, etc.)

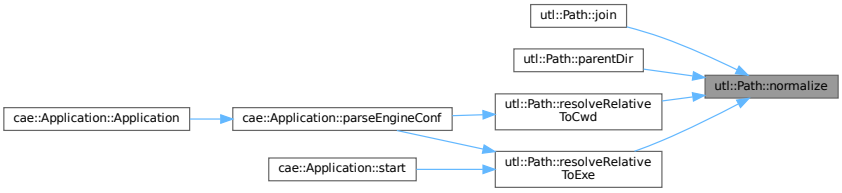
Parameters

path	Path to be normalized

Returns

Normalized path

Definition at line 44 of file Path.hpp.
Referenced by join(), parentDir(), resolveRelativeToCwd(), and resolveRelativeToExe().
Here is the caller graph for this function:



42.43.3.6 operator=() [1/2]

Path & utl::Path::operator= (
const Path &) [delete]

42.43.3.7 operator=() [2/2]

Path & utl::Path::operator= (
Path &&) [delete]

42.43.3.8 parentDir()

static fs::path utl::Path::parentDir (
const fs::path & path) [inline], [static]

Get the parent directory of a path.

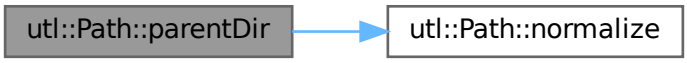
Parameters

path	Path to get the parent directory of

Returns

Parent directory of the path

Definition at line 75 of file Path.hpp.
References normalize().
Here is the call graph for this function:



42.43.3.9 resolveRelativeToCwd()

static fs::path utl::Path::resolveRelativeToCwd (
const fs::path & relativePath) [inline], [static]

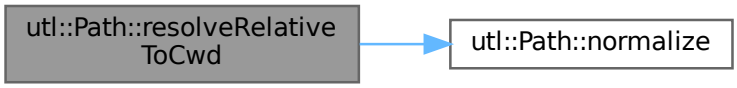
Parameters

relativePath	Relative path to be resolved

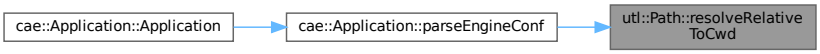
Returns

Resolved path relative to the user cwd

Definition at line 131 of file Path.hpp.
References normalize().
Referenced by cae::Application::parseEngineConf().
Here is the call graph for this function:



Here is the caller graph for this function:



42.43.3.10 resolveRelativeToExe()

static fs::path utl::Path::resolveRelativeToExe (
const fs::path & relativePath) [inline], [static]

Resolve a relative path to the executable directory.

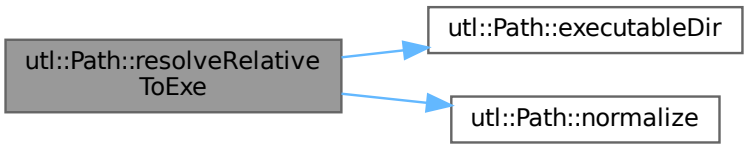
Parameters

relativePath	Relative path to be resolved

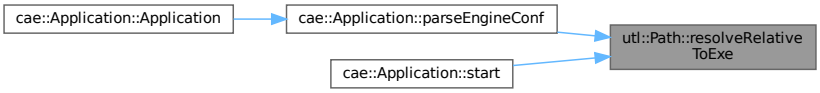
Returns

Resolved path relative to the executable directory

Definition at line 121 of file Path.hpp.
References executableDir(), and normalize().
Referenced by cae::Application::parseEngineConf(), and cae::Application::start().
Here is the call graph for this function:



Here is the caller graph for this function:



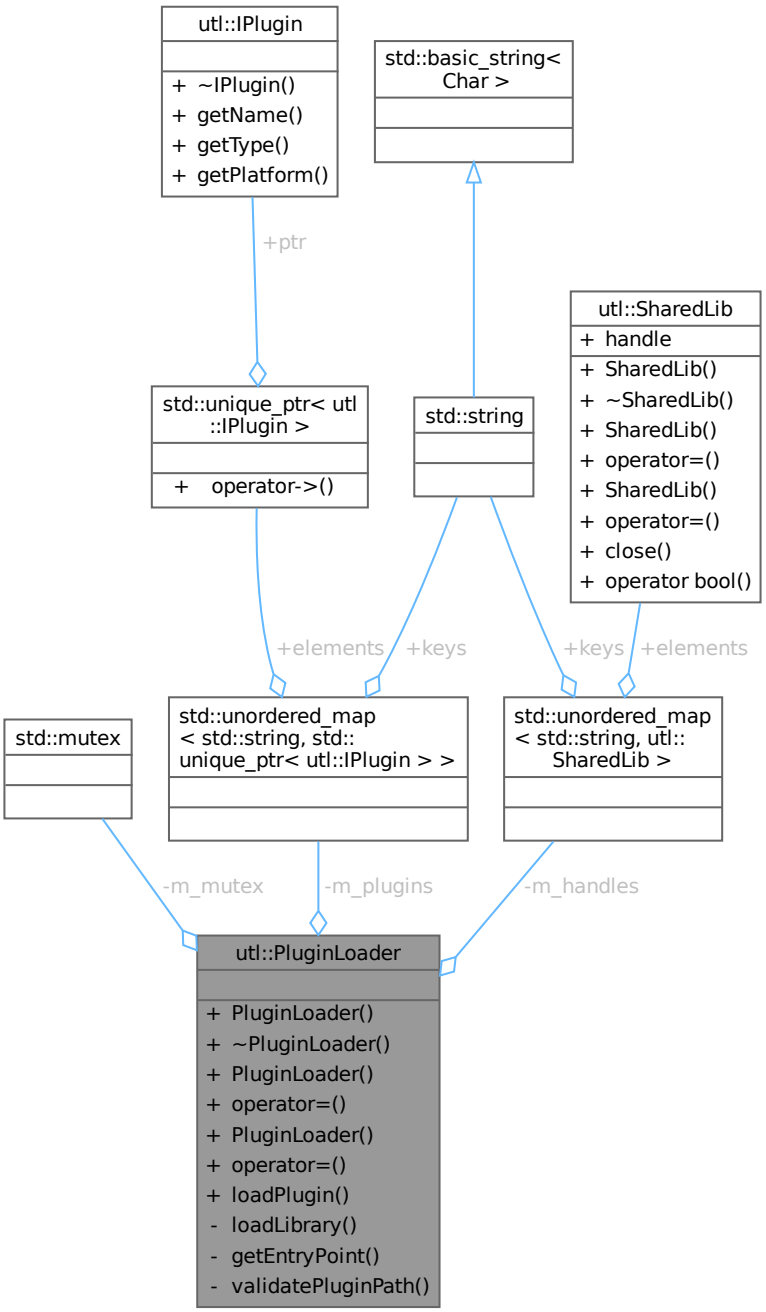
The documentation for this class was generated from the following file:

- modules/Utils/include/Utils/Path.hpp

42.44 utl::PluginLoader Class Reference

Modern, type-safe plugin loader.

#include <PluginLoader.hpp>
Collaboration diagram for utl::PluginLoader:



Public Member Functions

- PluginLoader ()=default
- ~PluginLoader ()=default
- PluginLoader (const PluginLoader &)=delete
- PluginLoader & operator= (const PluginLoader &)=delete
- PluginLoader (PluginLoader &&)=delete
- PluginLoader & operator= (PluginLoader &&)=delete

- template<std::derived_from< IPlugin > T>
  std::shared_ptr< T > loadPlugin (const std::string &path, const std::string_view &plugin↩
  Prefix="")

  Load a plugin of type T.

Static Private Member Functions

- static SharedLib loadLibrary (const std::string &path)

  Load a shared library.
- static EntryPointFn getEntryPoint (SharedLib &lib, const std::string &path)
- static void validatePluginPath (const std::filesystem::path &path, const std::string_view &plugin↩
  Prefix)

  Validate the plugin path.

Private Attributes

- std::mutex m_mutex
- std::unordered_map< std::string, SharedLib > m_handles
- std::unordered_map< std::string, std::unique_ptr< IPlugin > > m_plugins

42.44.1 Detailed Description

Modern, type-safe plugin loader.
Definition at line 86 of file PluginLoader.hpp.

42.44.2 Constructor & Destructor Documentation

42.44.2.1 PluginLoader() [1/3]

utl::PluginLoader::PluginLoader () [default]

42.44.2.2 ~PluginLoader()

utl::PluginLoader::~PluginLoader () [default]

42.44.2.3 PluginLoader() [2/3]

utl::PluginLoader::PluginLoader (
const PluginLoader &) [delete]

42.44.2.4 PluginLoader() [3/3]

utl::PluginLoader::PluginLoader (
PluginLoader &&) [delete]

42.44.3 Member Function Documentation

42.44.3.1 getEntryPoint()

static EntryPointFn utl::PluginLoader::getEntryPoint (
SharedLib & lib,
const std::string & path) [inline], [static], [private]

Parameters

lib	Loaded SharedLib
path	Path to the dynamic library

Returns

EntryPoint function pointer

Definition at line 186 of file PluginLoader.hpp.
References utl::SharedLib::handle.
Referenced by loadPlugin().
Here is the caller graph for this function:

42.44.3.2 loadLibrary()

static SharedLib utl::PluginLoader::loadLibrary (
const std::string & path) [inline], [static], [private]

Load a shared library.

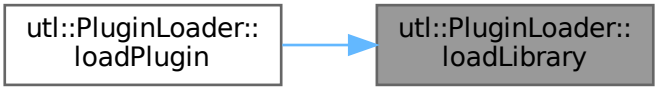
Parameters

path	Path to the dynamic library

Returns

Loaded SharedLib

Definition at line 160 of file PluginLoader.hpp.
Referenced by loadPlugin().
Here is the caller graph for this function:



42.44.3.3 loadPlugin()

template<std::derived_from< IPlugin > T>
std::shared_ptr< T > utl::PluginLoader::loadPlugin (
const std::string & path,
const std::string_view & pluginPrefix = "") [inline]

Load a plugin of type T.

Template Parameters

T	Expected plugin interface (must derive from IPlugin)

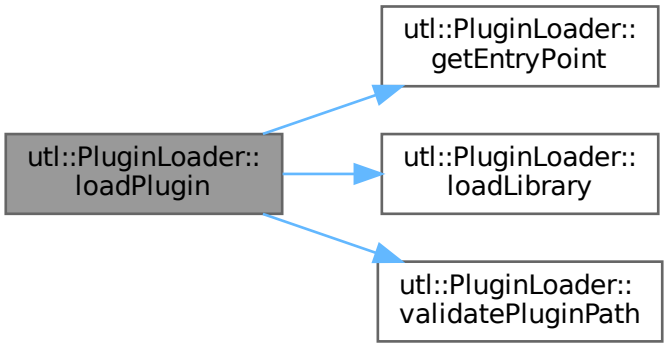
Parameters

path	Path to the dynamic library
pluginPrefix	Expected prefix for plugin filenames

Returns

shared_ptr<T> instance

Definition at line 105 of file PluginLoader.hpp.
References getEntryPoint(), loadLibrary(), m_handles, m_mutex, m_plugins, PLUGINS_PREFIX, and validatePluginPath().
Here is the call graph for this function:



42.44.3.4 operator=() [1/2]

PluginLoader & utl::PluginLoader::operator= (
const PluginLoader &) [delete]

42.44.3.5 operator=() [2/2]

PluginLoader & utl::PluginLoader::operator= (
PluginLoader &&) [delete]

42.44.3.6 validatePluginPath()

static void utl::PluginLoader::validatePluginPath (
const std::filesystem::path & path,
const std::string_view & pluginPrefix) [inline], [static], [private]

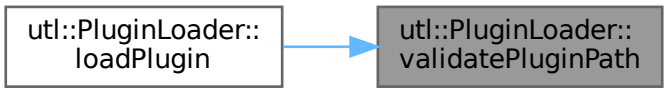
Validate the plugin path.

Parameters

path	Path to the dynamic library

pluginPrefix	Expected prefix for plugin filenames

Definition at line 206 of file PluginLoader.hpp.
Referenced by loadPlugin().
Here is the caller graph for this function:



42.44.4 Member Data Documentation

42.44.4.1 m_handles

std::unordered_map<std::string, SharedLib> utl::PluginLoader::m_handles [private]
Definition at line 152 of file PluginLoader.hpp.
Referenced by loadPlugin().

42.44.4.2 m_mutex

std::mutex utl::PluginLoader::m_mutex [private]
Definition at line 151 of file PluginLoader.hpp.
Referenced by loadPlugin().

42.44.4.3 m_plugins

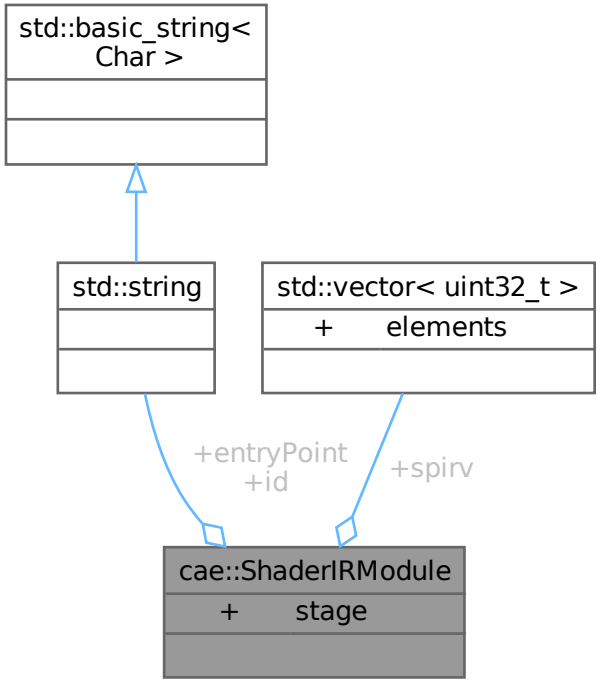
std::unordered_map<std::string, std::unique_ptr<IPlugin> > utl::PluginLoader::m_plugins [private]
Definition at line 153 of file PluginLoader.hpp.
Referenced by loadPlugin().
The documentation for this class was generated from the following file:

- modules/Utils/include/Utils/PluginLoader.hpp

42.45 cae::ShaderIRModule Struct Reference

Struct for shader intermediate representation module.
#include <IShaderFrontend.hpp>

Collaboration diagram for cae::ShaderIRModule:



Public Attributes

- ShaderID id
- ShaderStage stage
- std::vector< uint32_t > spirv
- std::string entryPoint = "main"

42.45.1 Detailed Description

Struct for shader intermediate representation module.
Definition at line 55 of file IShaderFrontend.hpp.

42.45.2 Member Data Documentation

42.45.2.1 entryPoint

std::string cae::ShaderIRModule::entryPoint = "main"
Definition at line 60 of file IShaderFrontend.hpp.
Referenced by cae::GLSL::compile(), and cae::OPGL::createGLShader().

42.45.2.2 id

ShaderID cae::ShaderIRModule::id
Definition at line 57 of file IShaderFrontend.hpp.
Referenced by cae::GLSL::compile().

42.45.2.3 spirv

std::vector<uint32_t> cae::ShaderIRModule::spirv
Definition at line 59 of file IShaderFrontend.hpp.
Referenced by cae::GLSL::compile(), and cae::OPGL::createGLShader().

42.45.2.4 stage

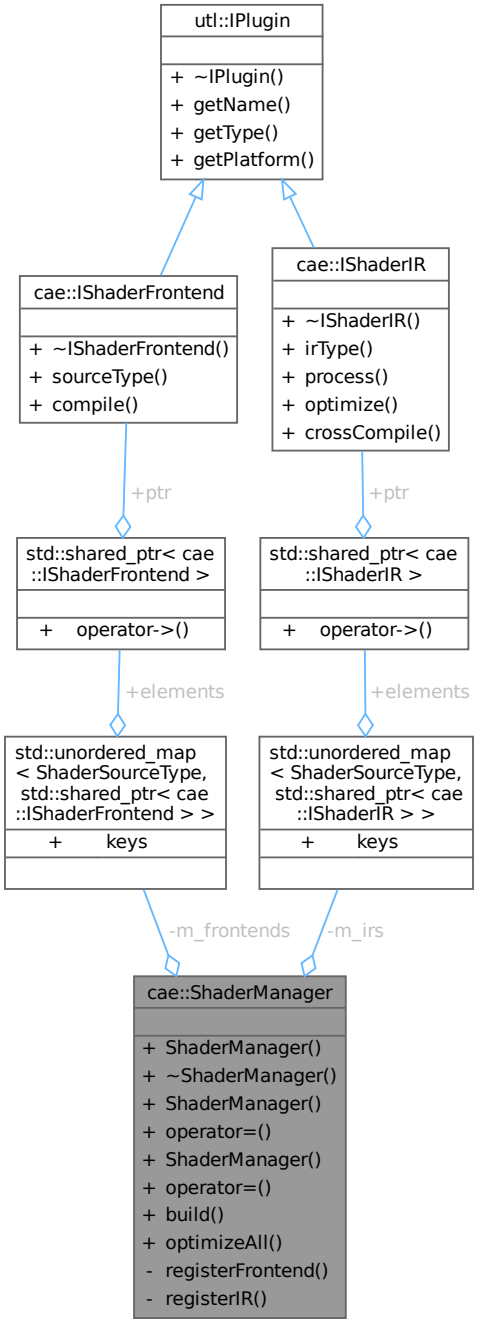
ShaderStage cae::ShaderIRModule::stage
Definition at line 58 of file IShaderFrontend.hpp.
Referenced by cae::GLSL::compile().
The documentation for this struct was generated from the following file:

- modules/Interfaces/include/Interfaces/Shader/Frontend/IShaderFrontend.hpp

42.46 cae::ShaderManager Class Reference

Class for managing shaders.
#include <ShaderManager.hpp>

Collaboration diagram for cae::ShaderManager:



Public Member Functions

- ShaderManager (const std::vector< std::function< std::shared_ptr< IShaderFrontend >()> > &shaderFrontendFactories, const std::function< std::shared_ptr< IShaderIR >()> &shader↩ IRFactory=nullptr)
- ~ShaderManager ()=default
- ShaderManager (const ShaderManager &)=delete
- ShaderManager & operator= (const ShaderManager &)=delete
- ShaderManager (ShaderManager &&)=delete

- ShaderManager & operator= (ShaderManager &&)=delete
- std::unordered_map< ShaderID, ShaderIRModule > build (const std::vector< ShaderSourceDesc > &sources, const ShaderSourceType targetIR) const
- template<std::ranges::input_range R>
  void optimizeAll (const ShaderSourceType irType, R &&modules) const

Private Member Functions

- void registerFrontend (const std::shared_ptr< IShaderFrontend > &f)
- void registerIR (const std::shared_ptr< IShaderIR > &ir)

Private Attributes

- std::unordered_map< ShaderSourceType, std::shared_ptr< IShaderFrontend > > m_frontends
- std::unordered_map< ShaderSourceType, std::shared_ptr< IShaderIR > > m_irs

42.46.1 Detailed Description

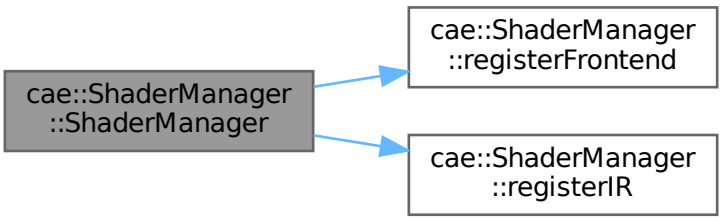
Class for managing shaders.
Definition at line 20 of file ShaderManager.hpp.

42.46.2 Constructor & Destructor Documentation

42.46.2.1 ShaderManager() [1/3]

cae::ShaderManager::ShaderManager (
const std::vector< std::function< std::shared_ptr< IShaderFrontend >()> > & shaderFrontendFactories,
const std::function< std::shared_ptr< IShaderIR >()> & shaderIRFactory = nullptr) [inline], [explicit]

Definition at line 23 of file ShaderManager.hpp.
References registerFrontend(), and registerIR().
Here is the call graph for this function:



42.46.2.2 ~ShaderManager()

cae::ShaderManager::~ShaderManager () [default]

42.46.2.3 ShaderManager() [2/3]

cae::ShaderManager::ShaderManager (
const ShaderManager &) [delete]

42.46.2.4 ShaderManager() [3/3]

cae::ShaderManager::ShaderManager (
ShaderManager &&) [delete]

42.46.3 Member Function Documentation

42.46.3.1 build()

std::unordered_map< ShaderID, ShaderIRModule > cae::ShaderManager::build (
const std::vector< ShaderSourceDesc > & sources,
const ShaderSourceType targetIR) const [inline]

Definition at line 44 of file ShaderManager.hpp.
References m_frontends, and m_irs.

42.46.3.2 operator=() [1/2]

ShaderManager & cae::ShaderManager::operator= (
const ShaderManager &) [delete]

42.46.3.3 operator=() [2/2]

ShaderManager & cae::ShaderManager::operator= (
ShaderManager &&) [delete]

42.46.3.4 optimizeAll()

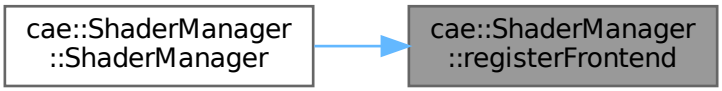
template<std::ranges::input_range R>
void cae::ShaderManager::optimizeAll (
const ShaderSourceType irType,
R && modules) const [inline]

Definition at line 61 of file ShaderManager.hpp.
References m_irs.

42.46.3.5 registerFrontend()

void cae::ShaderManager::registerFrontend (
const std::shared_ptr< IShaderFrontend > & f) [inline], [private]

Definition at line 88 of file ShaderManager.hpp.
References m_frontends.
Referenced by ShaderManager().
Here is the caller graph for this function:

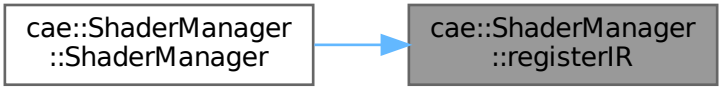


42.46.3.6 registerIR()

void cae::ShaderManager::registerIR (
const std::shared_ptr< IShaderIR > & ir) [inline], [private]

Definition at line 89 of file ShaderManager.hpp.
References m_irs.
Referenced by ShaderManager().

Here is the caller graph for this function:



42.46.4 Member Data Documentation

42.46.4.1 m_frontends

std::unordered_map<ShaderSourceType, std::shared_ptr<IShaderFrontend> > cae::ShaderManager::m_frontends [private]
Definition at line 91 of file ShaderManager.hpp.
Referenced by build(), and registerFrontend().

42.46.4.2 m_irs

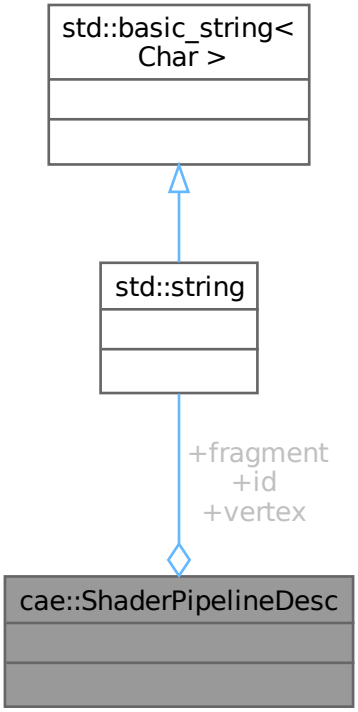
std::unordered_map<ShaderSourceType, std::shared_ptr<IShaderIR> > cae::ShaderManager::m_irs [private]
Definition at line 92 of file ShaderManager.hpp.
Referenced by build(), optimizeAll(), and registerIR().
The documentation for this class was generated from the following file:

- modules/Engine/include/Engine/ShaderManager.hpp

42.47 cae::ShaderPipelineDesc Struct Reference

Struct for shader pipeline description.
#include <IShaderFrontend.hpp>

Collaboration diagram for cae::ShaderPipelineDesc:



Public Attributes

- ShaderID id
- ShaderID vertex
- ShaderID fragment

42.47.1 Detailed Description

Struct for shader pipeline description.
Definition at line 68 of file IShaderFrontend.hpp.

42.47.2 Member Data Documentation

42.47.2.1 fragment

ShaderID cae::ShaderPipelineDesc::fragment
Definition at line 72 of file IShaderFrontend.hpp.

42.47.2.2 id

ShaderID cae::ShaderPipelineDesc::id
Definition at line 70 of file IShaderFrontend.hpp.

42.47.2.3 vertex

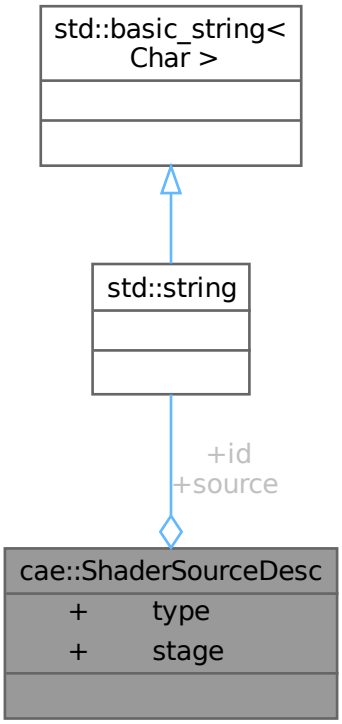
ShaderID cae::ShaderPipelineDesc::vertex
Definition at line 71 of file IShaderFrontend.hpp.

The documentation for this struct was generated from the following file:

- modules/Interfaces/include/Interfaces/Shader/Frontend/IShaderFrontend.hpp

42.48 cae::ShaderSourceDesc Struct Reference

Struct for shader source description.
#include <IShaderFrontend.hpp>
Collaboration diagram for cae::ShaderSourceDesc:



Public Attributes

- ShaderID id
- ShaderSourceType type
- std::string source
- ShaderStage stage

42.48.1 Detailed Description

Struct for shader source description.
Definition at line 42 of file IShaderFrontend.hpp.

42.48.2 Member Data Documentation

42.48.2.1 id

ShaderID cae::ShaderSourceDesc::id
Definition at line 44 of file IShaderFrontend.hpp.
Referenced by cae::GLSL::compile().

42.48.2.2 source

std::string cae::ShaderSourceDesc::source
Definition at line 46 of file IShaderFrontend.hpp.
Referenced by cae::GLSL::compile().

42.48.2.3 stage

ShaderStage cae::ShaderSourceDesc::stage
Definition at line 47 of file IShaderFrontend.hpp.
Referenced by cae::GLSL::compile().

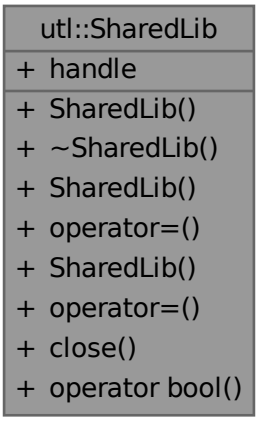
42.48.2.4 type

ShaderSourceType cae::ShaderSourceDesc::type
Definition at line 45 of file IShaderFrontend.hpp.
The documentation for this struct was generated from the following file:

- modules/Interfaces/include/Interfaces/Shader/Frontend/IShaderFrontend.hpp

42.49 utl::SharedLib Struct Reference

RAII wrapper for shared libraries.
#include <PluginLoader.hpp>
Collaboration diagram for utl::SharedLib:



Public Member Functions

- SharedLib (const LibHandle h=nullptr)
- ~SharedLib ()
- SharedLib (const SharedLib &)=delete
- SharedLib & operator= (const SharedLib &)=delete
- SharedLib (SharedLib &&other) noexcept
- SharedLib & operator= (SharedLib &&other) noexcept
- void close ()
- operator bool () const

Public Attributes

- LibHandle handle = nullptr

42.49.1 Detailed Description

RAII wrapper for shared libraries.
Definition at line 41 of file PluginLoader.hpp.

42.49.2 Constructor & Destructor Documentation

42.49.2.1 SharedLib() [1/3]

utl::SharedLib::SharedLib (
const LibHandle h = nullptr) [inline], [explicit]
Definition at line 45 of file PluginLoader.hpp.

42.49.2.2 ~SharedLib()

utl::SharedLib::~SharedLib () [inline]
Definition at line 46 of file PluginLoader.hpp.
References close().
Here is the call graph for this function:

42.49.2.3 SharedLib() [2/3]

utl::SharedLib::SharedLib (
const SharedLib &) [delete]

42.49.2.4 SharedLib() [3/3]

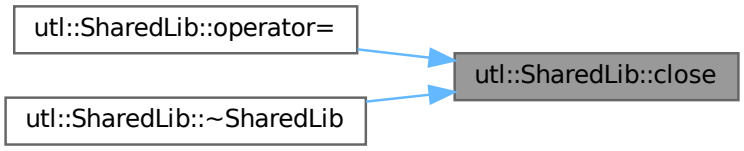
utl::SharedLib::SharedLib (
SharedLib && other) [inline], [noexcept]
Definition at line 50 of file PluginLoader.hpp.

42.49.3 Member Function Documentation

42.49.3.1 close()

void utl::SharedLib::close () [inline]
Definition at line 62 of file PluginLoader.hpp.
References handle.
Referenced by operator=(), and ~SharedLib().

Here is the caller graph for this function:



42.49.3.2 operator bool()

utl::SharedLib::operator bool () const [inline], [explicit]
Definition at line 76 of file PluginLoader.hpp.
References handle.

42.49.3.3 operator=() [1/2]

SharedLib & utl::SharedLib::operator= (
const SharedLib &) [delete]

42.49.3.4 operator=() [2/2]

SharedLib & utl::SharedLib::operator= (
SharedLib && other) [inline], [noexcept]
Definition at line 51 of file PluginLoader.hpp.
References close(), and handle.
Here is the call graph for this function:

42.49.4 Member Data Documentation

42.49.4.1 handle

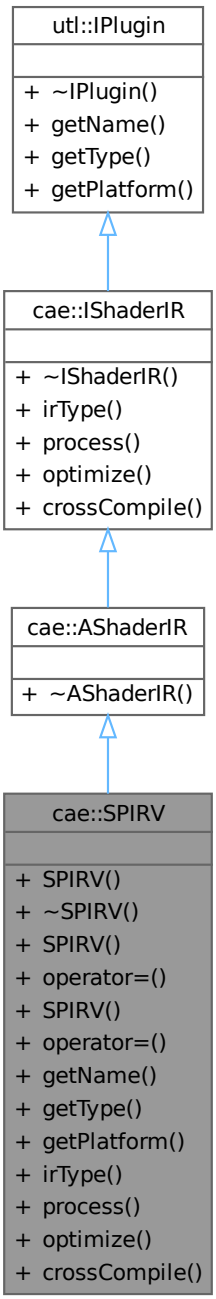
LibHandle utl::SharedLib::handle = nullptr
Definition at line 43 of file PluginLoader.hpp.
Referenced by close(), utl::PluginLoader::getEntryPoint(), operator bool(), and operator=().
The documentation for this struct was generated from the following file:

- modules/Utils/include/Utils/PluginLoader.hpp

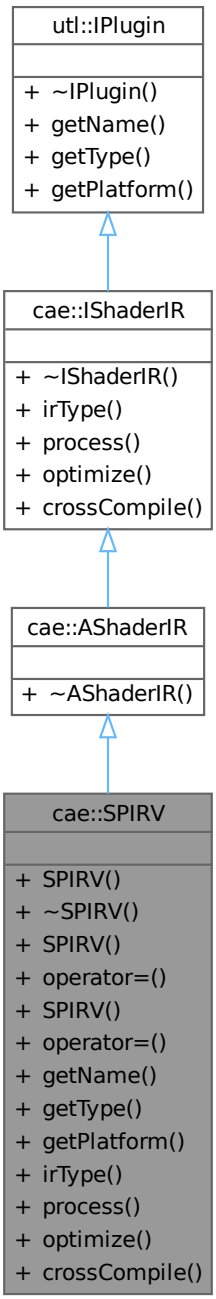
42.50 cae::SPIRV Class Reference

Class for the SPIR-V IR plugin.
#include <SPIRV.hpp>

Inheritance diagram for cae::SPIRV:



Collaboration diagram for cae::SPIRV:



Public Member Functions

- SPIRV ()=default
- ~SPIRV () override=default
- SPIRV (const SPIRV &)=delete
- SPIRV & operator= (const SPIRV &)=delete
- SPIRV (SPIRV &&)=delete
- SPIRV & operator= (SPIRV &&)=delete

- std::string getName () const override

  Get the name of the plugin.
- utl::PluginType getType () const override

  Get the type of the plugin.
- utl::PluginPlatform getPlatform () const override

  Get the handled platform of the plugin.
- ShaderSourceType irType () const override

  Get the IR type this processor handles.
- ShaderIRModule process (const ShaderIRModule &module) override

  Transform or validate a shader IR module.
- void optimize (std::span< ShaderIRModule > modules) override

  Optional: optimize a batch of IR modules.
- ShaderIRModule crossCompile (const ShaderIRModule &module, const ShaderSourceType targetType) override

  Optional: cross-compile from one IR to another (SPIR-V -> MSL, etc.)

Public Member Functions inherited from cae::AShaderIR

- ~AShaderIR () override=default

Public Member Functions inherited from cae::IShaderIR

- ~IShaderIR () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default

42.50.1 Detailed Description

Class for the SPIR-V IR plugin.
Definition at line 19 of file SPIRV.hpp.

42.50.2 Constructor & Destructor Documentation

42.50.2.1 SPIRV() [1/3]

cae::SPIRV::SPIRV () [default]

42.50.2.2 ~SPIRV()

cae::SPIRV::~SPIRV () [override], [default]

42.50.2.3 SPIRV() [2/3]

cae::SPIRV::SPIRV (
const SPIRV &) [delete]

42.50.2.4 SPIRV() [3/3]

cae::SPIRV::SPIRV (
SPIRV &&) [delete]

42.50.3 Member Function Documentation

42.50.3.1 crossCompile()

ShaderIRModule cae::SPIRV::crossCompile (
const ShaderIRModule & module,
const ShaderSourceType targetType) [inline], [override], [virtual]
Optional: cross-compile from one IR to another (SPIR-V -> MSL, etc.)
Implements cae::IShaderIR.
Definition at line 44 of file SPIRV.hpp.
References cae::MSL.

42.50.3.2 getName()

std::string cae::SPIRV::getName () const [inline], [nodiscard], [override], [virtual]
Get the name of the plugin.

Returns

The name of the plugin

Implements utl::IPlugin.
Definition at line 30 of file SPIRV.hpp.

42.50.3.3 getPlatform()

utl::PluginPlatform cae::SPIRV::getPlatform () const [inline], [nodiscard], [override], [virtual]
Get the handled platform of the plugin.

Returns

The handled platform of the plugin

Implements utl::IPlugin.
Definition at line 32 of file SPIRV.hpp.
References utl::ALL.

42.50.3.4 getType()

utl::PluginType cae::SPIRV::getType () const [inline], [nodiscard], [override], [virtual]
Get the type of the plugin.

Returns

The type of the plugin

Implements utl::IPlugin.
Definition at line 31 of file SPIRV.hpp.
References utl::SHADER_IR.

42.50.3.5 irType()

ShaderSourceType cae::SPIRV::irType () const [inline], [nodiscard], [override], [virtual]
Get the IR type this processor handles.

Returns

The IR type this processor handles

Implements cae::IShaderIR.
Definition at line 34 of file SPIRV.hpp.
References cae::SPIRV.

42.50.3.6 operator=() [1/2]

SPIRV & cae::SPIRV::operator= (
const SPIRV &) [delete]

42.50.3.7 operator=() [2/2]

SPIRV & cae::SPIRV::operator= (
SPIRV &&) [delete]

42.50.3.8 optimize()

void cae::SPIRV::optimize (
std::span< ShaderIRModule > modules) [inline], [override], [virtual]
Optional: optimize a batch of IR modules.
Implements cae::IShaderIR.
Definition at line 42 of file SPIRV.hpp.

42.50.3.9 process()

ShaderIRModule cae::SPIRV::process (
const ShaderIRModule & module) [inline], [override], [virtual]
Transform or validate a shader IR module.

Parameters

module	The input IR module

Returns

Transformed IR module ready for the backend

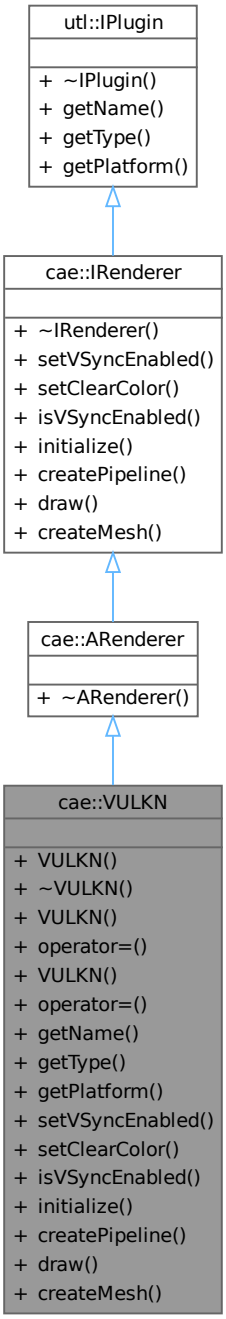
Implements cae::IShaderIR.
Definition at line 36 of file SPIRV.hpp.
The documentation for this class was generated from the following file:

- plugins/Shader/IR/SPIRV/include/SPIRV/SPIRV.hpp

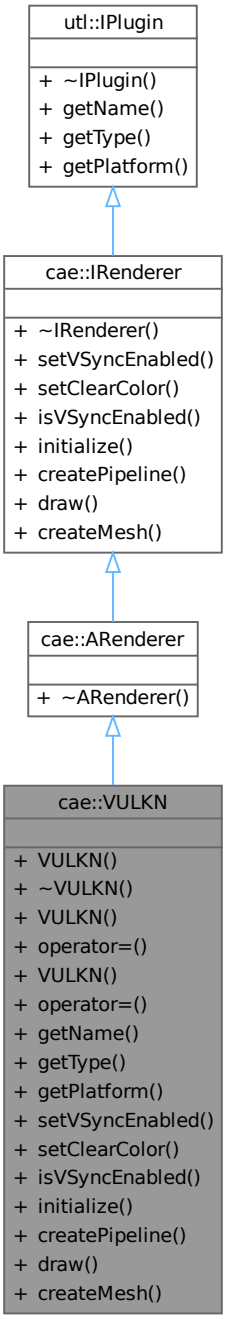
42.51 cae::VULKN Class Reference

Class for the Vulkan plugin.
#include <VULKN.hpp>

Inheritance diagram for cae::VULKN:



Collaboration diagram for cae::VULKN:



Public Member Functions

- VULKN ()=default
- ~VULKN () override=default
- VULKN (const VULKN &)=delete
- VULKN & operator= (const VULKN &)=delete
- VULKN (VULKN &&)=delete
- VULKN & operator= (VULKN &&)=delete

- std::string getName () const override

  Get the name of the plugin.
- utl::PluginType getType () const override

  Get the type of the plugin.
- utl::PluginPlatform getPlatform () const override

  Get the handled platform of the plugin.
- void setVSyncEnabled (bool enabled) override

  Enable or disable VSync.
- void setClearColor (const Color &color) override

  Set the clear color.
- bool isVSyncEnabled () const override

  Check if VSync is enabled.
- void initialize (const NativeWindowHandle &nativeWindowHandle, const Color &clearColor) override

  Initialize the renderer with a native window handle and clear color.
- void createPipeline (const ShaderID &id, const ShaderIRModule &vertex, const ShaderIRModule &fragment) override

  Create a rendering pipeline with vertex and fragment shaders.
- void draw (const WindowSize &windowSize, const ShaderID &shaderId, glm::mat4 mvp) override

  Draw the scene using the specified shader and window size.
- void createMesh (const std::vector< float > &vertices) override

  Create a mesh with the given vertex data.

Public Member Functions inherited from cae::ARenderer

- ~ARenderer () override=default

Public Member Functions inherited from cae::IRenderer

- ~IRenderer () override=default

Public Member Functions inherited from utl::IPlugin

- virtual ~IPlugin ()=default

42.51.1 Detailed Description

Class for the Vulkan plugin.
Definition at line 21 of file VULKN.hpp.

42.51.2 Constructor & Destructor Documentation

42.51.2.1 VULKN() [1/3]

cae::VULKN::VULKN () [default]

42.51.2.2 ~VULKN()

cae::VULKN::~VULKN () [override], [default]

42.51.2.3 VULKN() [2/3]

cae::VULKN::VULKN (
const VULKN &) [delete]

42.51.2.4 VULKN() [3/3]

cae::VULKN::VULKN (
VULKN &&) [delete]

42.51.3 Member Function Documentation

42.51.3.1 createMesh()

void cae::VULKN::createMesh (
const std::vector< float > & vertices) [inline], [override], [virtual]
Create a mesh with the given vertex data.

Parameters

vertices	Vertex data to create the mesh

Implements cae::IRenderer.
Definition at line 48 of file VULKN.hpp.

42.51.3.2 createPipeline()

void cae::VULKN::createPipeline (
const ShaderID & id,
const ShaderIRModule & vertex,
const ShaderIRModule & fragment) [inline], [override], [virtual]
Create a rendering pipeline with vertex and fragment shaders.

Parameters

id	Shader ID
vertex	Vertex shader IR module
fragment	Fragment shader IR module

Implements cae::IRenderer.
Definition at line 43 of file VULKN.hpp.

42.51.3.3 draw()

void cae::VULKN::draw (
const WindowSize & windowSize,
const ShaderID & shaderId,
glm::mat4 mvp) [inline], [override], [virtual]
Draw the scene using the specified shader and window size.

Parameters

windowSize	Current window size
shaderId	Shader ID to use for drawing
mvp	Model-View-Projection matrix

Implements cae::IRenderer.
Definition at line 47 of file VULKN.hpp.

42.51.3.4 getName()

std::string cae::VULKN::getName () const [inline], [nodiscard], [override], [virtual]
Get the name of the plugin.

Returns

The name of the plugin

Implements utl::IPlugin.
Definition at line 33 of file VULKN.hpp.

42.51.3.5 getPlatform()

utl::PluginPlatform cae::VULKN::getPlatform () const [inline], [nodiscard], [override], [virtual]
Get the handled platform of the plugin.

Returns

The handled platform of the plugin

Implements utl::IPlugin.
Definition at line 35 of file VULKN.hpp.
References utl::ALL.

42.51.3.6 getType()

utl::PluginType cae::VULKN::getType () const [inline], [nodiscard], [override], [virtual]
Get the type of the plugin.

Returns

The type of the plugin

Implements utl::IPlugin.
Definition at line 34 of file VULKN.hpp.
References utl::RENDERER.

42.51.3.7 initialize()

void cae::VULKN::initialize (
const NativeWindowHandle & nativeWindowHandle,
const Color & clearColor) [inline], [override], [virtual]
Initialize the renderer with a native window handle and clear color.

Parameters

nativeWindowHandle	Native window handle
clearColor	Clear color (default: white)

Implements cae::IRenderer.
Definition at line 42 of file VULKN.hpp.

42.51.3.8 isVSyncEnabled()

bool cae::VULKN::isVSyncEnabled () const [inline], [nodiscard], [override], [virtual]
Check if VSync is enabled.

Returns

Whether VSync is enabled

Implements cae::IRenderer.
Definition at line 40 of file VULKN.hpp.

42.51.3.9 operator=() [1/2]

VULKN & cae::VULKN::operator= (
const VULKN &) [delete]

42.51.3.10 operator=() [2/2]

VULKN & cae::VULKN::operator= (
VULKN &&) [delete]

42.51.3.11 setClearColor()

void cae::VULKN::setClearColor (
const Color & color) [inline], [override], [virtual]
Set the clear color.

Parameters

color	Clear color to set

Implements cae::IRenderer.
Definition at line 38 of file VULKN.hpp.

42.51.3.12 setVSyncEnabled()

void cae::VULKN::setVSyncEnabled (
bool enabled) [inline], [override], [virtual]
Enable or disable VSync.

Parameters

enabled	Whether VSync is enabled

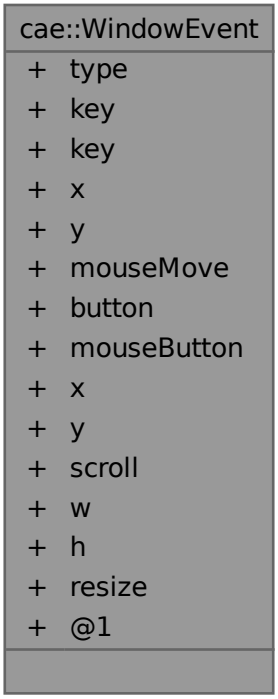
Implements cae::IRenderer.
Definition at line 37 of file VULKN.hpp.
The documentation for this class was generated from the following file:

- plugins/Renderer/Vulkan/include/VULKN/VULKN.hpp

42.52 cae::WindowEvent Struct Reference

Struct for window events.
#include <IWindow.hpp>

Collaboration diagram for cae::WindowEvent:



Public Attributes

- WindowEventType type
- union {
  struct {
  KeyCode key
  } key
  struct {
  int x
  int y
  } mouseMove
  struct {
  MouseButton button
  } mouseButton
  struct {
  float x
  float y
  } scroll
  struct {
  uint16_t w
  uint16_t h
  } resize
  };

42.52.1 Detailed Description

Struct for window events.
Definition at line 62 of file IWindow.hpp.

42.52.2 Member Data Documentation

42.52.2.1 [union]

union { ... } cae::WindowEvent

42.52.2.2 button

MouseButton cae::WindowEvent::button
Definition at line 78 of file IWindow.hpp.

42.52.2.3 h

uint16_t cae::WindowEvent::h
Definition at line 86 of file IWindow.hpp.

42.52.2.4 key [1/2]

KeyCode cae::WindowEvent::key
Definition at line 70 of file IWindow.hpp.

42.52.2.5 [struct] [2/2]

struct { ... } cae::WindowEvent::key

42.52.2.6 [struct]

struct { ... } cae::WindowEvent::mouseButton

42.52.2.7 [struct]

struct { ... } cae::WindowEvent::mouseMove

42.52.2.8 [struct]

struct { ... } cae::WindowEvent::resize

42.52.2.9 [struct]

struct { ... } cae::WindowEvent::scroll

42.52.2.10 type

WindowEventType cae::WindowEvent::type
Definition at line 64 of file IWindow.hpp.
Referenced by cae::GLFW::cursorPosCallback(), cae::GLFW::frameBufferResizeCallback(), cae::GLFW::keyCallback(), cae::GLFW::mouseButtonCallback(), and cae::GLFW::scrollCallback().

42.52.2.11 w

uint16_t cae::WindowEvent::w
Definition at line 86 of file IWindow.hpp.

42.52.2.12 x [1/2]

int cae::WindowEvent::x
Definition at line 74 of file IWindow.hpp.

42.52.2.13 x [2/2]

float cae::WindowEvent::x
Definition at line 82 of file IWindow.hpp.

42.52.2.14 y [1/2]

int cae::WindowEvent::y
Definition at line 74 of file IWindow.hpp.

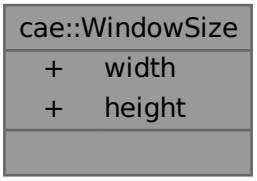
42.52.2.15 y [2/2]

float cae::WindowEvent::y
Definition at line 82 of file IWindow.hpp.
The documentation for this struct was generated from the following file:

- modules/Interfaces/include/Interfaces/Window/IWindow.hpp

42.53 cae::WindowSize Struct Reference

Struct for window size.
#include <IWindow.hpp>
Collaboration diagram for cae::WindowSize:



Public Attributes

- uint16_t width
- uint16_t height

42.53.1 Detailed Description

Struct for window size.
Definition at line 22 of file IWindow.hpp.

42.53.2 Member Data Documentation

42.53.2.1 height

uint16_t cae::WindowSize::height
Definition at line 25 of file IWindow.hpp.
Referenced by cae::GLFW::create(), and cae::OPGL::draw().

42.53.2.2 width

uint16_t cae::WindowSize::width
Definition at line 24 of file IWindow.hpp.

Referenced by cae::GLFW::create(), and cae::OPGL::draw().
The documentation for this struct was generated from the following file:

- modules/Interfaces/include/Interfaces/Window/IWindow.hpp

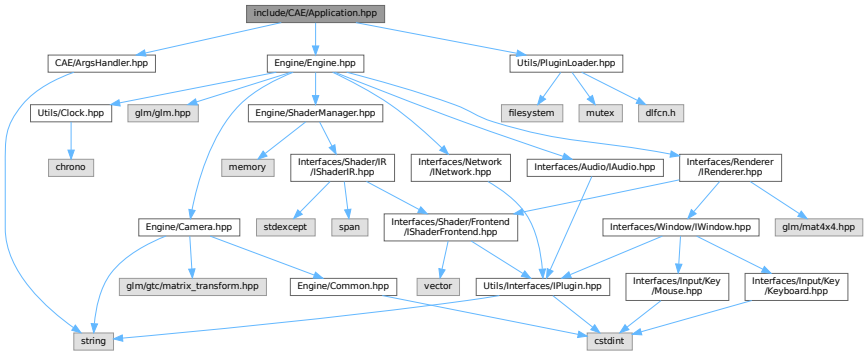
Chapter 43

File Documentation

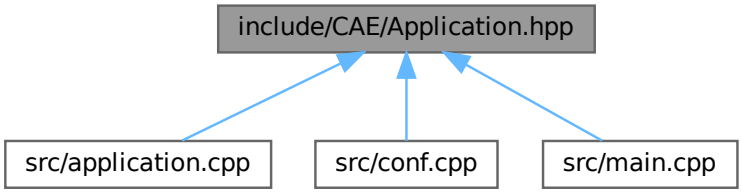
43.1 CONTRIBUTING.md File Reference

43.2 include/CAE/Application.hpp File Reference

This file contains the Application class declaration.
#include ”CAE/ArgsHandler.hpp”
#include ”Engine/Engine.hpp”
#include ”Utils/PluginLoader.hpp”
Include dependency graph for Application.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::AppConfig

Struct for application configuration.

- class cae::Application

  Main class.

Namespaces

- namespace cae

43.2.1 Detailed Description

This file contains the Application class declaration.
Definition in file Application.hpp.

43.3 Application.hpp

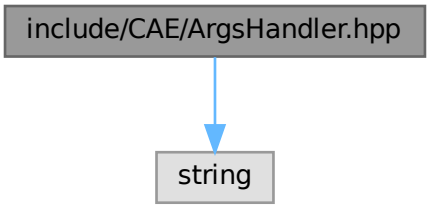
Go to the documentation of this file.

```
00001 ///
00002 /// @file Application.hpp
00003 /// @brief This file contains the Application class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "CAE/ArgsHandler.hpp"
00010
00011 #include "Engine/Engine.hpp"
00012 #include "Utils/PluginLoader.hpp"
00013
00014 namespace cae
00015 {
00016
00017     ///
00018     /// @struct AppConfig
00019     /// @brief Struct for application configuration
00020     /// @namespace cae
00021     ///
00022     struct AppConfig
00023     {
00024         EngineConfig engineConfig;
00025         EnvConfig envConfig;
00026     };
00027
00028     ///
00029     /// @class Application
00030     /// @brief Main class
00031     /// @namespace cae
00032     ///
00033     class Application
00034     {
00035
00036         public:
00037             ///
00038             /// @param argsConfig Command line arguments configuration
00039             /// @param envConfig Environment variables configuration
00040             /// @brief Construct the Application with given configurations
00041             ///
00042             Application(const ArgsConfig &argsConfig, const EnvConfig &envConfig);
00043             ~Application() = default;
00044
00045             Application(const Application &) = delete;
00046             Application &operator=(const Application &) = delete;
00047             Application(Application &&) = delete;
00048             Application &operator=(Application &&) = delete;
00049
00050             ///
00051             /// @brief Start the application
00052             ///
00053             void start();
00054
00055             ///
00056             /// @brief Stop the application
00057             ///
00058             void stop();
00059
00060         private:
00061             ///
00062             /// @param rendererName renderer plugin name
00063             /// @param windowName window plugin name
```

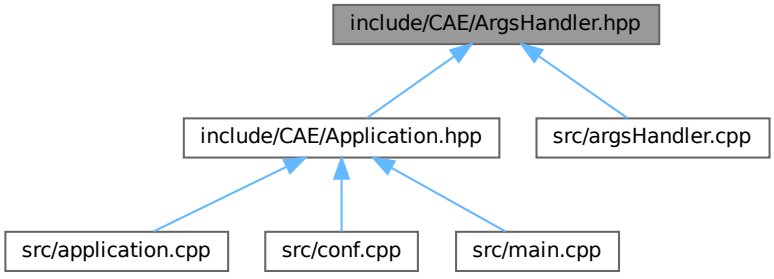
```
00064           /// @param shaderFrontendName shader frontend plugin name
00065           /// @param shaderIRName shader IR plugin name
00066           /// @brief Setup the engine with the given plugins
00067           ///
00068           void setupEngine(const std::string &rendererName, const std::string &windowName,
00069                            const std::string &shaderFrontendName, const std::string &shaderIRName);
00070
00071           ///
00072           /// @param path Path to the engine configuration file
00073           /// @return Parsed EngineConfig
00074           /// @brief Parse the engine configuration file
00075           ///
00076           static EngineConfig parseEngineConf(const std::string &path);
00077
00078           ///
00079           /// @brief main loop
00080           ///
00081           void mainLoop();
00082
00083           std::unique_ptr<utl::PluginLoader> m_pluginLoader = nullptr;
00084           std::unique_ptr<Engine> m_engine = nullptr;
00085
00086           AppConfig m_appConfig;
00087           std::unordered_map<KeyCode, bool> m_keyState;
00088
00089       }; // class Application
00090
00091 } // namespace cae
```

43.4 include/CAE/ArgsHandler.hpp File Reference

This file contains the ArgsHandler class declaration.
#include <string>
Include dependency graph for ArgsHandler.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::ArgsConfig

  Struct for command line arguments configuration.
- struct cae::EnvConfig

  Struct for environment variables configuration.
- class cae::ArgsHandler

  Class to handle command line arguments.

Namespaces

- namespace cae

43.4.1 Detailed Description

This file contains the ArgsHandler class declaration.
Definition in file ArgsHandler.hpp.

43.5 ArgsHandler.hpp

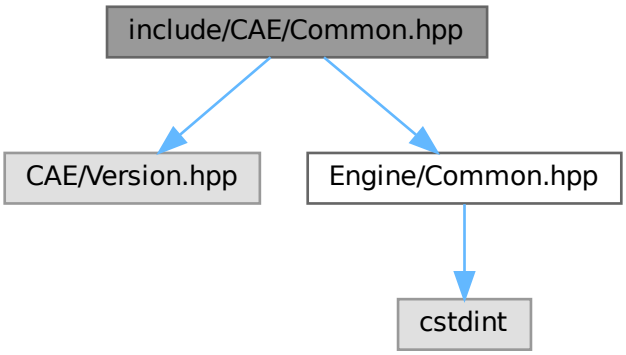
Go to the documentation of this file.

```
00001 ///
00002 /// @file ArgsHandler.hpp
00003 /// @brief This file contains the ArgsHandler class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include <string>
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @struct ArgsConfig
00016     /// @brief Struct for command line arguments configuration
00017     /// @namespace cae
00018     ///
00019     struct ArgsConfig
00020     {
00021         bool run = false;
00022         std::string config_path;
00023     };
00024
00025     ///
00026     /// @struct EnvConfig
00027     /// @brief Struct for environment variables configuration
00028     /// @namespace cae
00029     ///
00030     struct EnvConfig
00031     {
00032         std::string user_name;
00033         std::string pwd;
00034     };
00035
00036     ///
00037     /// @class ArgsHandler
00038     /// @brief Class to handle command line arguments
00039     /// @namespace cae
00040     ///
00041     class ArgsHandler
00042     {
00043
00044         public:
00045             ArgsHandler() = default;
00046             ~ArgsHandler() = default;
00047
00048             ArgsHandler(const ArgsHandler &) = delete;
00049             ArgsHandler &operator=(const ArgsHandler &) = delete;
00050             ArgsHandler(ArgsHandler &&) = delete;
00051             ArgsHandler &operator=(ArgsHandler &&) = delete;
00052
00053             ///
00054             /// @param argc argument count
00055             /// @param argv argument vector
```

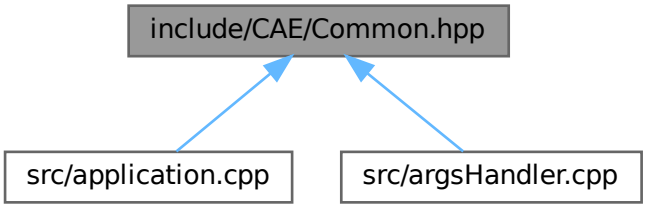
```
00056             /// @return Parsed ArgsConfig
00057             /// @brief Parse command line arguments
00058             ///
00059             static ArgsConfig ParseArgs(int argc, const char *const *argv);
00060
00061             ///
00062             /// @param envp environment pointer
00063             /// @return Parsed EnvConfig
00064             /// @brief Parse environment variables
00065             ///
00066             static EnvConfig ParseEnv(const char *const *envp);
00067
00068         private:
00069     }; // class ArgsHandler
00070
00071 } // namespace cae
```

43.6 include/CAE/Common.hpp File Reference

This file contains the common definitions.
#include "CAE/Version.hpp"
#include "Engine/Common.hpp"
Include dependency graph for Common.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace cae

- namespace cae::MESSAGE
- namespace cae::PLUGINS
- namespace cae::PLUGINS::NAME
- namespace cae::PLUGINS::NAME::RENDERER
- namespace cae::PLUGINS::NAME::SHADER
- namespace cae::PLUGINS::NAME::SHADER::IR
- namespace cae::PLUGINS::NAME::SHADER::FRONTEND
- namespace cae::PLUGINS::NAME::WINDOW
- namespace cae::USER

Variables

- static constexpr std::string_view cae::MESSAGE::HELP_MSG
- static constexpr std::string_view cae::MESSAGE::VERSION_MSG
- constexpr auto cae::PLUGINS::NAME::RENDERER::OPENGL = "OpenGL"
- constexpr auto cae::PLUGINS::NAME::RENDERER::VULKAN = "Vulkan"
- constexpr auto cae::PLUGINS::NAME::SHADER::IR::SPIRV = "SPIRV"
- constexpr auto cae::PLUGINS::NAME::SHADER::FRONTEND::GLSL = "GLSL"
- constexpr auto cae::PLUGINS::NAME::WINDOW::COCOA = "Cocoa"
- constexpr auto cae::PLUGINS::NAME::WINDOW::GLFW = "GLFW"
- constexpr auto cae::PLUGINS::NAME::WINDOW::WIN32_ = "Win32"
- constexpr auto cae::PLUGINS::NAME::WINDOW::X11 = "X11"
- constexpr auto cae::USER::NAME = "User"

43.6.1 Detailed Description

This file contains the common definitions.
Definition in file Common.hpp.

43.7 Common.hpp

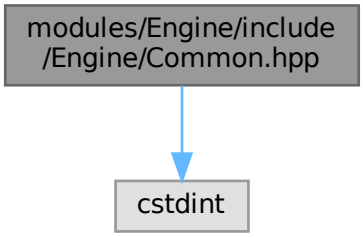
Go to the documentation of this file.

```
00001 ///
00002 /// @file Common.hpp
00003 /// @brief This file contains the common definitions.
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "CAE/Version.hpp"
00010
00011 #include "Engine/Common.hpp"
00012
00013 namespace cae
00014 {
00015     namespace MESSAGE
00016     {
00017         static constexpr std::string_view HELP_MSG = "Usage: " PROJECT_NAME APP_EXTENSION " [options]\n\n"
00018                                                      "Options:\n"
00019                                                      "  -h, --help              Show this help message\n"
00020                                                      "  -v, --version           Show version information\n"
00021                                                      "  -c, --config <path>     Specify JSON configuration file";
00022         static constexpr std::string_view VERSION_MSG = PROJECT_NAME
00023             " v" PROJECT_VERSION " " BUILD_TYPE " (" GIT_TAG ", commit " GIT_COMMIT_HASH ") "
    __DATE__ " " __TIME__;
00024     } // namespace MESSAGE
00025
00026     namespace PLUGINS::NAME
00027     {
00028         namespace RENDERER
00029         {
00030             inline constexpr auto OPENGL = "OpenGL";
00031             inline constexpr auto VULKAN = "Vulkan";
00032         } // namespace RENDERER
00033
00034         namespace SHADER
00035         {
00036             namespace IR
00037             {
```

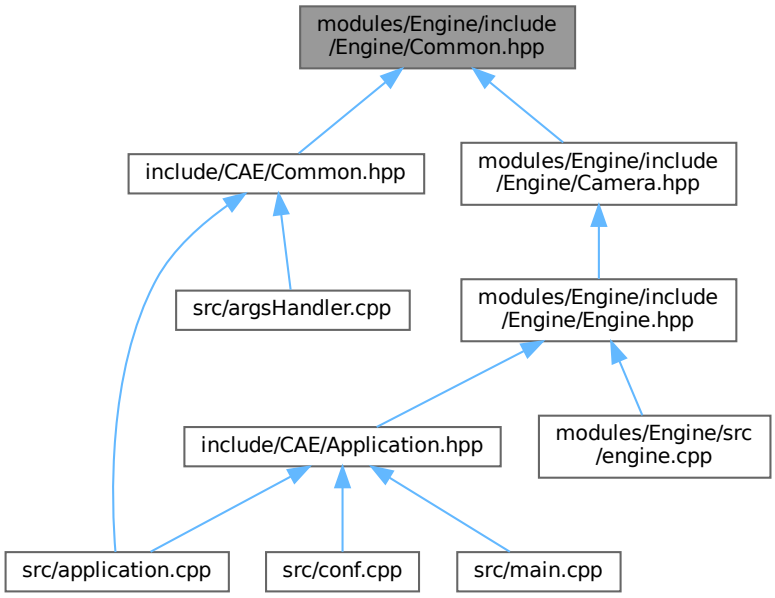
```
00038             inline constexpr auto SPIRV = "SPIRV";
00039         } // namespace IR
00040
00041         namespace FRONTEND
00042         {
00043             inline constexpr auto GLSL = "GLSL";
00044         } // namespace FRONTEND
00045
00046     } // namespace SHADER
00047
00048     namespace WINDOW
00049     {
00050         inline constexpr auto COCOA = "Cocoa";
00051         inline constexpr auto GLFW = "GLFW";
00052         inline constexpr auto WIN32_ = "Win32";
00053         inline constexpr auto X11 = "X11";
00054     } // namespace WINDOW
00055
00056 } // namespace PLUGINS::NAME
00057
00058 namespace USER
00059 {
00060     inline constexpr auto NAME = "User";
00061 } // namespace USER
00062
00063 } // namespace cae
```

43.8 modules/Engine/include/Engine/Common.hpp File Reference

This file contains common constants used across the engine.
#include <cstdint>
Include dependency graph for Common.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace cae
- namespace cae::AUDIO
- namespace cae::CAMERA
- namespace cae::LOG
- namespace cae::NETWORK
- namespace cae::RENDERER
- namespace cae::WINDOW

Macros

- #define APP_EXTENSION ""

Variables

- constexpr auto cae::AUDIO::VOLUME = 1.F
- constexpr auto cae::AUDIO::MUTED = false
- constexpr auto cae::CAMERA::NAME = "Default name"
- constexpr auto cae::CAMERA::MOVE_SPEED = 2.5F
- constexpr auto cae::CAMERA::LOOK_SPEED = 10.0F
- constexpr auto cae::CAMERA::FOV = 45.F
- constexpr auto cae::CAMERA::NEAR_PLANE = 0.1F
- constexpr auto cae::CAMERA::FAR_PLANE = 100.F
- constexpr auto cae::LOG::LOG_FPS = false
- constexpr auto cae::NETWORK::HOST = "127.0.0.1"
- constexpr auto cae::NETWORK::PORT = 4242
- constexpr auto cae::RENDERER::VSYNC = false
- constexpr auto cae::RENDERER::FRAME_RATE_LIMIT = 90
- constexpr auto cae::RENDERER::CLEAR_COLOR_R = 1.0F

- constexpr auto cae::RENDERER::CLEAR_COLOR_G = 1.0F
- constexpr auto cae::RENDERER::CLEAR_COLOR_B = 1.0F
- constexpr auto cae::RENDERER::CLEAR_COLOR_A = 1.0F
- constexpr uint16_t cae::WINDOW::WIDTH = 960
- constexpr uint16_t cae::WINDOW::HEIGHT = 540
- constexpr auto cae::WINDOW::NAME = "Default name"
- constexpr auto cae::WINDOW::FULLSCREEN = false
- constexpr auto cae::WINDOW::ICON_PATH = ""

43.8.1 Detailed Description

This file contains common constants used across the engine.
Definition in file Common.hpp.

43.8.2 Macro Definition Documentation

43.8.2.1 APP_EXTENSION

#define APP_EXTENSION ""
Definition at line 14 of file Common.hpp.

43.9 Common.hpp

Go to the documentation of this file.

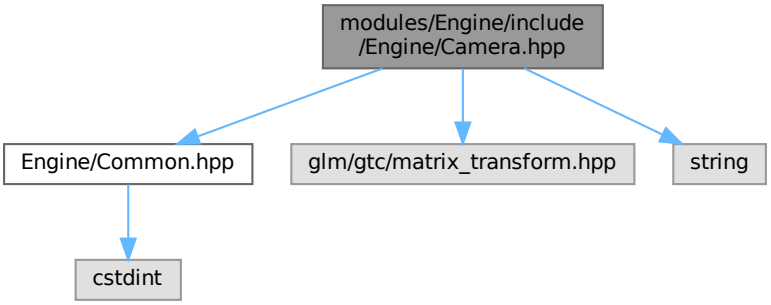
```
00001 ///
00002 /// @file Common.hpp
00003 /// @brief This file contains common constants used across the engine
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include <cstdint>
00010
00011 #ifdef _WIN32
00012 #define APP_EXTENSION ".exe"
00013 #else
00014 #define APP_EXTENSION ""
00015 #endif
00016
00017 namespace cae
00018 {
00019     namespace AUDIO
00020     {
00021         inline constexpr auto VOLUME = 1.F;
00022         inline constexpr auto MUTED = false;
00023     } // namespace AUDIO
00024
00025     namespace CAMERA
00026     {
00027         inline constexpr auto NAME = "Default name";
00028         inline constexpr auto MOVE_SPEED = 2.5F;
00029         inline constexpr auto LOOK_SPEED = 10.0F;
00030         inline constexpr auto FOV = 45.F;
00031         inline constexpr auto NEAR_PLANE = 0.1F;
00032         inline constexpr auto FAR_PLANE = 100.F;
00033     } // namespace CAMERA
00034
00035     namespace LOG
00036     {
00037         inline constexpr auto LOG_FPS = false;
00038     } // namespace LOG
00039
00040     namespace NETWORK
00041     {
00042         inline constexpr auto HOST = "127.0.0.1";
00043         inline constexpr auto PORT = 4242;
00044     } // namespace NETWORK
00045
00046     namespace RENDERER
00047     {
00048         inline constexpr auto VSYNC = false;
00049         inline constexpr auto FRAME_RATE_LIMIT = 90;
00050         inline constexpr auto CLEAR_COLOR_R = 1.0F;
00051         inline constexpr auto CLEAR_COLOR_G = 1.0F;
```

```
00052         inline constexpr auto CLEAR_COLOR_B = 1.0F;
00053         inline constexpr auto CLEAR_COLOR_A = 1.0F;
00054     } // namespace RENDERER
00055
00056     namespace WINDOW
00057     {
00058         inline constexpr uint16_t WIDTH = 960;
00059         inline constexpr uint16_t HEIGHT = 540;
00060         inline constexpr auto NAME = "Default name";
00061         inline constexpr auto FULLSCREEN = false;
00062         inline constexpr auto ICON_PATH = "";
00063     } // namespace WINDOW
00064
00065 } // namespace cae
```

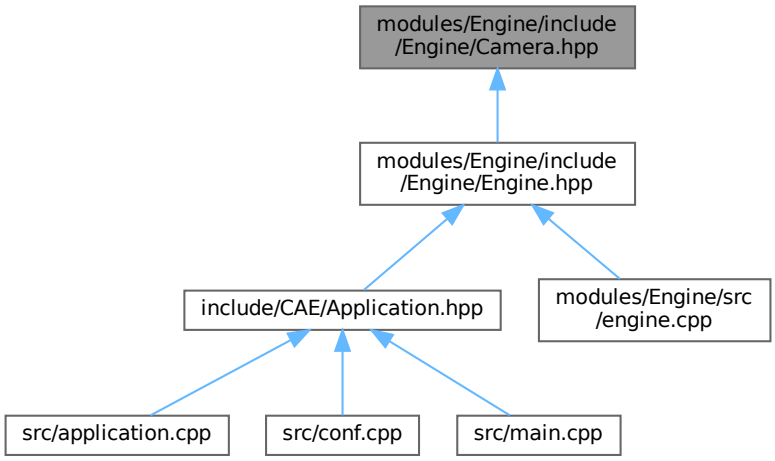
43.10 LICENSE.md File Reference

43.11 modules/Engine/include/Engine/Camera.hpp File Reference

This file contains the camera class declaration.
#include "Engine/Common.hpp"
#include <glm/gtc/matrix_transform.hpp>
#include <string>
Include dependency graph for Camera.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class cae::Camera

  Class for camera.

Namespaces

- namespace cae

43.11.1 Detailed Description

This file contains the camera class declaration.
Definition in file Camera.hpp.

43.12 Camera.hpp

Go to the documentation of this file.

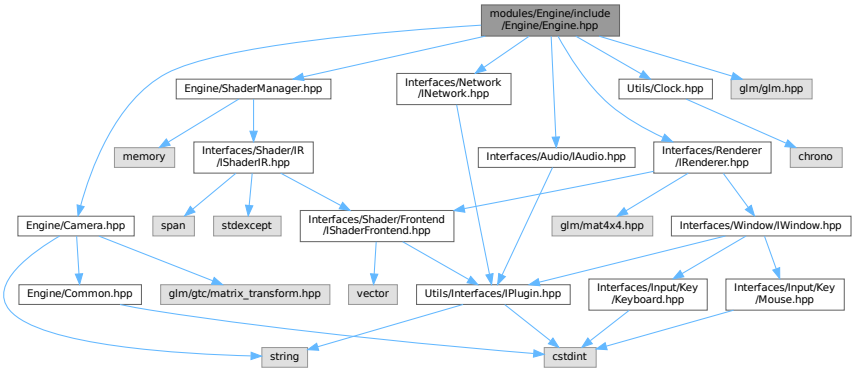
```
00001 ///
00002 /// @file Camera.hpp
00003 /// @brief This file contains the camera class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Engine/Common.hpp"
00010
00011 #include <glm/gtc/matrix_transform.hpp>
00012
00013 #include <string>
00014
00015 namespace cae
00016 {
00017
00018     ///
00019     /// @class Camera
00020     /// @brief Class for camera
00021     /// @namespace cae
00022     ///
00023     class Camera
00024     {
00025         public:
00026             Camera(const glm::vec3 position, const glm::vec3 rotation, const glm::vec3 direction,
00027                 const float moveSpeed = CAMERA::MOVE_SPEED, const float lookSpeed = CAMERA::LOOK_SPEED,
```

```
00028                const float fov = CAMERA::FOV, const float nearPlane = CAMERA::NEAR_PLANE,
00029                const float farPlane = CAMERA::FAR_PLANE)
00030            : m_position(position), m_rotation(rotation), m_direction(direction), m_moveSpeed(moveSpeed),
00031              m_lookSpeed(lookSpeed), m_fov(fov), m_near(nearPlane), m_far(farPlane)
00032        {
00033        }
00034        ~Camera() = default;
00035
00036        Camera(const Camera &) = delete;
00037        Camera &operator=(const Camera &) = delete;
00038        Camera(Camera &&) = delete;
00039        Camera &operator=(Camera &&) = delete;
00040
00041        void setName(const std::string &name) { m_name = name; }
00042        void setPosition(const glm::vec3 &position) { m_position = position; }
00043        void setRotation(const glm::vec3 &rotation) { m_rotation = rotation; }
00044        void setDirection(const glm::vec3 &direction) { m_direction = direction; }
00045        void setMoveSpeed(const float speed) { m_moveSpeed = speed; }
00046        void setLookSpeed(const float speed) { m_lookSpeed = speed; }
00047        void setFov(const float fov) { m_fov = fov; }
00048        void setNear(const float nearPlane) { m_near = nearPlane; }
00049        void setFar(const float farPlane) { m_far = farPlane; }
00050
00051        [[nodiscard]] const std::string &getName() const { return m_name; }
00052        [[nodiscard]] const glm::vec3 &getPosition() const { return m_position; }
00053        [[nodiscard]] const glm::vec3 &getRotation() const { return m_rotation; }
00054        [[nodiscard]] const glm::vec3 &getDirection() const { return m_direction; }
00055        [[nodiscard]] const float &getMoveSpeed() const { return m_moveSpeed; }
00056        [[nodiscard]] const float &getLookSpeed() const { return m_lookSpeed; }
00057        [[nodiscard]] const float &getFov() const { return m_fov; }
00058        [[nodiscard]] const float &getNear() const { return m_near; }
00059        [[nodiscard]] const float &getFar() const { return m_far; }
00060
00061        [[nodiscard]] glm::mat4 getViewMatrix() const
00062        {
00063            return glm::lookAt(m_position, m_position + m_direction, glm::vec3(0.0F, 1.0F, 0.0F));
00064        }
00065        [[nodiscard]] glm::mat4 getProjectionMatrix(const float aspectRatio) const
00066        {
00067            return glm::perspective(glm::radians(m_fov), aspectRatio, m_near, m_far);
00068        }
00069        [[nodiscard]] glm::mat4 getViewProjection(const float aspectRatio) const
00070        {
00071            return getProjectionMatrix(aspectRatio) * getViewMatrix();
00072        }
00073
00074        ///
00075        /// @param direction Direction to move the camera
00076        /// @param deltaTime Time delta for movement
00077        /// @brief Move the camera in a given direction
00078        ///
00079        void move(const glm::vec3 &direction, const float deltaTime)
00080        {
00081            m_position += direction * m_moveSpeed * deltaTime;
00082        }
00083
00084        ///
00085        /// @param yawOffset Yaw offset to rotate the camera
00086        /// @param pitchOffset Pitch offset to rotate the camera
00087        /// @param deltaTime Time delta for rotation
00088        /// @brief Rotate the camera by given yaw and pitch offsets
00089        ///
00090        void rotate(const float yawOffset, const float pitchOffset, const float deltaTime)
00091        {
00092            m_rotation.y += yawOffset * m_lookSpeed * deltaTime;
00093            m_rotation.x += pitchOffset * m_lookSpeed * deltaTime;
00094
00095            m_rotation.x = std::min(m_rotation.x, 89.0F);
00096            m_rotation.x = std::max(m_rotation.x, -89.0F);
00097
00098            updateDirectionFromRotation();
00099        }
00100
00101    private:
00102        std::string m_name = CAMERA::NAME;
00103
00104        glm::vec3 m_position = glm::vec3(0.0F, 0.0F, 0.0F);
00105        glm::vec3 m_rotation = glm::vec3(0.0F, 0.0F, 0.0F);
00106        glm::vec3 m_direction = glm::vec3(0.0F, 0.0F, -1.0F);
00107
00108        float m_moveSpeed = CAMERA::MOVE_SPEED;
00109        float m_lookSpeed = CAMERA::LOOK_SPEED;
00110
00111        float m_fov = CAMERA::FOV;
00112        float m_near = CAMERA::NEAR_PLANE;
00113        float m_far = CAMERA::FAR_PLANE;
00114
```

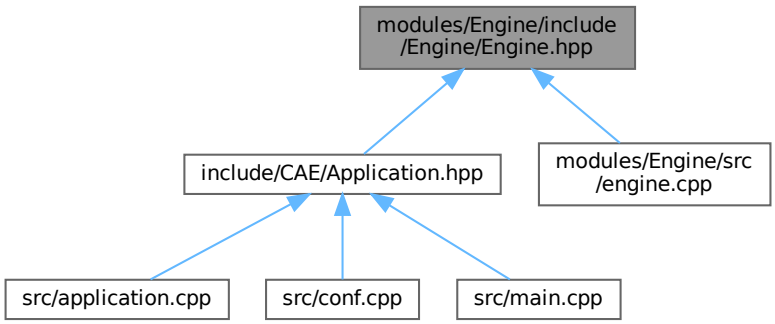
```
00115         void updateDirectionFromRotation()
00116         {
00117             const float yaw = glm::radians(m_rotation.y);
00118             const float pitch = glm::radians(m_rotation.x);
00119 
00120             m_direction.x = cos(pitch) * sin(yaw);
00121             m_direction.y = std::sin(pitch);
00122             m_direction.z = -cos(pitch) * cos(yaw);
00123             m_direction = glm::normalize(m_direction);
00124         }
00125 
00126     }; // class Camera
00127 
00128 } // namespace cae
```

43.13 modules/Engine/include/Engine/Engine.hpp File Reference

This file contains the engine class declaration.
#include ”Engine/Camera.hpp”
#include ”Engine/ShaderManager.hpp”
#include ”Interfaces/Audio/IAudio.hpp”
#include ”Interfaces/Network/INetwork.hpp”
#include ”Interfaces/Renderer/IRenderer.hpp”
#include ”Utils/Clock.hpp”
#include <glm/glm.hpp>
Include dependency graph for Engine.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::EngineConfig

  Struct for engine configuration.
- class cae::Engine

  Engine class.

Namespaces

- namespace cae

43.13.1 Detailed Description

This file contains the engine class declaration.
Definition in file Engine.hpp.

43.14 Engine.hpp

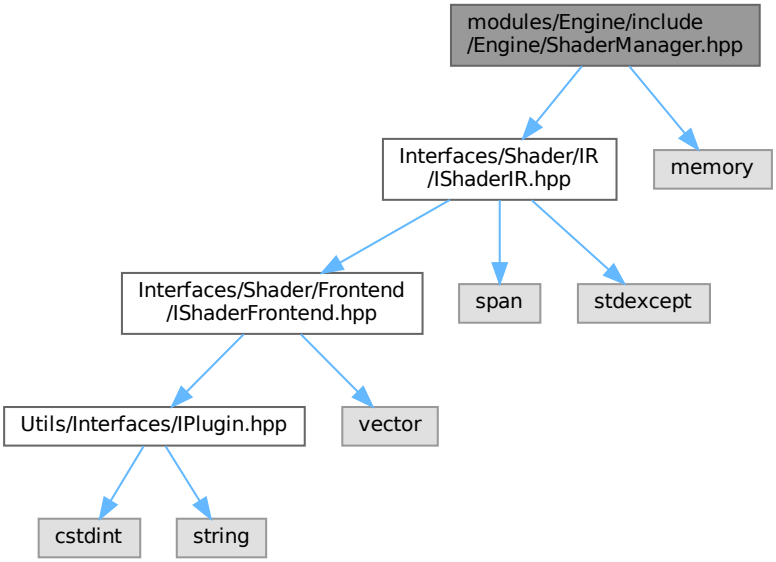
Go to the documentation of this file.

```
00001 ///
00002 /// @file Engine.hpp
00003 /// @brief This file contains the engine class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Engine/Camera.hpp"
00010 #include "Engine/ShaderManager.hpp"
00011
00012 #include "Interfaces/Audio/IAudio.hpp"
00013 #include "Interfaces/Network/INetwork.hpp"
00014 #include "Interfaces/Renderer/IRenderer.hpp"
00015 #include "Utils/Clock.hpp"
00016
00017 #include <glm/glm.hpp>
00018
00019 namespace cae
00020 {
00021
00022     ///
00023     /// @struct EngineConfig
00024     /// @brief Struct for engine configuration
00025     /// @namespace cae
00026     ///
00027     struct EngineConfig
00028     {
00029         float audio_master_volume = AUDIO::VOLUME;
00030         bool audio_muted = AUDIO::MUTED;
00031
00032         glm::vec3 camera_position = glm::vec3(0.0F, 0.0F, 0.0F);
00033         glm::vec3 camera_rotation = glm::vec3(0.0F, 0.0F, 0.0F);
00034         glm::vec3 camera_direction = glm::vec3(0.0F, 0.0F, -1.0F);
00035         float camera_move_speed = CAMERA::MOVE_SPEED;
00036         float camera_look_speed = CAMERA::LOOK_SPEED;
00037         float camera_fov = CAMERA::FOV;
00038         float camera_near_plane = CAMERA::NEAR_PLANE;
00039         float camera_far_plane = CAMERA::FAR_PLANE;
00040
00041         bool log_fps = LOG::LOG_FPS;
00042
00043         std::string network_host = NETWORK::HOST;
00044         uint16_t network_port = NETWORK::PORT;
00045
00046         bool renderer_vsync = RENDERER::VSYNC;
00047         uint16_t renderer_frame_rate_limit = RENDERER::FRAME_RATE_LIMIT;
00048         Color renderer_clear_color = {.r = RENDERER::CLEAR_COLOR_R,
00049                                       .g = RENDERER::CLEAR_COLOR_G,
00050                                       .b = RENDERER::CLEAR_COLOR_B,
00051                                       .a = RENDERER::CLEAR_COLOR_A};
00052
00053         uint16_t window_width = WINDOW::WIDTH;
00054         uint16_t window_height = WINDOW::HEIGHT;
00055         bool window_fullscreen = WINDOW::FULLSCREEN;
00056         std::string window_name = WINDOW::NAME;
00057         std::string window_icon_path = WINDOW::ICON_PATH;
00058     };
00059
```

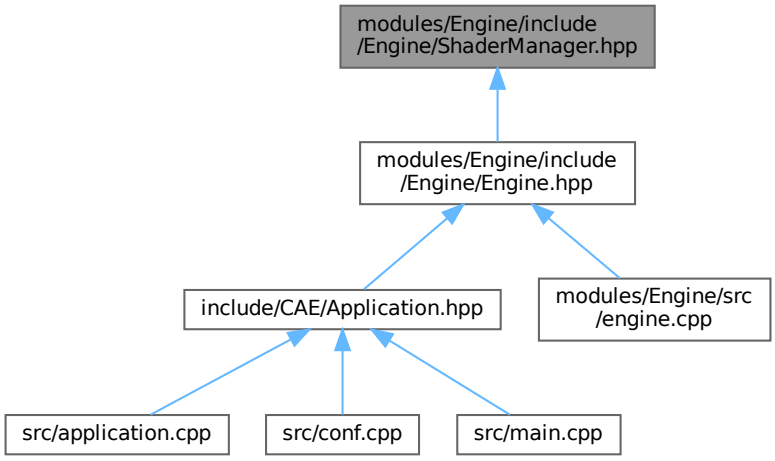
```
00060     ///
00061     /// @class Engine
00062     /// @brief Engine class
00063     /// @namespace cae
00064     ///
00065     class Engine
00066     {
00067
00068         public:
00069             Engine(const EngineConfig &config, const std::function<std::shared_ptr<IAudio>()> &audioFactory,
00070                    const std::function<std::shared_ptr<INetwork>()> &networkFactory,
00071                    const std::function<std::shared_ptr<IRenderer>()> &rendererFactory,
00072                    const std::function<std::shared_ptr<IShaderIR>()> &shaderIRFactory,
00073                    const std::vector<std::function<std::shared_ptr<IShaderFrontend>()» &shaderFrontendFactories,
00074                    const std::function<std::shared_ptr<IWindow>()> &windowFactory);
00075             ~Engine() = default;
00076
00077             Engine(const Engine &) = delete;
00078             Engine &operator=(const Engine &) = delete;
00079             Engine(Engine &&) = delete;
00080             Engine &operator=(Engine &&) = delete;
00081
00082             [[nodiscard]] const std::shared_ptr<IAudio> &getAudio() const { return m_audioPlugin; }
00083             [[nodiscard]] const std::shared_ptr<INetwork> &getNetwork() const { return m_networkPlugin; }
00084             [[nodiscard]] const std::shared_ptr<IRenderer> &getRenderer() const { return m_rendererPlugin; }
00085             [[nodiscard]] const std::shared_ptr<IWindow> &getWindow() const { return m_windowPlugin; }
00086
00087             [[nodiscard]] const std::unique_ptr<utl::Clock> &getClock() { return m_clock; }
00088             [[nodiscard]] const std::unique_ptr<ShaderManager> &getShaderManager() const { return m_shaderManager; }
00089             [[nodiscard]] const std::unique_ptr<Camera> &getCamera() const { return m_camera; }
00090
00091             ///
00092             /// @param shaderSources Shader sources to initialize
00093             /// @param vertices Vertex data to initialize
00094             /// @brief Initialize render resources
00095             ///
00096             void initializeRenderResources(const std::vector<ShaderSourceDesc> &shaderSources,
00097                                            const std::vector<float> &vertices) const;
00098
00099             ///
00100             /// @param fpsBuffer
00101             /// @param fpsIndex
00102             /// @brief
00103             ///
00104             void update(std::array<float, 10> &fpsBuffer, int &fpsIndex);
00105
00106             ///
00107             /// @brief
00108             ///
00109             void render();
00110
00111             ///
00112             /// @brief Stop the engine
00113             ///
00114             void stop();
00115
00116         private:
00117             std::shared_ptr<IAudio> m_audioPlugin = nullptr;
00118             std::shared_ptr<INetwork> m_networkPlugin = nullptr;
00119             std::shared_ptr<IRenderer> m_rendererPlugin = nullptr;
00120             std::shared_ptr<IWindow> m_windowPlugin = nullptr;
00121
00122             std::unique_ptr<utl::Clock> m_clock = nullptr;
00123             std::unique_ptr<ShaderManager> m_shaderManager = nullptr;
00124             std::unique_ptr<Camera> m_camera = nullptr;
00125
00126             bool m_logFps = false;
00127
00128             ///
00129             /// @param windowName window name
00130             /// @param windowSize window size
00131             /// @param iconPath path to window icon
00132             /// @brief Initialize the window
00133             ///
00134             void initWindow(const std::string &windowName, const WindowSize &windowSize,
00135                             const std::string &iconPath) const;
00136
00137             ///
00138             /// @param shaderSources Shader sources to initialize
00139             /// @brief Initialize shaders
00140             ///
00141             void initShaders(const std::vector<ShaderSourceDesc> &shaderSources) const;
00142
00143     }; // class Engine
00144
00145 } // namespace cae
```

43.15 modules/Engine/include/Engine/ShaderManager.hpp File Reference

This file contains the ShaderManager class declaration.
#include "Interfaces/Shader/IR/IShaderIR.hpp"
#include <memory>
Include dependency graph for ShaderManager.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class cae::ShaderManager

Class for managing shaders.

Namespaces

- namespace cae

43.15.1 Detailed Description

This file contains the ShaderManager class declaration.
Definition in file ShaderManager.hpp.

43.16 ShaderManager.hpp

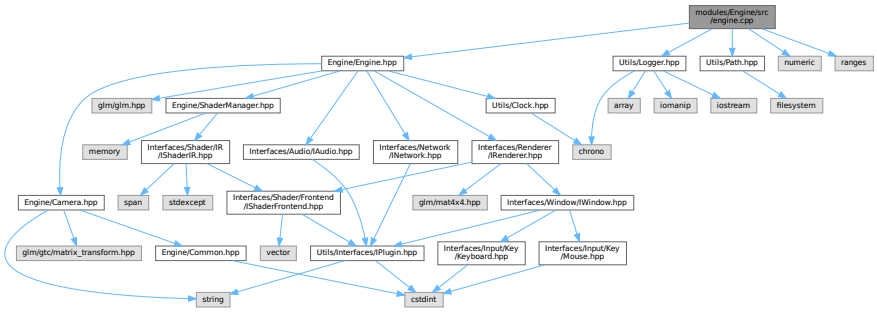
Go to the documentation of this file.

```
00001 ///
00002 /// @file ShaderManager.hpp
00003 /// @brief This file contains the ShaderManager class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Shader/IR/IShaderIR.hpp"
00010
00011 #include <memory>
00012
00013 namespace cae
00014 {
00015     ///
00016     /// @class ShaderManager
00017     /// @brief Class for managing shaders
00018     /// @namespace cae
00019     ///
00020     class ShaderManager
00021     {
00022         public:
00023             explicit ShaderManager(
00024                 const std::vector<std::function<std::shared_ptr<IShaderFrontend>()» &shaderFrontendFactories,
00025                 const std::function<std::shared_ptr<IShaderIR>()> &shaderIRFactory = nullptr)
00026             {
00027                 for (const auto &factory : shaderFrontendFactories)
00028                 {
00029                     auto frontend = factory();
00030                     registerFrontend(frontend);
00031                 }
00032                 if (shaderIRFactory)
00033                 {
00034                     registerIR(shaderIRFactory());
00035                 }
00036             }
00037             ~ShaderManager() = default;
00038
00039             ShaderManager(const ShaderManager &) = delete;
00040             ShaderManager &operator=(const ShaderManager &) = delete;
00041             ShaderManager(ShaderManager &&) = delete;
00042             ShaderManager &operator=(ShaderManager &&) = delete;
00043
00044             std::unordered_map<ShaderID, ShaderIRModule> build(const std::vector<ShaderSourceDesc> &sources,
00045                                                                const ShaderSourceType targetIR) const
00046             {
00047                 std::unordered_map<ShaderID, ShaderIRModule> out;
00048
00049                 const auto irProcessor = m_irs.at(targetIR);
00050                 for (const auto &src : sources)
00051                 {
00052                     const auto f = m_frontends.at(src.type);
00053                     ShaderIRModule ir = f->compile(src);
00054                     const ShaderIRModule final = irProcessor->process(ir);
00055                     out[src.id] = final;
00056                 }
00057
00058                 return out;
00059             }
00060
00061             template <std::ranges::input_range R> void optimizeAll(const ShaderSourceType irType, R &&modules) const
00062             {
00063                 if (auto it = m_irs.find(irType); it != m_irs.end())
00064                 {
00065                     std::vector<ShaderIRModule *> ptrs;
00066                     for (auto &m : modules)
```

```
00067                 {
00068                     ptrs.push_back(&m);
00069                 }
00070
00071                 std::vector<ShaderIRModule> tmp;
00072                 tmp.reserve(ptrs.size());
00073                 for (auto *p : ptrs)
00074                 {
00075                     tmp.push_back(*p);
00076                 }
00077
00078                 it->second->optimize(tmp);
00079
00080                 for (size_t i = 0; i < ptrs.size(); ++i)
00081                 {
00082                     *ptrs[i] = std::move(tmp[i]);
00083                 }
00084             }
00085         }
00086
00087     private:
00088         void registerFrontend(const std::shared_ptr<IShaderFrontend> &f) { m_frontends[f->sourceType()] = f; }
00089         void registerIR(const std::shared_ptr<IShaderIR> &ir) { m_irs[ir->irType()] = ir; }
00090
00091         std::unordered_map<ShaderSourceType, std::shared_ptr<IShaderFrontend>> m_frontends;
00092         std::unordered_map<ShaderSourceType, std::shared_ptr<IShaderIR>> m_irs;
00093
00094 }; // class ShaderManager
00095
00096 } // namespace cae
```

43.17 modules/Engine/src/engine.cpp File Reference

#include "Engine/Engine.hpp"
#include "Utils/Logger.hpp"
#include "Utils/Path.hpp"
#include <numeric>
#include <ranges>
Include dependency graph for engine.cpp:



Functions

- void printFps (std::array< float, 10 > &fpsBuffer, int &fpsIndex, const float deltaTime)

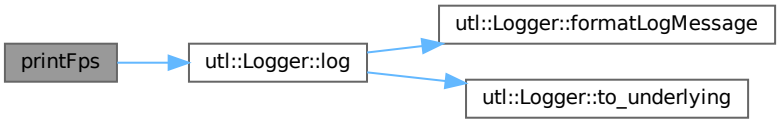
43.17.1 Function Documentation

43.17.1.1 printFps()

void printFps (
    std::array< float, 10 > & fpsBuffer,
    int & fpsIndex,
    const float deltaTime)

Definition at line 9 of file engine.cpp.
References utl::INFO, and utl::Logger::log().
Referenced by cae::Engine::update().

Here is the call graph for this function:



Here is the caller graph for this function:

43.18 engine.cpp

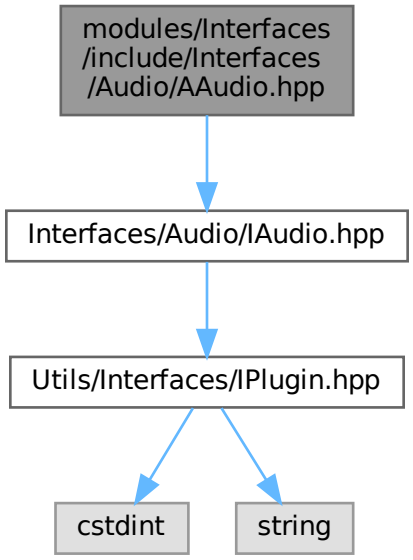
Go to the documentation of this file.

```
00001 #include "Engine/Engine.hpp"
00002
00003 #include "Utils/Logger.hpp"
00004 #include "Utils/Path.hpp"
00005
00006 #include <numeric>
00007 #include <ranges>
00008
00009 void printFps(std::array<float, 10> &fpsBuffer, int &fpsIndex, const float deltaTime)
00010 {
00011     fpsBuffer[fpsIndex % 10] = 1.0F / deltaTime;
00012     fpsIndex++;
00013
00014     float avgFps = std::accumulate(fpsBuffer.begin(), fpsBuffer.end(), 0.0F) / 10.0F;
00015     utl::Logger::log(std::format("FPS: {}", avgFps), utl::LogLevel::INFO);
00016 }
00017
00018 cae::Engine::Engine(const EngineConfig &config, const std::function<std::shared_ptr<IAudio>()> &audioFactory,
00019                     const std::function<std::shared_ptr<INetwork>()> &networkFactory,
00020                     const std::function<std::shared_ptr<IRenderer>()> &rendererFactory,
00021                     const std::function<std::shared_ptr<IShaderIR>()> &shaderIRFactory,
00022                     const std::vector<std::function<std::shared_ptr<IShaderFrontend>()» &shaderFrontendFactories,
00023                     const std::function<std::shared_ptr<IWindow>()> &windowFactory)
00024     : m_audioPlugin(audioFactory()), m_networkPlugin(networkFactory()), m_rendererPlugin(rendererFactory()),
00025       m_windowPlugin(windowFactory()), m_clock(std::make_unique<utl::Clock>()),
00026       m_shaderManager(std::make_unique<ShaderManager>(shaderFrontendFactories, shaderIRFactory)),
00027       m_camera(std::make_unique<Camera>(config.camera_position, config.camera_rotation, config.camera_direction,
00028                                         config.camera_move_speed, config.camera_look_speed, config.camera_fov,
00029                                         config.camera_near_plane, config.camera_far_plane)),
00030       m_logFps(config.log_fps)
00031 {
00032     constexpr auto boolToStr = [](const bool b) { return b ? "true" : "false"; };
00033     std::ostringstream msg;
00034     msg « "Starting engine with configuration:\n"
00035         « "\tAudio master volume: " « config.audio_master_volume « "\n"
00036         « "\tAudio muted: " « boolToStr(config.audio_muted) « "\n"
00037         « "\tLog FPS: " « boolToStr(config.log_fps) « "\n"
00038         « "\tNetwork host: " « config.network_host « "\n"
00039         « "\tNetwork port: " « config.network_port « "\n"
00040         « "\tRenderer vsync: " « boolToStr(config.renderer_vsync) « "\n"
00041         « "\tRenderer frame rate limit: " « config.renderer_frame_rate_limit « "\n"
00042         « "\tRenderer clear color: (" « config.renderer_clear_color.r « ", " « config.renderer_clear_color.g « ", "
00043         « config.renderer_clear_color.b « ", " « config.renderer_clear_color.a « ")\n"
00044         « "\tWindow width: " « config.window_width « "\n"
00045         « "\tWindow height: " « config.window_height « "\n"
```

```
00046         « "\tWindow fullscreen: " « boolToStr(config.window_fullscreen) « "\n"
00047         « "\tWindow name: " « config.window_name « "\n\tWindow icon path: " « config.window_icon_path « '\n';
00048     utl::Logger::log(msg.str(), utl::LogLevel::INFO);
00049
00050     initWindow(config.window_name, {.width = config.window_width, .height = config.window_height},
00051                config.window_icon_path);
00052     m_rendererPlugin->initialize(m_windowPlugin->getNativeHandle(), config.renderer_clear_color);
00053 }
00054
00055 void cae::Engine::initializeRenderResources(const std::vector<ShaderSourceDesc> &shaderSources,
00056                                             const std::vector<float> &vertices) const
00057 {
00058     initShaders(shaderSources);
00059     m_rendererPlugin->createMesh(vertices);
00060 }
00061
00062 void cae::Engine::render()
00063 {
00064     constexpr auto model = glm::mat4(1.0F);
00065
00066     const glm::mat4 mvp =
      m_camera->getViewProjection(static_cast<float>(m_windowPlugin->getWindowSize().width) /
00067                                                    m_windowPlugin->getWindowSize().height) *
00068                         model;
00069     m_rendererPlugin->draw(m_windowPlugin->getWindowSize(), "basic", mvp);
00070 }
00071
00072 void cae::Engine::update(std::array<float, 10> &fpsBuffer, int &fpsIndex)
00073 {
00074     if (m_logFps)
00075     {
00076         printFps(fpsBuffer, fpsIndex, m_clock->getDeltaSeconds());
00077     }
00078     m_clock->restart();
00079 }
00080
00081 void cae::Engine::stop()
00082 {
00083     utl::Logger::log("Stopping engine...", utl::LogLevel::INFO);
00084     m_windowPlugin->close();
00085
00086     m_audioPlugin = nullptr;
00087     m_networkPlugin = nullptr;
00088     m_rendererPlugin = nullptr;
00089     m_windowPlugin = nullptr;
00090
00091     m_clock = nullptr;
00092     m_shaderManager = nullptr;
00093     m_camera = nullptr;
00094 }
00095
00096 void cae::Engine::initWindow(const std::string &windowName, const WindowSize &windowSize,
00097                              const std::string &iconPath) const
00098 {
00099     m_windowPlugin->create(windowName, windowSize);
00100     if (!iconPath.empty())
00101     {
00102         m_windowPlugin->setIcon(iconPath);
00103     }
00104 }
00105
00106 void cae::Engine::initShaders(const std::vector<ShaderSourceDesc> &shaderSources) const
00107 {
00108     auto shaders = m_shaderManager->build(shaderSources, ShaderSourceType::SPIRV);
00109     m_shaderManager->optimizeAll(ShaderSourceType::SPIRV, shaders | std::views::values);
00110     m_rendererPlugin->createPipeline("basic", shaders["basic_vertex"], shaders["basic_fragment"]);
00111 }
```

43.19 modules/Interfaces/include/Interfaces/Audio/AAudio.hpp File Reference

#include "Interfaces/Audio/IAudio.hpp"
Include dependency graph for AAudio.hpp:



Classes

- interface cae::AAudio

  Abstract class for audio.

Namespaces

- namespace cae

43.20 AAudio.hpp

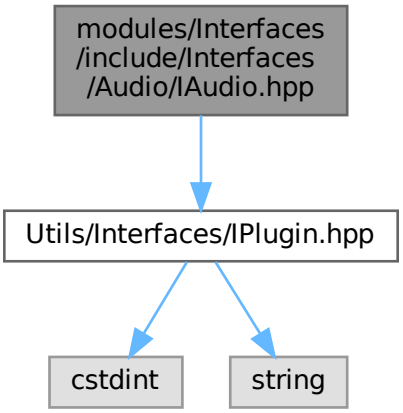
Go to the documentation of this file.

```
00001 ///
00002 /// @file IAudio.hpp
00003 /// @brief This file contains the audio abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Audio/IAudio.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface AAudio
00016     /// @brief Abstract class for audio
00017     /// @namespace cae
00018     ///
00019     class AAudio : public IAudio
00020     {
00021
```

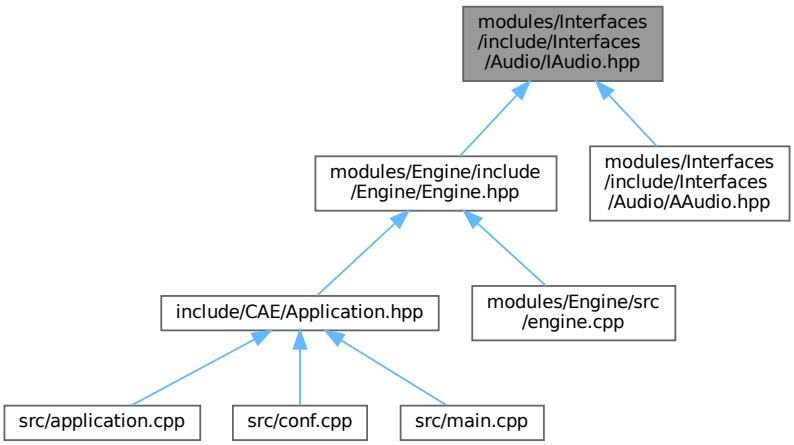
```
00022         public:
00023             ~AAudio() override = default;
00024
00025     }; // interface AAudio
00026
00027 } // namespace cae
```

43.21 modules/Interfaces/include/Interfaces/Audio/IAudio.hpp File Reference

This file contains the audio abstract class.
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IAudio.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IAudio

  Interface for audio.

Namespaces

- namespace cae

43.21.1 Detailed Description

This file contains the audio abstract class.
This file contains the audio interface.
Definition in file IAudio.hpp.

43.22 IAudio.hpp

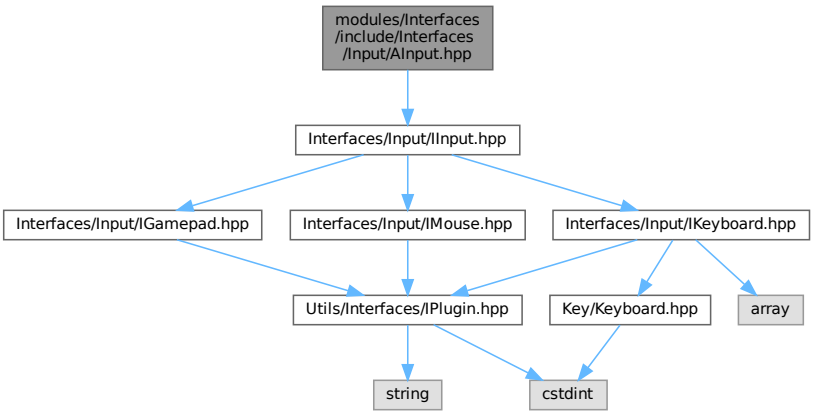
Go to the documentation of this file.

```
00001 ///
00002 /// @file IAudio.hpp
00003 /// @brief This file contains the audio interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Utils/Interfaces/IPlugin.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface IAudio
00016     /// @brief Interface for audio
00017     /// @namespace cae
00018     ///
00019     class IAudio : public utl::IPlugin
00020     {
00021
00022         public:
00023             ~IAudio() override = default;
00024
00025     }; // interface IAudio
00026
00027 } // namespace cae
```

43.23 modules/Interfaces/include/Interfaces/Input/AInput.hpp File Reference

This file contains the input abstract class.

#include ”Interfaces/Input/IInput.hpp”
Include dependency graph for AInput.hpp:



Classes

- interface cae::AInput

  Abstract class for inputs.

Namespaces

- namespace cae

43.23.1 Detailed Description

This file contains the input abstract class.
Definition in file AInput.hpp.

43.24 AInput.hpp

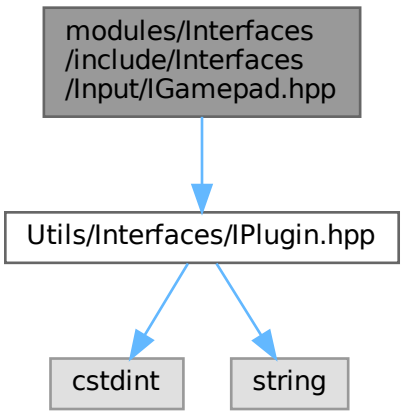
Go to the documentation of this file.

```
00001 ///
00002 /// @file AInput.hpp
00003 /// @brief This file contains the input abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include ”Interfaces/Input/IInput.hpp”
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface AInput
00016     /// @brief Abstract class for inputs
00017     /// @namespace cae
00018     ///
00019     class AInput : public IInput
00020     {
00021
00022         public:
00023             ~AInput() override = default;
00024
00025             const std::unique_ptr<IKeyboard> &getKeyboard() const override { return m_keyboard; }
00026             const std::unique_ptr<IMouse> &getMouse() const override { return m_mouse; }
00027             const std::vector<std::unique_ptr<IGamepad» &getGamepads() const override { return m_gamepads; }
00028
00029             void setGamepads(std::vector<std::unique_ptr<IGamepad» &gamepads) override
00030             {
```

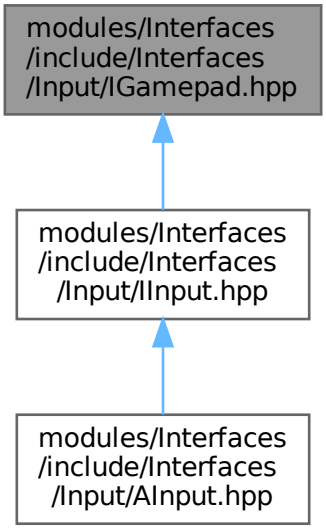
```
00031             m_gamepads = std::move(gamepads);
00032         }
00033         void setKeyboard(std::unique_ptr<IKeyboard> &keyboard) override { m_keyboard = std::move(keyboard); }
00034         void setMouse(std::unique_ptr<IMouse> &mouse) override { m_mouse = std::move(mouse); }
00035
00036     private:
00037         std::unique_ptr<IKeyboard> m_keyboard;
00038         std::unique_ptr<IMouse> m_mouse;
00039         std::vector<std::unique_ptr<IGamepad» m_gamepads;
00040 }; // interface AInput
00041
00042 } // namespace cae
```

43.25 modules/Interfaces/include/Interfaces/Input/IGamepad.hpp File Reference

This file contains the input gamepad interface.
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IGamepad.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IGamepad

  Interface for gamepad.

Namespaces

- namespace cae

43.25.1 Detailed Description

This file contains the input gamepad interface.
Definition in file IGamepad.hpp.

43.26 IGamepad.hpp

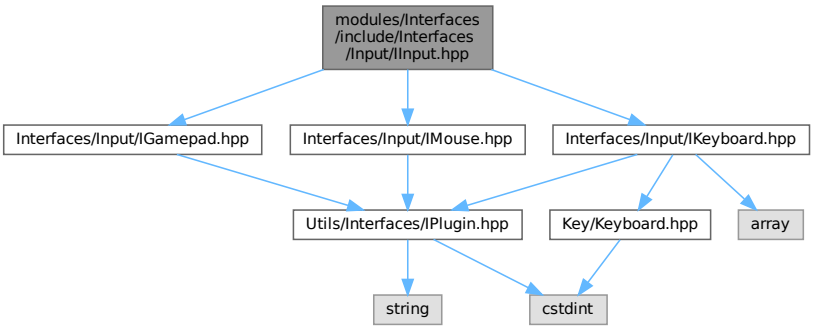
Go to the documentation of this file.

```
00001 ///
00002 /// @file IGamepad.hpp
00003 /// @brief This file contains the input gamepad interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Utils/Interfaces/IPlugin.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface IGamepad
00016     /// @brief Interface for gamepad
00017     /// @namespace cae
00018     ///
00019     class IGamepad : public utl::IPlugin
00020     {
00021
00022         public:
```

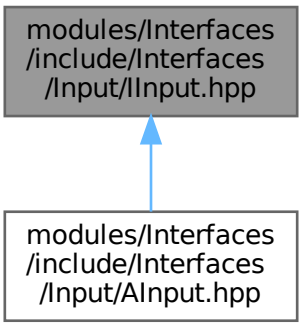
```
00023         ~IGamepad() override = default;
00024
00025     }; // interface IGamepad
00026
00027 } // namespace cae
```

43.27 modules/Interfaces/include/Interfaces/Input/IInput.hpp File Reference

This file contains the input interface.
#include "Interfaces/Input/IGamepad.hpp"
#include "Interfaces/Input/IKeyboard.hpp"
#include "Interfaces/Input/IMouse.hpp"
Include dependency graph for IInput.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IInput

  Interface for inputs.

Namespaces

- namespace cae

43.27.1 Detailed Description

This file contains the input interface.
Definition in file IInput.hpp.

43.28 IInput.hpp

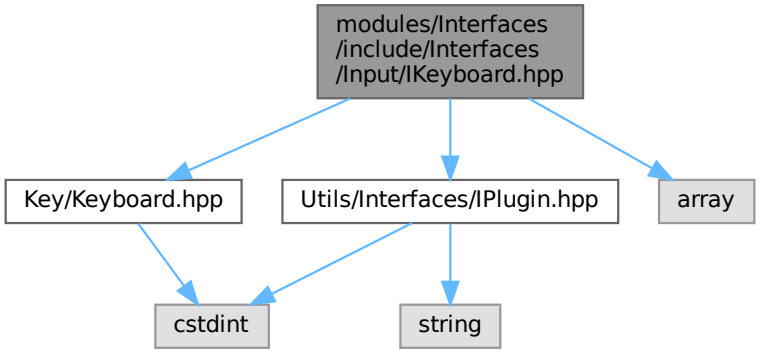
Go to the documentation of this file.

```
00001 ///
00002 /// @file IInput.hpp
00003 /// @brief This file contains the input interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Input/IGamepad.hpp"
00010 #include "Interfaces/Input/IKeyboard.hpp"
00011 #include "Interfaces/Input/IMouse.hpp"
00012
00013 namespace cae
00014 {
00015
00016     ///
00017     /// @interface IInput
00018     /// @brief Interface for inputs
00019     /// @namespace cae
00020     ///
00021     class IInput : public utl::IPlugin
00022     {
00023
00024         public:
00025             ~IInput() override = default;
00026
00027             virtual const std::unique_ptr<IKeyboard> &getKeyboard() const = 0;
00028             virtual const std::unique_ptr<IMouse> &getMouse() const = 0;
00029             virtual const std::vector<std::unique_ptr<IGamepad» &getGamepads() const = 0;
00030
00031             virtual void setGamepads(std::vector<std::unique_ptr<IGamepad» &gamepads) = 0;
00032             virtual void setKeyboard(std::unique_ptr<IKeyboard> &keyboard) = 0;
00033             virtual void setMouse(std::unique_ptr<IMouse> &mouse) = 0;
00034
00035     }; // interface IInput
00036
00037 } // namespace cae
```

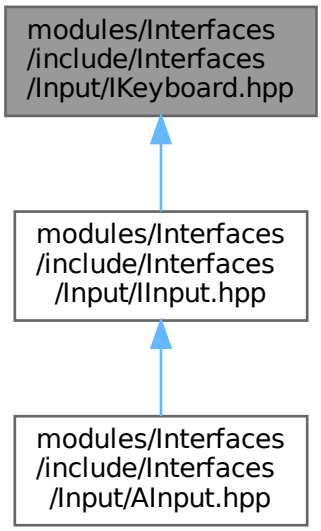
43.29 modules/Interfaces/include/Interfaces/Input/IKeyboard.hpp File Reference

This file contains the input keyboard interface.
#include "Key/Keyboard.hpp"
#include "Utils/Interfaces/IPlugin.hpp"
#include <array>

Include dependency graph for IKeyboard.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IKeyboard
  Interface for keyboard.

Namespaces

- namespace cae

43.29.1 Detailed Description

This file contains the input keyboard interface.

Definition in file IKeyboard.hpp.

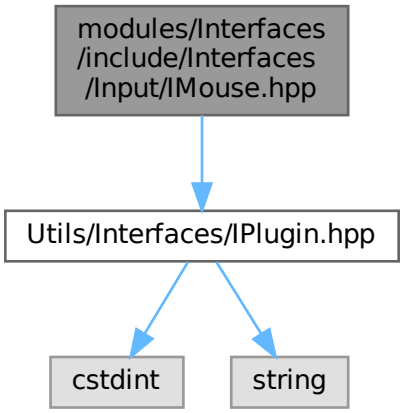
43.30 IKeyboard.hpp

Go to the documentation of this file.

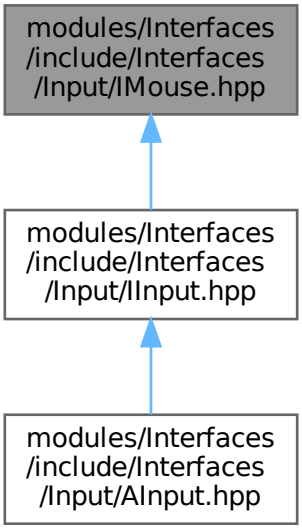
```
00001 ///
00002 /// @file IKeyboard.hpp
00003 /// @brief This file contains the input keyboard interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Key/Keyboard.hpp"
00010
00011 #include "Utils/Interfaces/IPlugin.hpp"
00012
00013 #include <array>
00014
00015 namespace cae
00016 {
00017
00018     ///
00019     /// @interface IKeyboard
00020     /// @brief Interface for keyboard
00021     /// @namespace cae
00022     ///
00023     class IKeyboard : public utl::IPlugin
00024     {
00025
00026         public:
00027             ~IKeyboard() override = default;
00028
00029             virtual bool isKeyPressed(KeyCode keyCode) const = 0;
00030
00031         private:
00032             std::array<KeyState, static_cast<size_t>(KeyCode::Count)> m_keyMap{};
00033     }; // interface IKeyboard
00034
00035 } // namespace cae
```

43.31 modules/Interfaces/include/Interfaces/Input/IMouse.hpp File Reference

This file contains the input mouse interface.
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IMouse.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IMouse

  Interface for mouse.

Namespaces

- namespace cae

43.31.1 Detailed Description

This file contains the input mouse interface.
Definition in file IMouse.hpp.

43.32 IMouse.hpp

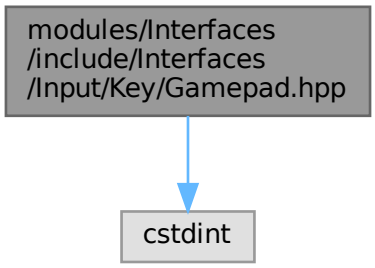
Go to the documentation of this file.

```
00001 ///
00002 /// @file IMouse.hpp
00003 /// @brief This file contains the input mouse interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Utils/Interfaces/IPlugin.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface IMouse
00016     /// @brief Interface for mouse
00017     /// @namespace cae
00018     ///
00019     class IMouse : public utl::IPlugin
00020     {
00021
00022         public:
```

```
00023           ~IMouse() override = default;
00024
00025     }; // interface IMouse
00026
00027 } // namespace cae
```

43.33 modules/Interfaces/include/Interfaces/Input/Key/Gamepad.hpp File Reference

This file contains the gamepad keys.
#include <cstdint>
Include dependency graph for Gamepad.hpp:



Namespaces

- namespace cae

Enumerations

- enum class cae::GamepadButton : uint8_t {
  cae::A = 0 , cae::B , cae::X , cae::Y ,
  cae::Back , cae::Guide , cae::Start , cae::LThumb ,
  cae::RThumb , cae::LShoulder , cae::RShoulder , cae::DPadUp ,
  cae::DPadDown , cae::DPadLeft , cae::DPadRight }
- enum class cae::GamepadAxis : uint8_t {
  cae::LeftX = 0 , cae::LeftY , cae::RightX , cae::RightY ,
  cae::TriggerLeft , cae::TriggerRight }

43.33.1 Detailed Description

This file contains the gamepad keys.
Definition in file Gamepad.hpp.

43.34 Gamepad.hpp

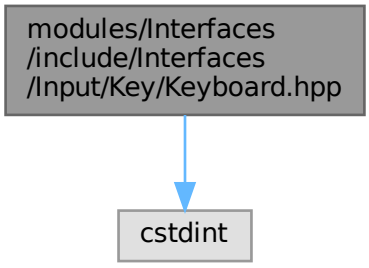
Go to the documentation of this file.

```
00001 ///
00002 /// @file Gamepad.hpp
00003 /// @brief This file contains the gamepad keys
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include <cstdint>
```

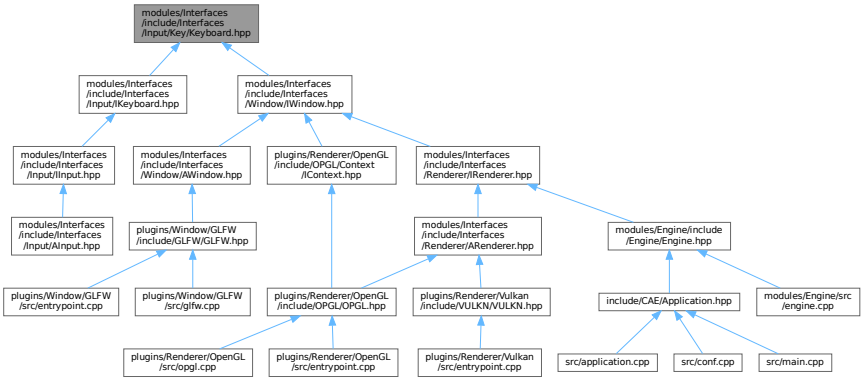
```
00010
00011 namespace cae
00012 {
00013     enum class GamepadButton : uint8_t
00014     {
00015         A = 0,
00016         B,
00017         X,
00018         Y,
00019         Back,
00020         Guide,
00021         Start,
00022         LThumb,
00023         RThumb,
00024         LShoulder,
00025         RShoulder,
00026         DPadUp,
00027         DPadDown,
00028         DPadLeft,
00029         DPadRight
00030     };
00031
00032     enum class GamepadAxis : uint8_t
00033     {
00034         LeftX = 0,
00035         LeftY,
00036         RightX,
00037         RightY,
00038         TriggerLeft,
00039         TriggerRight
00040     };
00041 } // namespace cae
```

43.35 modules/Interfaces/include/Interfaces/Input/Key/Keyboard.hpp File Reference

This file contains the keyboard keys.
#include <cstdint>
Include dependency graph for Keyboard.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace cae

Enumerations

- enum class cae::KeyState : std::uint8_t { cae::Pressed = 0 , cae::Released = 1 , cae::Held = 2 , cae::Toggled = 3 }
- enum class cae::KeyCode : uint8_t {
  cae::A , cae::B , cae::C , cae::D ,
  cae::E , cae::F , cae::G , cae::H ,
  cae::I , cae::J , cae::K , cae::L ,
  cae::M , cae::N , cae::O , cae::P ,
  cae::Q , cae::R , cae::S , cae::T ,
  cae::U , cae::V , cae::W , cae::X ,
  cae::Y , cae::Z , cae::Num0 , cae::Num1 ,
  cae::Num2 , cae::Num3 , cae::Num4 , cae::Num5 ,
  cae::Num6 , cae::Num7 , cae::Num8 , cae::Num9 ,
  cae::Escape , cae::F1 , cae::F2 , cae::F3 ,
  cae::F4 , cae::F5 , cae::F6 , cae::F7 ,
  cae::F8 , cae::F9 , cae::F10 , cae::F11 ,
  cae::F12 , cae::Left , cae::Right , cae::Up ,
  cae::Down , cae::Home , cae::End , cae::PageUp ,
  cae::PageDown , cae::Insert , cae::Delete , cae::Backspace ,
  cae::Tab , cae::Enter , cae::Space , cae::LShift ,
  cae::RShift , cae::LCtrl , cae::RCtrl , cae::LAlt ,
  cae::RAlt , cae::LSuper , cae::RSuper , cae::Numpad0 ,
  cae::Numpad1 , cae::Numpad2 , cae::Numpad3 , cae::Numpad4 ,
  cae::Numpad5 , cae::Numpad6 , cae::Numpad7 , cae::Numpad8 ,
  cae::Numpad9 , cae::NumpadAdd , cae::NumpadSubtract , cae::NumpadMultiply ,
  cae::NumpadDivide , cae::CapsLock , cae::NumLock , cae::ScrollLock ,
  cae::PrintScreen , cae::Pause , cae::Menu , cae::Count }

43.35.1 Detailed Description

This file contains the keyboard keys.
Definition in file Keyboard.hpp.

43.36 Keyboard.hpp

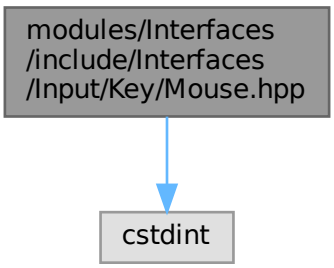
Go to the documentation of this file.

```
00001 ///
00002 /// @file Keyboard.hpp
00003 /// @brief This file contains the keyboard keys
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include <cstdint>
00010
00011 namespace cae
00012 {
00013
00014     enum class KeyState : std::uint8_t
00015     {
00016         Pressed = 0,
00017         Released = 1,
00018         Held = 2,
00019         Toggled = 3,
00020     };
00021
00022     enum class KeyCode : uint8_t
00023     {
00024         A,
00025         B,
00026         C,
00027         D,
00028         E,
00029         F,
00030         G,
00031         H,
00032         I,
00033         J,
00034         K,
00035         L,
00036         M,
00037         N,
00038         O,
00039         P,
00040         Q,
00041         R,
00042         S,
00043         T,
00044         U,
00045         V,
00046         W,
00047         X,
00048         Y,
00049         Z,
00050
00051         Num0,
00052         Num1,
00053         Num2,
00054         Num3,
00055         Num4,
00056         Num5,
00057         Num6,
00058         Num7,
00059         Num8,
00060         Num9,
00061
00062         Escape,
00063         F1,
00064         F2,
00065         F3,
00066         F4,
00067         F5,
00068         F6,
00069         F7,
00070         F8,
00071         F9,
00072         F10,
00073         F11,
00074         F12,
00075
00076         Left,
00077         Right,
00078         Up,
00079         Down,
00080         Home,
00081         End,
00082         PageUp,
00083         PageDown,
00084         Insert,
00085         Delete,
00086         Backspace,
00087         Tab,
```

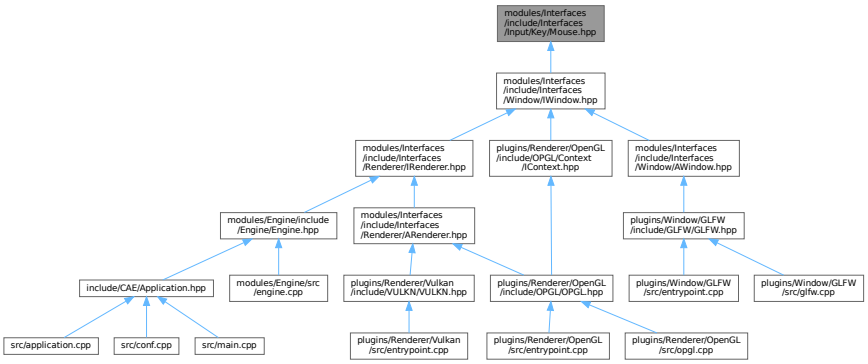
```
00088         Enter,
00089         Space,
00090
00091         LShift,
00092         RShift,
00093         LCtrl,
00094         RCtrl,
00095         LAlt,
00096         RAlt,
00097         LSuper,
00098         RSuper,
00099
00100         Numpad0,
00101         Numpad1,
00102         Numpad2,
00103         Numpad3,
00104         Numpad4,
00105         Numpad5,
00106         Numpad6,
00107         Numpad7,
00108         Numpad8,
00109         Numpad9,
00110         NumpadAdd,
00111         NumpadSubtract,
00112         NumpadMultiply,
00113         NumpadDivide,
00114
00115         CapsLock,
00116         NumLock,
00117         ScrollLock,
00118
00119         PrintScreen,
00120         Pause,
00121         Menu,
00122
00123         Count
00124     };
00125 } // namespace cae
```

43.37 modules/Interfaces/include/Interfaces/Input/Key/Mouse.hpp File Reference

This file contains the mouse keys.
#include <cstdint>
Include dependency graph for Mouse.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace cae

Enumerations

- enum class cae::MouseButton : uint8_t { cae::Left = 0 , cae::Right , cae::Middle , cae::XButton1 , cae::XButton2 , cae::WheelUp , cae::WheelDown , cae::Count }

43.37.1 Detailed Description

This file contains the mouse keys.
Definition in file Mouse.hpp.

43.38 Mouse.hpp

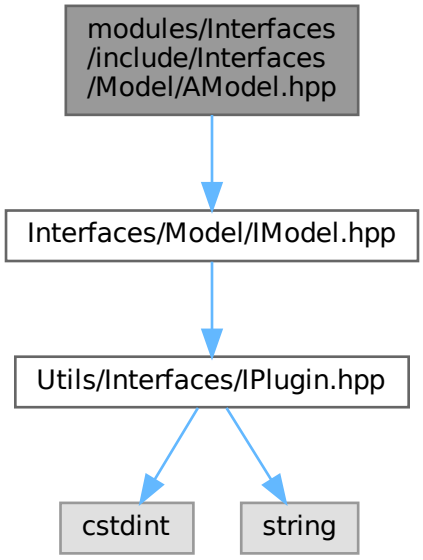
Go to the documentation of this file.

```
00001 ///
00002 /// @file Mouse.hpp
00003 /// @brief This file contains the mouse keys
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include <cstdint>
00010
00011 namespace cae
00012 {
00013     enum class MouseButton : uint8_t
00014     {
00015         Left = 0,
00016         Right,
00017         Middle,
00018         XButton1,
00019         XButton2,
00020         WheelUp,
00021         WheelDown,
00022         Count
00023     };
00024 } // namespace cae
```

43.39 modules/Interfaces/include/Interfaces/Model/AModel.hpp File Reference

This file contains the model abstract class.

#include ”Interfaces/Model/IModel.hpp”
Include dependency graph for AModel.hpp:



Classes

- interface cae::AModel

  Abstract class for model.

Namespaces

- namespace cae

43.39.1 Detailed Description

This file contains the model abstract class.
Definition in file AModel.hpp.

43.40 AModel.hpp

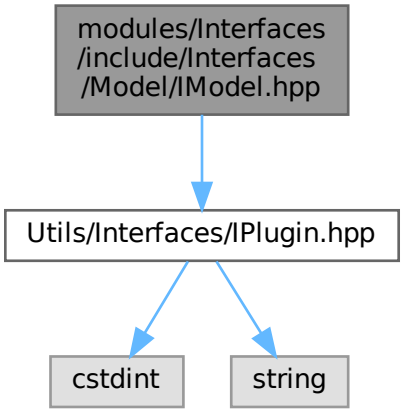
Go to the documentation of this file.

```
00001 ///
00002 /// @file AModel.hpp
00003 /// @brief This file contains the model abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include ”Interfaces/Model/IModel.hpp”
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface AModel
00016     /// @brief Abstract class for model
00017     /// @namespace cae
00018     ///
00019     class AModel : public IModel
```

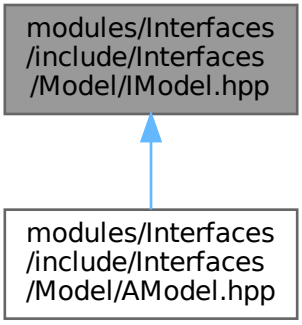
```
00020     {
00021
00022         public:
00023             ~AModel() override = default;
00024
00025     }; // interface AModel
00026
00027 } // namespace cae
```

43.41 modules/Interfaces/include/Interfaces/Model/IModel.hpp File Reference

This file contains the model interface.
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IModel.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IModel

  Interface for model.

Namespaces

- namespace cae

43.41.1 Detailed Description

This file contains the model interface.
Definition in file IModel.hpp.

43.42 IModel.hpp

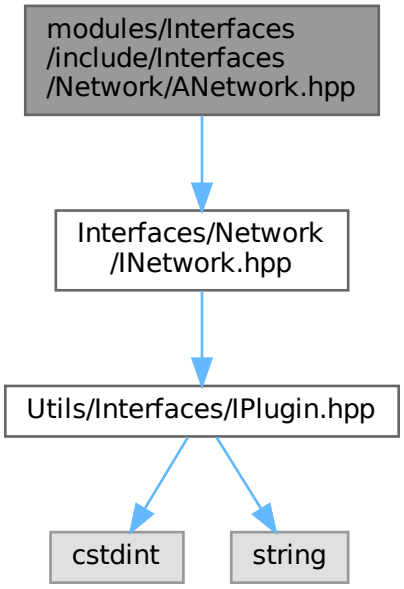
Go to the documentation of this file.

```
00001 ///
00002 /// @file IModel.hpp
00003 /// @brief This file contains the model interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Utils/Interfaces/IPlugin.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface IModel
00016     /// @brief Interface for model
00017     /// @namespace cae
00018     ///
00019     class IModel : public utl::IPlugin
00020     {
00021
00022         public:
00023             ~IModel() override = default;
00024
00025     }; // interface IModel
00026
00027 } // namespace cae
```

43.43 modules/Interfaces/include/Interfaces/Network/ANetwork.hpp File Reference

This file contains the network abstract class.

#include ”Interfaces/Network/INetwork.hpp”
Include dependency graph for ANetwork.hpp:



Classes

- interface cae::ANetwork

  Abstract class for network.

Namespaces

- namespace cae

43.43.1 Detailed Description

This file contains the network abstract class.
Definition in file ANetwork.hpp.

43.44 ANetwork.hpp

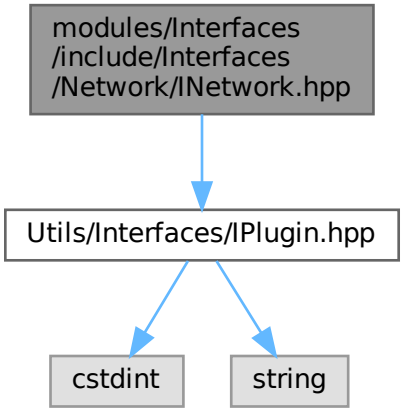
Go to the documentation of this file.

```
00001 ///
00002 /// @file ANetwork.hpp
00003 /// @brief This file contains the network abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include ”Interfaces/Network/INetwork.hpp”
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface ANetwork
00016     /// @brief Abstract class for network
00017     /// @namespace cae
```

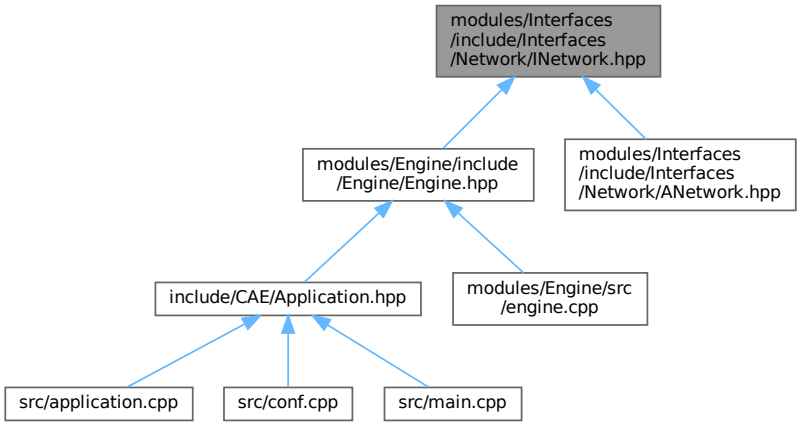
```
00018     ///
00019     class ANetwork : public INetwork
00020     {
00021
00022         public:
00023             ~ANetwork() override = default;
00024
00025     }; // interface ANetwork
00026
00027 } // namespace cae
```

43.45 modules/Interfaces/include/Interfaces/Network/INetwork.hpp File Reference

This file contains the network interface.
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for INetwork.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::INetwork

  Interface for network.

Namespaces

- namespace cae

43.45.1 Detailed Description

This file contains the network interface.
Definition in file INetwork.hpp.

43.46 INetwork.hpp

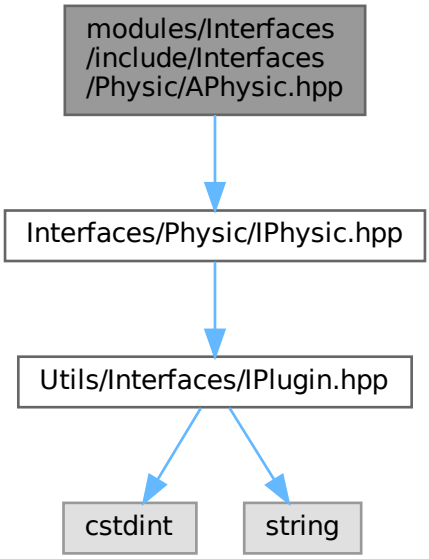
Go to the documentation of this file.

```
00001 ///
00002 /// @file INetwork.hpp
00003 /// @brief This file contains the network interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Utils/Interfaces/IPlugin.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface INetwork
00016     /// @brief Interface for network
00017     /// @namespace cae
00018     ///
00019     class INetwork : public utl::IPlugin
00020     {
00021
00022         public:
00023             ~INetwork() override = default;
00024
00025             virtual bool connect(const std::string &host, uint16_t port) = 0;
00026
00027     }; // interface INetwork
00028
00029 } // namespace cae
```

43.47 modules/Interfaces/include/Interfaces/Physic/APhysic.hpp File Reference

This file contains the physic abstract class.

#include ”Interfaces/Physic/IPhysic.hpp”
Include dependency graph for APhysic.hpp:



Classes

- interface cae::APhysic

  Abstract class for physic.

Namespaces

- namespace cae

43.47.1 Detailed Description

This file contains the physic abstract class.
Definition in file APhysic.hpp.

43.48 APhysic.hpp

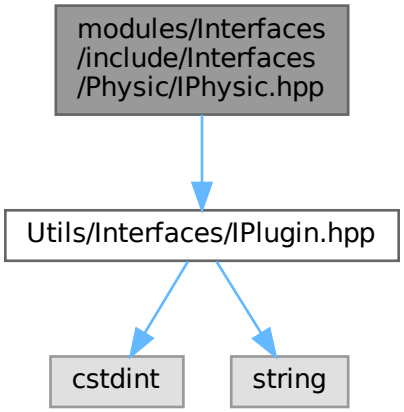
Go to the documentation of this file.

```
00001 ///
00002 /// @file APhysic.hpp
00003 /// @brief This file contains the physic abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include ”Interfaces/Physic/IPhysic.hpp”
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface APhysic
00016     /// @brief Abstract class for physic
00017     /// @namespace cae
00018     ///
00019     class APhysic : public IPhysic
```

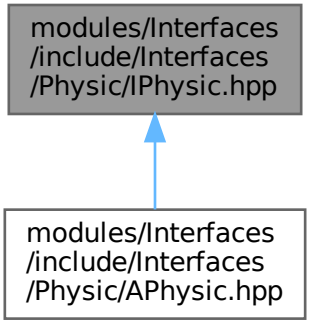
```
00020     {
00021
00022         public:
00023             ~APhysic() override = default;
00024
00025     }; // interface APhysic
00026
00027 } // namespace cae
```

43.49 modules/Interfaces/include/Interfaces/Physic/IPhysic.hpp File Reference

This file contains the physic interface.
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IPhysic.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IPhysic

  Interface for physics.

Namespaces

- namespace cae

43.49.1 Detailed Description

This file contains the physic interface.
Definition in file IPhysic.hpp.

43.50 IPhysic.hpp

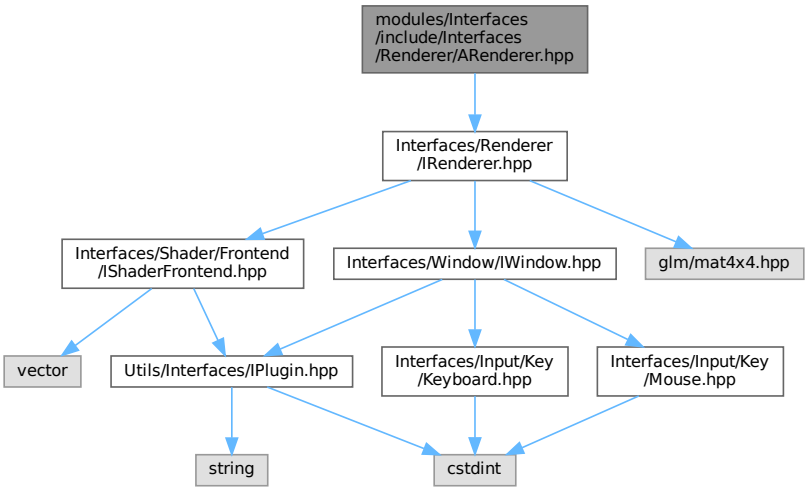
Go to the documentation of this file.

```
00001 ///
00002 /// @file IPhysic.hpp
00003 /// @brief This file contains the physic interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Utils/Interfaces/IPlugin.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface IPhysic
00016     /// @brief Interface for physics
00017     /// @namespace cae
00018     ///
00019     class IPhysic : public utl::IPlugin
00020     {
00021
00022         public:
00023             ~IPhysic() override = default;
00024
00025     }; // interface IPhysic
00026
00027 } // namespace cae
```

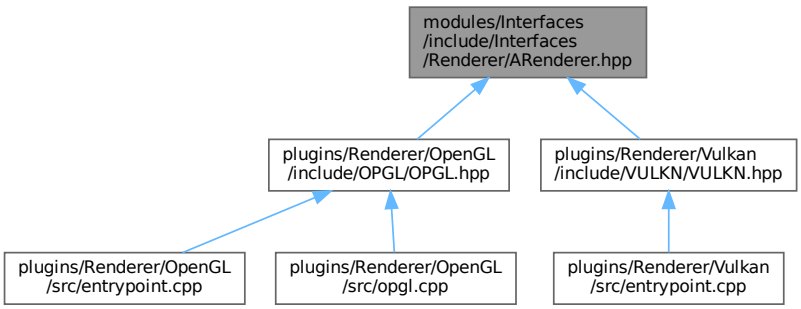
43.51 modules/Interfaces/include/Interfaces/Renderer/ARenderer.hpp File Reference

This file contains the Renderer abstract class.

#include ”Interfaces/Renderer/IRenderer.hpp”
Include dependency graph for ARenderer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::ARenderer

  Abstract class for renderer.

Namespaces

- namespace cae

43.51.1 Detailed Description

This file contains the Renderer abstract class.
Definition in file ARenderer.hpp.

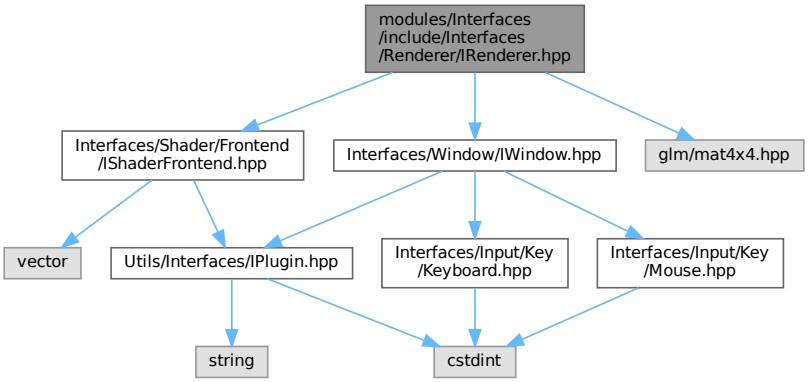
43.52 ARenderer.hpp

Go to the documentation of this file.

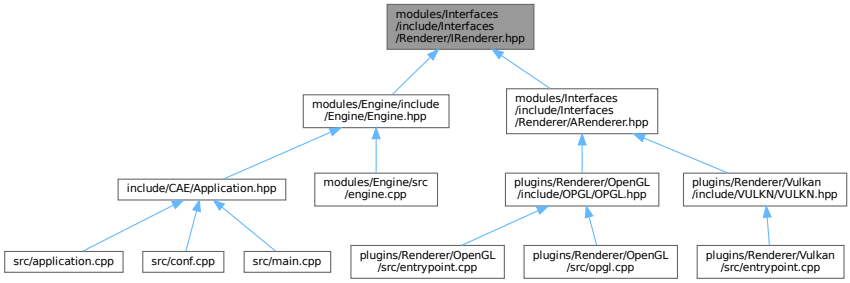
```
00001 ///
00002 /// @file ARenderer.hpp
00003 /// @brief This file contains the Renderer abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Renderer/IRenderer.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface ARenderer
00016     /// @brief Abstract class for renderer
00017     /// @namespace cae
00018     ///
00019     class ARenderer : public IRenderer
00020     {
00021
00022         public:
00023             ~ARenderer() override = default;
00024
00025     }; // interface ARenderer
00026
00027 } // namespace cae
```

43.53 modules/Interfaces/include/Interfaces/Renderer/IRenderer.hpp File Reference

This file contains the Renderer interface.
#include "Interfaces/Shader/Frontend/IShaderFrontend.hpp"
#include "Interfaces/Window/IWindow.hpp"
#include <glm/mat4x4.hpp>
Include dependency graph for IRenderer.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::Color

  Struct for color.
- interface cae::IRenderer

  Interface for renderer.

Namespaces

- namespace cae

43.53.1 Detailed Description

This file contains the Renderer interface.
Definition in file IRenderer.hpp.

43.54 IRenderer.hpp

Go to the documentation of this file.

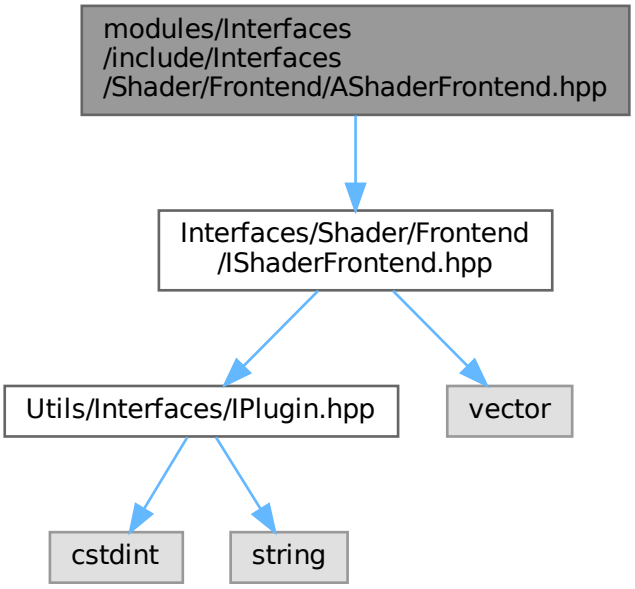
```
00001 ///
00002 /// @file IRenderer.hpp
00003 /// @brief This file contains the Renderer interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Shader/Frontend/IShaderFrontend.hpp"
00010 #include "Interfaces/Window/IWindow.hpp"
00011
00012 #include <glm/mat4x4.hpp>
00013
00014 namespace cae
00015 {
00016
00017     ///
00018     /// @struct Color
00019     /// @brief Struct for color
00020     /// @namespace cae
00021     ///
00022     struct Color
00023     {
00024         float r;
00025         float g;
00026         float b;
00027         float a;
00028     };
00029
00030     ///
00031     /// @interface IRenderer
00032     /// @brief Interface for renderer
00033     /// @namespace cae
00034     ///
00035     class IRenderer : public utl::IPlugin
00036     {
```

```
00037
00038         public:
00039             ~IRenderer() override = default;
00040
00041             ///
00042             /// @param enabled Whether VSync is enabled
00043             /// @brief Enable or disable VSync
00044             ///
00045             virtual void setVSyncEnabled(bool enabled) = 0;
00046
00047             ///
00048             /// @param color Clear color to set
00049             /// @brief Set the clear color
00050             ///
00051             virtual void setClearColor(const Color &color) = 0;
00052
00053             ///
00054             /// @return Whether VSync is enabled
00055             /// @brief Check if VSync is enabled
00056             ///
00057             [[nodiscard]] virtual bool isVSyncEnabled() const = 0;
00058
00059             ///
00060             /// @param nativeWindowHandle Native window handle
00061             /// @param clearColor Clear color (default: white)
00062             /// @brief Initialize the renderer with a native window handle and clear color
00063             ///
00064             virtual void initialize(const NativeWindowHandle &nativeWindowHandle,
00065                                     const Color &clearColor = {.r = 1.F, .g = 1.F, .b = 1.F, .a = 1.F}) = 0;
00066
00067             ///
00068             /// @param id Shader ID
00069             /// @param vertex Vertex shader IR module
00070             /// @param fragment Fragment shader IR module
00071             /// @brief Create a rendering pipeline with vertex and fragment shaders
00072             ///
00073             virtual void createPipeline(const ShaderID &id, const ShaderIRModule &vertex,
00074                                         const ShaderIRModule &fragment) = 0;
00075
00076             ///
00077             /// @param windowSize Current window size
00078             /// @param shaderId Shader ID to use for drawing
00079             /// @param mvp Model-View-Projection matrix
00080             /// @brief Draw the scene using the specified shader and window size
00081             ///
00082             virtual void draw(const WindowSize &windowSize, const ShaderID &shaderId, glm::mat4 mvp) = 0;
00083
00084             ///
00085             /// @param vertices Vertex data to create the mesh
00086             /// @brief Create a mesh with the given vertex data
00087             ///
00088             virtual void createMesh(const std::vector<float> &vertices) = 0;
00089
00090     }; // interface IRenderer
00091
00092 } // namespace cae
```

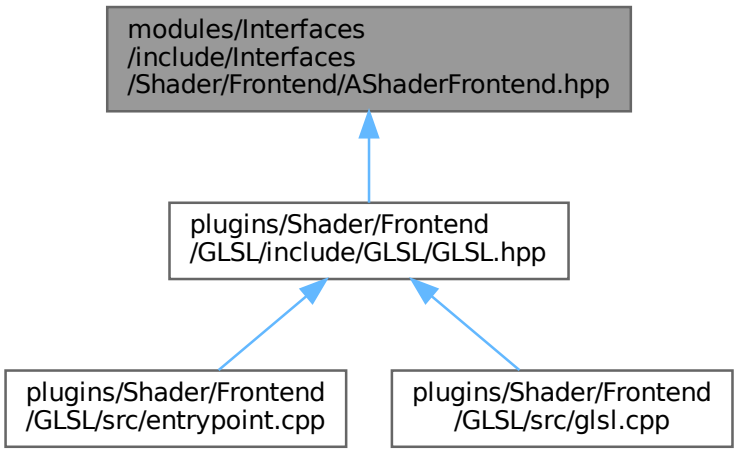
43.55 modules/Interfaces/include/Interfaces/Shader/Frontend/↩ AShaderFrontend.hpp File Reference

This file contains the ShaderFrontend abstract class.

#include "Interfaces/Shader/Frontend/IShaderFrontend.hpp"
Include dependency graph for AShaderFrontend.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::AShaderFrontend

  Abstract class for shader frontend.

Namespaces

- namespace cae

43.55.1 Detailed Description

This file contains the ShaderFrontend abstract class.
Definition in file AShaderFrontend.hpp.

43.56 AShaderFrontend.hpp

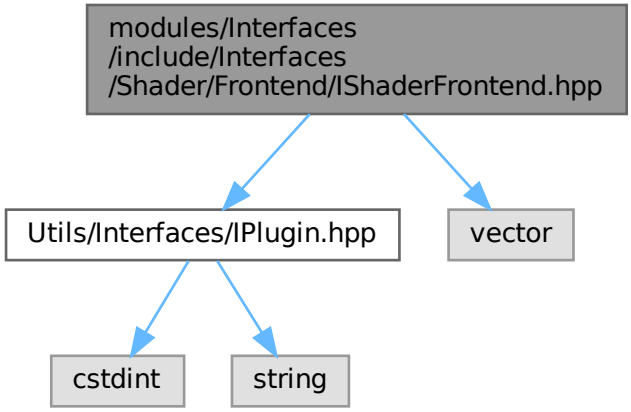
Go to the documentation of this file.

```
00001 ///
00002 /// @file AShaderFrontend.hpp
00003 /// @brief This file contains the ShaderFrontend abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Shader/Frontend/IShaderFrontend.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface AShaderFrontend
00016     /// @brief Abstract class for shader frontend
00017     /// @namespace cae
00018     ///
00019     class AShaderFrontend : public IShaderFrontend
00020     {
00021
00022         public:
00023             ~AShaderFrontend() override = default;
00024
00025     }; // interface AShaderFrontend
00026
00027 } // namespace cae
```

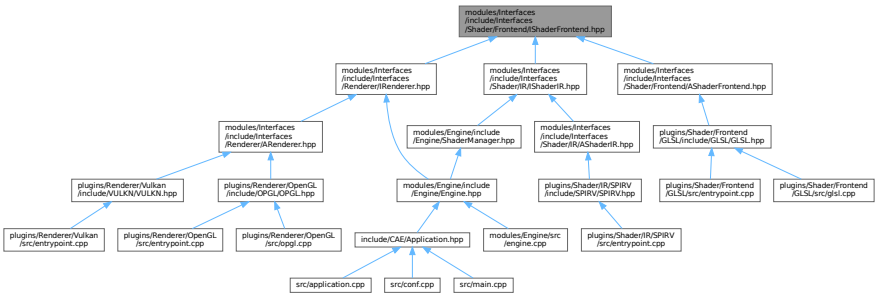
43.57 modules/Interfaces/include/Interfaces/Shader/Frontend/IShader↩ Frontend.hpp File Reference

This file contains the ShaderFrontend interface.
#include "Utils/Interfaces/IPlugin.hpp"
#include <vector>

Include dependency graph for IShaderFrontend.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::ShaderSourceDesc

  Struct for shader source description.
- struct cae::ShaderIRModule

  Struct for shader intermediate representation module.
- struct cae::ShaderPipelineDesc

  Struct for shader pipeline description.
- interface cae::IShaderFrontend

  Interface for shaders frontend.

Namespaces

- namespace cae

Typedefs

- using cae::ShaderID = std::string

Enumerations

- enum class cae::ShaderSourceType : uint8_t { cae::GLSL = 0 , cae::HLSL = 1 , cae::WGSL = 2 , cae::MSL = 3 , cae::SPIRV = 4 , cae::UNDEFINED = 255 }
- enum class cae::ShaderStage : uint8_t { cae::VERTEX = 0 , cae::FRAGMENT = 1 , cae::GEOMETRY = 2 , cae::COMPUTE = 3 , cae::UNDEFINED = 255 }

43.57.1 Detailed Description

This file contains the ShaderFrontend interface.
Definition in file IShaderFrontend.hpp.

43.58 IShaderFrontend.hpp

Go to the documentation of this file.

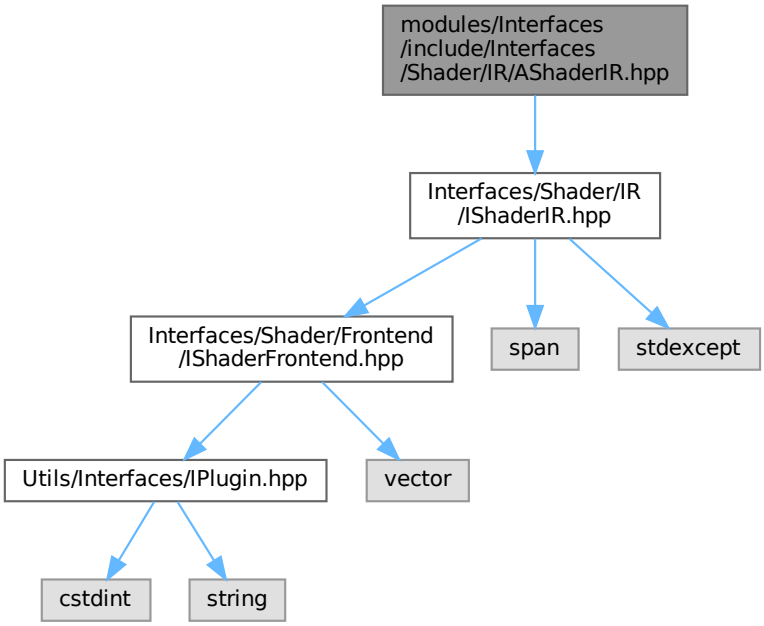
```
00001 ///
00002 /// @file IShaderFrontend.hpp
00003 /// @brief This file contains the ShaderFrontend interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Utils/Interfaces/IPlugin.hpp"
00010
00011 #include <vector>
00012
00013 namespace cae
00014 {
00015
00016     using ShaderID = std::string;
00017
00018     enum class ShaderSourceType : uint8_t
00019     {
00020         GLSL = 0,
00021         HLSL = 1,
00022         WGSL = 2,
00023         MSL = 3,
00024         SPIRV = 4,
00025         UNDEFINED = 255
00026     };
00027
00028     enum class ShaderStage : uint8_t
00029     {
00030         VERTEX = 0,
00031         FRAGMENT = 1,
00032         GEOMETRY = 2,
00033         COMPUTE = 3,
00034         UNDEFINED = 255
00035     };
00036
00037     ///
00038     /// @struct ShaderSourceDesc
00039     /// @brief Struct for shader source description
00040     /// @namespace cae
00041     ///
00042     struct ShaderSourceDesc
00043     {
00044         ShaderID id;
00045         ShaderSourceType type;
00046         std::string source;
00047         ShaderStage stage;
00048     };
00049
00050     ///
00051     /// @struct ShaderIRModule
00052     /// @brief Struct for shader intermediate representation module
00053     /// @namespace cae
00054     ///
00055     struct ShaderIRModule
00056     {
00057         ShaderID id;
00058         ShaderStage stage;
00059         std::vector<uint32_t> spirv;
00060         std::string entryPoint = "main";
00061     };
00062
```

```
00063     ///
00064     /// @struct ShaderPipelineDesc
00065     /// @brief Struct for shader pipeline description
00066     /// @namespace cae
00067     ///
00068     struct ShaderPipelineDesc
00069     {
00070         ShaderID id;
00071         ShaderID vertex;
00072         ShaderID fragment;
00073     };
00074
00075     ///
00076     /// @interface IShaderFrontend
00077     /// @brief Interface for shaders frontend
00078     /// @namespace cae
00079     ///
00080     class IShaderFrontend : public utl::IPlugin
00081     {
00082
00083         public:
00084             ~IShaderFrontend() override = default;
00085
00086             ///
00087             /// @return The source type this frontend handles
00088             /// @brief Get the source type this frontend handles
00089             ///
00090             virtual ShaderSourceType sourceType() const = 0;
00091
00092             ///
00093             /// @param desc Shader source description
00094             /// @return Compiled shader intermediate representation module
00095             /// @brief Compile shader source to intermediate representation
00096             ///
00097             virtual ShaderIRModule compile(const ShaderSourceDesc &desc) = 0;
00098
00099     }; // interface IShaderFrontend
00100
00101 } // namespace cae
```

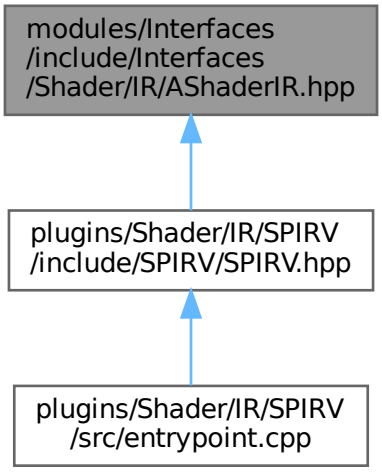
43.59 modules/Interfaces/include/Interfaces/Shader/IR/AShaderIR.hpp File Reference

This file contains the ShaderIR abstract class.

#include ”Interfaces/Shader/IR/IShaderIR.hpp”
Include dependency graph for AShaderIR.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::AShaderIR

  Abstract class for shader IR.

Namespaces

- namespace cae

43.59.1 Detailed Description

This file contains the ShaderIR abstract class.
Definition in file AShaderIR.hpp.

43.60 AShaderIR.hpp

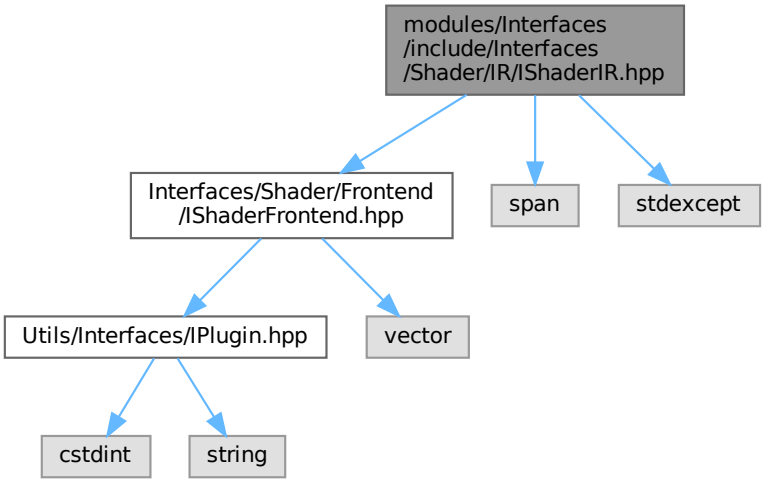
Go to the documentation of this file.

```
00001 ///
00002 /// @file AShaderIR.hpp
00003 /// @brief This file contains the ShaderIR abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Shader/IR/IShaderIR.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface AShaderIR
00016     /// @brief Abstract class for shader IR
00017     /// @namespace cae
00018     ///
00019     class AShaderIR : public IShaderIR
00020     {
00021
00022         public:
00023             ~AShaderIR() override = default;
00024
00025     }; // interface AShaderIR
00026
00027 } // namespace cae
```

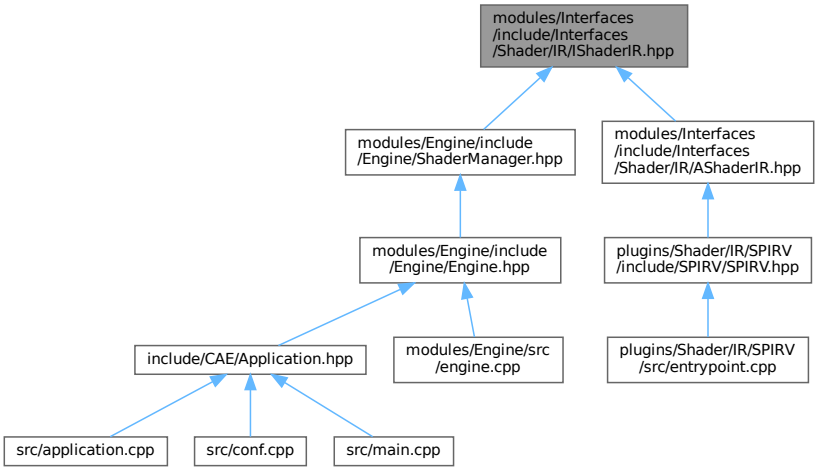
43.61 modules/Interfaces/include/Interfaces/Shader/IR/IShaderIR.hpp File Reference

This file contains the ShaderIR interface.
#include "Interfaces/Shader/Frontend/IShaderFrontend.hpp"
#include
#include <stdexcept>

Include dependency graph for IShaderIR.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IShaderIR

  Interface for shaders IR.

Namespaces

- namespace cae

43.61.1 Detailed Description

This file contains the ShaderIR interface.
Definition in file IShaderIR.hpp.

43.62 IShaderIR.hpp

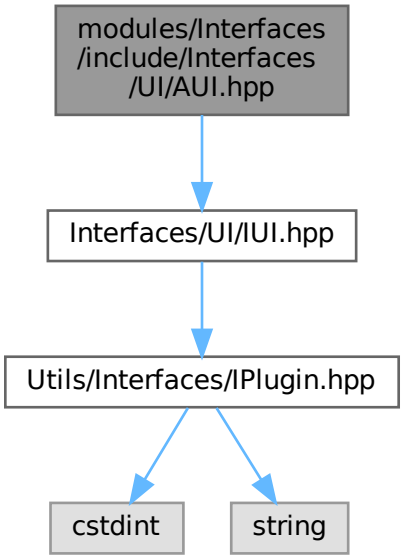
Go to the documentation of this file.

```
00001 ///
00002 /// @file IShaderIR.hpp
00003 /// @brief This file contains the ShaderIR interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Shader/Frontend/IShaderFrontend.hpp"
00010
00011 #include <span>
00012 #include <stdexcept>
00013
00014 namespace cae
00015 {
00016
00017     ///
00018     /// @interface IShaderIR
00019     /// @brief Interface for shaders IR
00020     /// @namespace cae
00021     ///
00022     class IShaderIR : public utl::IPlugin
00023     {
00024
00025         public:
00026             ~IShaderIR() override = default;
00027
00028             ///
00029             /// @return The IR type this processor handles
00030             /// @brief Get the IR type this processor handles
00031             ///
00032             virtual ShaderSourceType irType() const = 0;
00033
00034             ///
00035             /// @brief Transform or validate a shader IR module
00036             /// @param module The input IR module
00037             /// @return Transformed IR module ready for the backend
00038             ///
00039             virtual ShaderIRModule process(const ShaderIRModule &module) = 0;
00040
00041             ///
00042             /// @brief Optional: optimize a batch of IR modules
00043             ///
00044             virtual void optimize(std::span<ShaderIRModule> modules) { /* default: no-op */ }
00045
00046             ///
00047             /// @brief Optional: cross-compile from one IR to another (SPIR-V -> MSL, etc.)
00048             ///
00049             virtual ShaderIRModule crossCompile(const ShaderIRModule &module, ShaderSourceType targetType)
00050             {
00051                 throw std::runtime_error("Cross-compilation not implemented");
00052             }
00053
00054     }; // interface IShaderIR
00055
00056 } // namespace cae
```

43.63 modules/Interfaces/include/Interfaces/UI/AUI.hpp File Reference

This file contains the ui abstract class.

#include ”Interfaces/UI/IUI.hpp”
Include dependency graph for AUI.hpp:



Classes

- interface cae::AUI

  Abstract class for ui.

Namespaces

- namespace cae

43.63.1 Detailed Description

This file contains the ui abstract class.
Definition in file AUI.hpp.

43.64 AUI.hpp

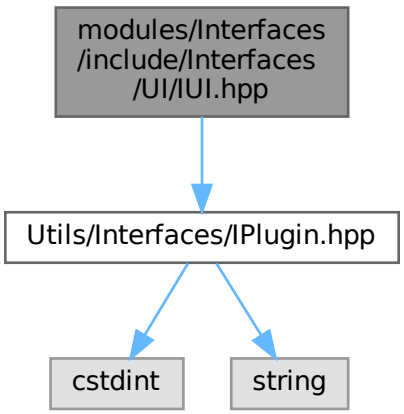
Go to the documentation of this file.

```
00001 ///
00002 /// @file AUI.hpp
00003 /// @brief This file contains the ui abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include ”Interfaces/UI/IUI.hpp”
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface AUI
00016     /// @brief Abstract class for ui
00017     /// @namespace cae
00018     ///
00019     class AUI : public IUI
```

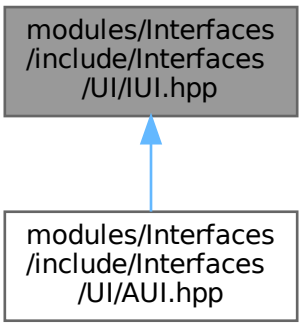
```
00020     {
00021
00022         public:
00023             ~AUI() override = default;
00024
00025     }; // interface AUI
00026
00027 } // namespace cae
```

43.65 modules/Interfaces/include/Interfaces/UI/IUI.hpp File Reference

This file contains the ui interface.
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IUI.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IUI
  Interface for ui.

Namespaces

- namespace cae

43.65.1 Detailed Description

This file contains the ui interface.
Definition in file IUI.hpp.

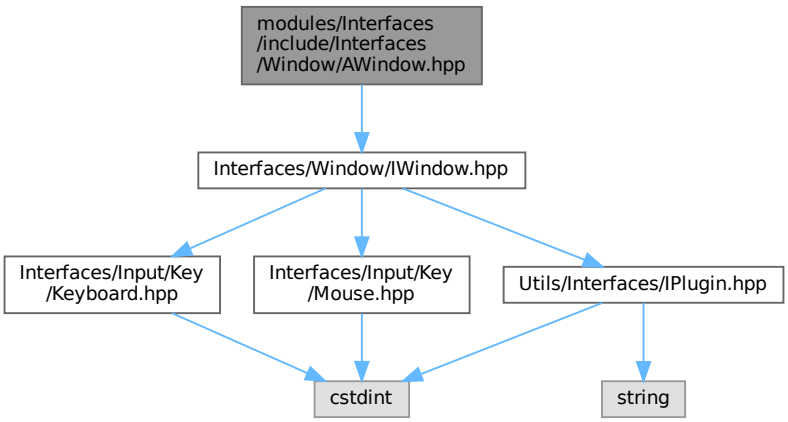
43.66 IUI.hpp

Go to the documentation of this file.

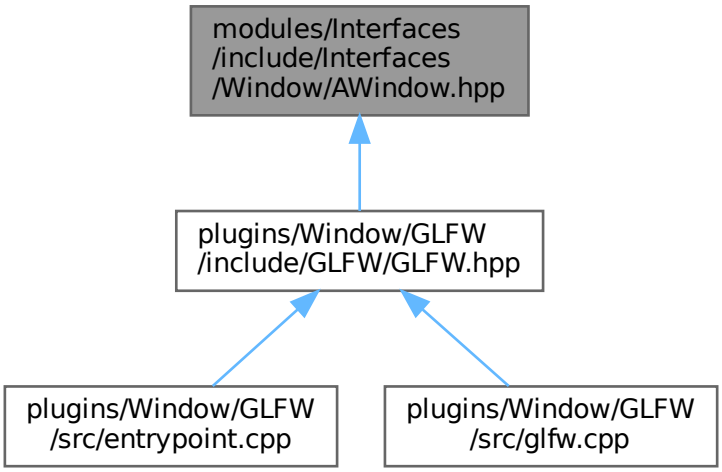
```
00001 ///
00002 /// @file IUI.hpp
00003 /// @brief This file contains the ui interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Utils/Interfaces/IPlugin.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface IUI
00016     /// @brief Interface for ui
00017     /// @namespace cae
00018     ///
00019     class IUI : public utl::IPlugin
00020     {
00021
00022         public:
00023             ~IUI() override = default;
00024
00025     }; // interface IUI
00026
00027 } // namespace cae
```

43.67 modules/Interfaces/include/Interfaces/Window/AWindow.hpp File Reference

This file contains the Window abstract class.
#include "Interfaces/Window/IWindow.hpp"
Include dependency graph for AWindow.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class cae::AWindow

Namespaces

- namespace cae

43.67.1 Detailed Description

This file contains the Window abstract class.
Definition in file AWindow.hpp.

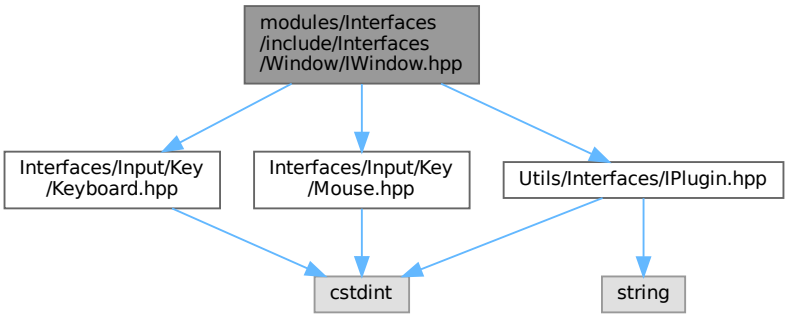
43.68 AWindow.hpp

Go to the documentation of this file.

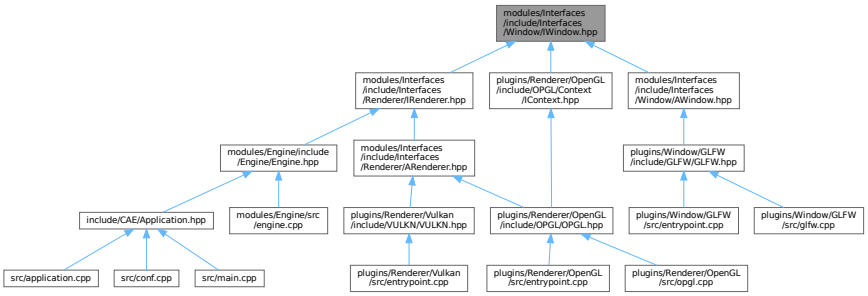
```
00001 ///
00002 /// @file AWindow.hpp
00003 /// @brief This file contains the Window abstract class
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Window/IWindow.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @interface IWindow
00016     /// @brief Abstract class for window
00017     /// @namespace cae
00018     ///
00019     class AWindow : public IWindow
00020     {
00021
00022         public:
00023             ~AWindow() override = default;
00024
00025     }; // interface AWindow
00026
00027 } // namespace cae
```

43.69 modules/Interfaces/include/Interfaces/Window/IWindow.hpp File Reference

This file contains the Window interface.
#include "Interfaces/Input/Key/Keyboard.hpp"
#include "Interfaces/Input/Key/Mouse.hpp"
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IWindow.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::WindowSize

  Struct for window size.
- struct cae::NativeWindowHandle

  Struct for native window handle.
- struct cae::WindowEvent

  Struct for window events.
- interface cae::IWindow

  Abstract class for window.

Namespaces

- namespace cae

Enumerations

- enum class cae::WindowEventType : uint8_t { cae::KeyDown , cae::KeyUp , cae::MouseMove , cae::MouseButtonDown , cae::MouseButtonUp , cae::MouseScroll , cae::Resize , cae::Focus , cae::Close }

  Enum for window event types.

43.69.1 Detailed Description

This file contains the Window interface.
Definition in file IWindow.hpp.

43.70 IWindow.hpp

Go to the documentation of this file.

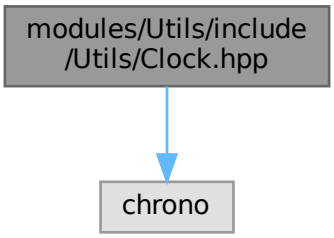
```
00001 ///
00002 /// @file IWindow.hpp
00003 /// @brief This file contains the Window interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Input/Key/Keyboard.hpp"
00010 #include "Interfaces/Input/Key/Mouse.hpp"
00011
00012 #include "Utils/Interfaces/IPlugin.hpp"
00013
00014 namespace cae
00015 {
00016
00017     ///
00018     /// @struct WindowSize
00019     /// @brief Struct for window size
00020     /// @namespace cae
00021     ///
00022     struct WindowSize
00023     {
00024         uint16_t width;
00025         uint16_t height;
00026     };
00027
00028     ///
00029     /// @struct NativeWindowHandle
00030     /// @brief Struct for native window handle
00031     /// @namespace cae
00032     ///
00033     struct NativeWindowHandle
00034     {
00035         void *window;
00036         void *display;
00037     };
00038
00039     ///
00040     /// @enum WindowEventType
00041     /// @brief Enum for window event types
00042     /// @namespace cae
00043     ///
00044     enum class WindowEventType : uint8_t
00045     {
00046         KeyDown,
00047         KeyUp,
00048         MouseMove,
00049         MouseButtonDown,
00050         MouseButtonUp,
00051         MouseScroll,
00052         Resize,
00053         Focus,
00054         Close
00055     };
00056
00057     ///
00058     /// @struct WindowEvent
00059     /// @brief Struct for window events
00060     /// @namespace cae
00061     ///
00062     struct WindowEvent
00063     {
```

```
00064           WindowEventType type;
00065
00066           union
00067           {
00068                   struct
00069                   {
00070                           KeyCode key;
00071                   } key;
00072                   struct
00073                   {
00074                           int x, y;
00075                   } mouseMove;
00076                   struct
00077                   {
00078                           MouseButton button;
00079                   } mouseButton;
00080                   struct
00081                   {
00082                           float x, y;
00083                   } scroll;
00084                   struct
00085                   {
00086                           uint16_t w, h;
00087                   } resize;
00088           };
00089   };
00090
00091   ///
00092   /// @interface IWindow
00093   /// @brief Interface for window
00094   /// @namespace cae
00095   ///
00096   class IWindow : public utl::IPlugin
00097   {
00098
00099       public:
00100           ~IWindow() override = default;
00101
00102           ///
00103           /// @param name Window name
00104           /// @param size Window size
00105           /// @return True if the window was created successfully
00106           /// @brief Create a window with the given name and size
00107           ///
00108           virtual bool create(const std::string &name, WindowSize size) = 0;
00109
00110           ///
00111           /// @brief Close the window
00112           ///
00113           virtual void close() = 0;
00114
00115           ///
00116           /// @return Native window handle
00117           /// @brief Get the native window handle
00118           ///
00119           virtual NativeWindowHandle getNativeHandle() const = 0;
00120
00121           ///
00122           /// @return Current window size
00123           /// @brief Get the current window size
00124           ///
00125           virtual WindowSize getWindowSize() const = 0;
00126
00127           ///
00128           /// @param path Path to the icon image
00129           /// @return True if the icon was set successfully
00130           /// @brief Set the window icon from the given image path
00131           ///
00132           virtual void setIcon(const std::string &path) const = 0;
00133
00134           ///
00135           /// @return True if the window should close
00136           /// @brief Check if the window should close
00137           ///
00138           virtual bool shouldClose() const = 0;
00139
00140           ///
00141           /// @brief Poll window events
00142           ///
00143           virtual void pollEvents() = 0;
00144
00145           ///
00146           /// @param outEvent Event to be filled
00147           /// @return True if an event was polled
00148           /// @brief Poll window events into outEvent
00149           ///
00150           virtual bool pollEvent(WindowEvent &outEvent) = 0;
```

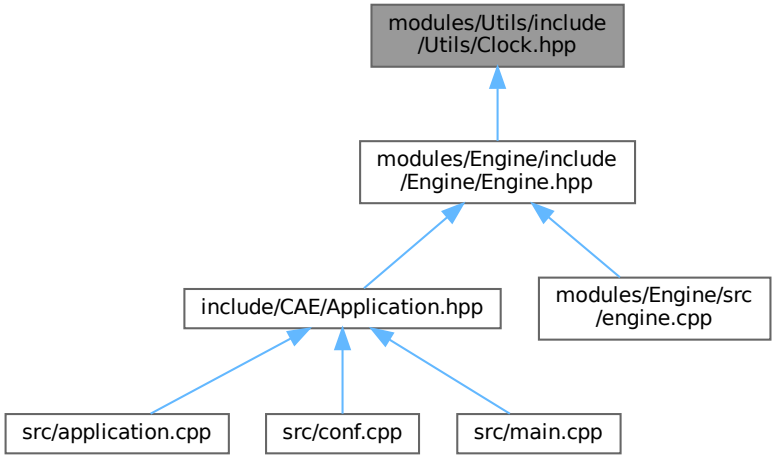
```
00151
00152             ///
00153             /// @return True if the window was resized
00154             /// @brief Check if the window was resized
00155             ///
00156             virtual bool wasResized() const = 0;
00157
00158             ///
00159             /// @brief Reset the resized flag
00160             ///
00161             virtual void resetResizedFlag() = 0;
00162
00163             // virtual bool isFullScreen() const = 0;
00164             // virtual void setFullScreen(bool fullScreen) const = 0;
00165
00166     }; // interface IWindow
00167
00168 } // namespace cae
```

43.71 modules/Utils/include/Utils/Clock.hpp File Reference

This file contains the Clock class.
#include <chrono>
Include dependency graph for Clock.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class utl::Clock

  Class for clock.

Namespaces

- namespace utl

43.71.1 Detailed Description

This file contains the Clock class.
Definition in file Clock.hpp.

43.72 Clock.hpp

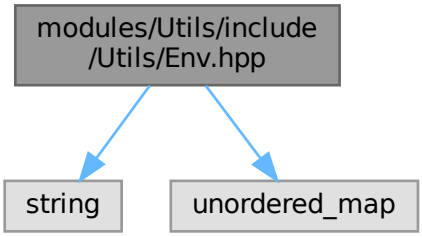
Go to the documentation of this file.

```
00001 ///
00002 /// @file Clock.hpp
00003 /// @brief This file contains the Clock class
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #include <chrono>
00010
00011 namespace utl
00012 {
00013
00014     ///
00015     /// @class Clock
00016     /// @brief Class for clock
00017     /// @namespace utl
00018     ///
00019     class Clock
00020     {
00021
00022         public:
00023             using TimePoint = std::chrono::time_point<std::chrono::high_resolution_clock>;
00024
00025             explicit Clock(const bool startNow = true) : m_start{startNow ? now() : TimePoint()}, m_pausedDuration{0} {}
00026             ~Clock() = default;
00027
00028             Clock(const Clock &) = delete;
00029             Clock &operator=(const Clock &) = delete;
00030             Clock(Clock &&) = delete;
00031             Clock &operator=(Clock &&) = delete;
00032
00033             friend std::ostream &operator«(std::ostream &os, const Clock &clock)
00034             {
00035                 os « "Elapsed time: " « clock.getDeltaSeconds() « " seconds";
00036                 return os;
00037             }
00038
00039             ///
00040             /// @return Current time point
00041             /// @brief Get the current time point
00042             ///
00043             static TimePoint now() { return std::chrono::high_resolution_clock::now(); }
00044
00045             ///
00046             /// @brief Restart the clock
00047             ///
00048             void restart()
00049             {
00050                 m_start = now();
00051                 m_pausedDuration = Duration(0);
00052                 m_isPaused = false;
00053             }
00054             ///
00055             /// @brief Pause the clock
00056             ///
00057             void pause()
00058             {
00059                 if (!m_isPaused)
00060                 {
00061                     m_pausedTime = now();
00062                     m_isPaused = true;
```

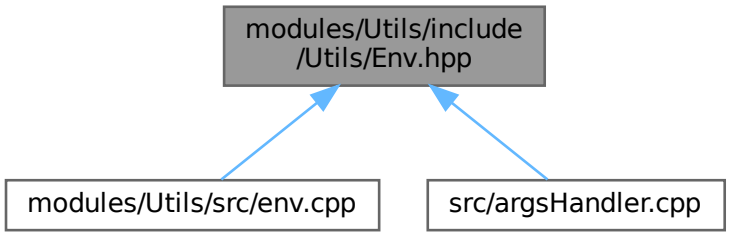
```
00063             }
00064         }
00065
00066         ///
00067         /// @brief Resume the clock
00068         ///
00069         void resume()
00070         {
00071             if (m_isPaused)
00072             {
00073                 m_pausedDuration += now() - m_pausedTime;
00074                 m_isPaused = false;
00075             }
00076         }
00077
00078         ///
00079         /// @return Elapsed time in seconds
00080         /// @brief Get the elapsed time in seconds
00081         ///
00082         [[nodiscard]] float getDeltaSeconds() const
00083         {
00084             if (m_isPaused)
00085             {
00086                 return std::chrono::duration<float>(m_pausedTime - m_start - m_pausedDuration).count();
00087             }
00088             return std::chrono::duration<float>(now() - m_start - m_pausedDuration).count();
00089         }
00090
00091         ///
00092         /// @tparam Duration Type of duration to return (default: seconds)
00093         /// @return Elapsed time in specified duration
00094         /// @brief Get the elapsed time in specified duration
00095         ///
00096         template <typename Duration = std::chrono::seconds> [[nodiscard]] auto getElapsed() const
00097         {
00098             return std::chrono::duration_cast<Duration>(now() - m_start - m_pausedDuration);
00099         }
00100
00101     private:
00102         using Duration = std::chrono::high_resolution_clock::duration;
00103
00104         TimePoint m_start;
00105         TimePoint m_pausedTime;
00106         Duration m_pausedDuration;
00107         bool m_isPaused{false};
00108
00109     }; // class Clock
00110
00111 } // namespace utl
```

43.73 modules/Utils/include/Utils/Env.hpp File Reference

This file contains env utility functions.
#include <string>
#include <unordered_map>
Include dependency graph for Env.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace utl

Functions

- std::unordered_map< std::string, std::string > utl::getEnvMap (const char *const *env)
  Get environment variables as a map.

43.73.1 Detailed Description

This file contains env utility functions.
Definition in file Env.hpp.

43.74 Env.hpp

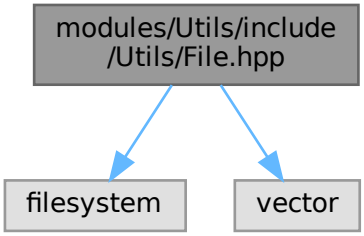
Go to the documentation of this file.

```
00001 ///
00002 /// @file Env.hpp
00003 /// @brief This file contains env utility functions
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #include <string>
00010 #include <unordered_map>
00011
00012 namespace utl
00013 {
00014     ///
00015     /// @param env Pointer to environment variables
00016     /// @return A map of environment variables
00017     /// @brief Get environment variables as a map
00018     ///
00019     [[nodiscard]] std::unordered_map<std::string, std::string> getEnvMap(const char *const *env);
00020 } // namespace utl
```

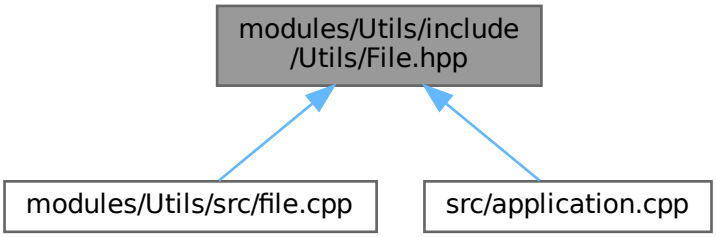
43.75 modules/Utils/include/Utils/File.hpp File Reference

This file contains file utility functions.
#include <filesystem>
#include <vector>

Include dependency graph for File.hpp:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace utl

Functions

- std::vector< char > utl::fileToVector (const std::filesystem::path &path)

  Read a file and return its contents as a vector of chars.
- std::string utl::fileToString (const std::filesystem::path &path)

  Read a file and return its contents as a string.

43.75.1 Detailed Description

This file contains file utility functions.
Definition in file File.hpp.

43.76 File.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file File.hpp
00003 /// @brief This file contains file utility functions
00004 /// @namespace utl
00005 ///
```

```
00006
00007 #pragma once
00008
00009 #include <filesystem>
00010 #include <vector>
00011
00012 namespace utl
00013 {
00014     ///
00015     /// @param path Path to the file
00016     /// @return A vector of chars containing the file contents
00017     /// @brief Read a file and return its contents as a vector of chars
00018     ///
00019     [[nodiscard]] std::vector<char> fileToVector(const std::filesystem::path &path);
00020
00021     ///
00022     /// @param path Path to the file
00023     /// @return A string containing the file contents
00024     /// @brief Read a file and return its contents as a string
00025     ///
00026     [[nodiscard]] std::string fileToString(const std::filesystem::path &path);
00027 } // namespace utl
```

43.77 modules/Utils/include/Utils/Generated/Version.hpp File Reference

Macros

- #define PROJECT_NAME "utils"
- #define PROJECT_VERSION "0.0.0"
- #define PROJECT_VERSION_MAJOR "0"
- #define PROJECT_VERSION_MINOR "0"
- #define PROJECT_VERSION_PATCH "0"
- #define GIT_COMMIT_HASH "0bd446e"
- #define GIT_TAG "0bd446e"
- #define BUILD_TYPE "Release"

43.77.1 Macro Definition Documentation

43.77.1.1 BUILD_TYPE

#define BUILD_TYPE "Release"
Definition at line 15 of file Version.hpp.

43.77.1.2 GIT_COMMIT_HASH

#define GIT_COMMIT_HASH "0bd446e"
Definition at line 13 of file Version.hpp.

43.77.1.3 GIT_TAG

#define GIT_TAG "0bd446e"
Definition at line 14 of file Version.hpp.

43.77.1.4 PROJECT_NAME

#define PROJECT_NAME "utils"
Definition at line 7 of file Version.hpp.

43.77.1.5 PROJECT_VERSION

#define PROJECT_VERSION "0.0.0"
Definition at line 8 of file Version.hpp.

43.77.1.6 PROJECT_VERSION_MAJOR

#define PROJECT_VERSION_MAJOR "0"
Definition at line 9 of file Version.hpp.

43.77.1.7 PROJECT_VERSION_MINOR

#define PROJECT_VERSION_MINOR "0"
Definition at line 10 of file Version.hpp.

43.77.1.8 PROJECT_VERSION_PATCH

#define PROJECT_VERSION_PATCH "0"
Definition at line 11 of file Version.hpp.

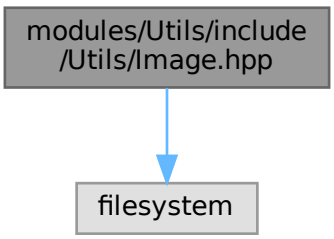
43.78 Version.hpp

Go to the documentation of this file.

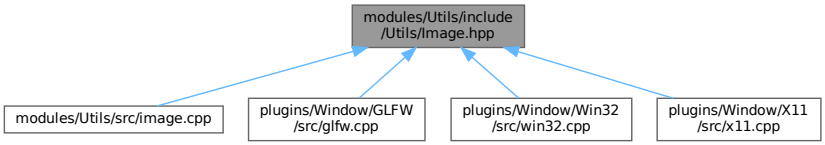
```
00001 #pragma once
00002
00003 //
      ================================================================================
00004 // DO NOT EDIT THIS FILE MANUALLY. IT IS GENERATED BY CMAKE DURING THE BUILD PROCESS.
00005 //
      ================================================================================
00006
00007 #define PROJECT_NAME "utils"
00008 #define PROJECT_VERSION "0.0.0"
00009 #define PROJECT_VERSION_MAJOR "0"
00010 #define PROJECT_VERSION_MINOR "0"
00011 #define PROJECT_VERSION_PATCH "0"
00012
00013 #define GIT_COMMIT_HASH "0bd446e"
00014 #define GIT_TAG "0bd446e"
00015 #define BUILD_TYPE "Release"
```

43.79 modules/Utils/include/Utils/Image.hpp File Reference

This file contains image struct.
#include <filesystem>
Include dependency graph for Image.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct utl::Image

  Struct for image.

Namespaces

- namespace utl

43.79.1 Detailed Description

This file contains image struct.
Definition in file Image.hpp.

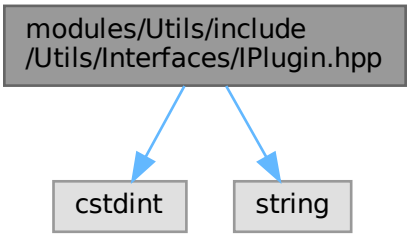
43.80 Image.hpp

Go to the documentation of this file.

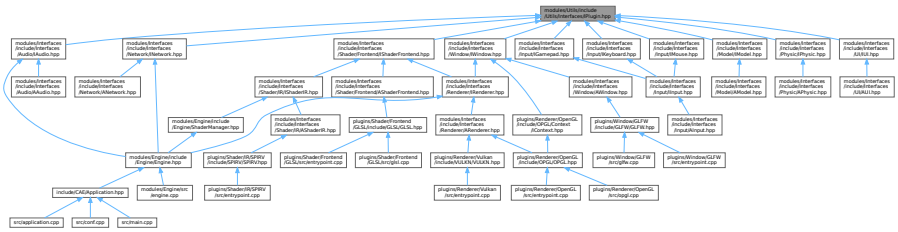
```
00001 ///
00002 /// @file Image.hpp
00003 /// @brief This file contains image struct
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #include <filesystem>
00010
00011 namespace utl
00012 {
00013
00014     ///
00015     /// @struct Image
00016     /// @brief Struct for image
00017     /// @namespace utl
00018     ///
00019     struct Image
00020     {
00021
00022         using pixel = unsigned char *;
00023
00024         ///
00025         /// @param path Path to the image file
00026         /// @param flip Whether to flip the image vertically
00027         /// @brief Load an image from a file
00028         ///
00029         explicit Image(const std::filesystem::path &path, bool flip = false);
00030         ~Image();
00031
00032         pixel pixels = nullptr;
00033         int width = 0;
00034         int height = 0;
00035         int channels = 0;
00036
00037     }; // struct Image
00038
00039 } // namespace utl
```

43.81 modules/Utils/include/Utils/Interfaces/IPlugin.hpp File Reference

This file contains the plugin interface.
#include <cstdint>
#include <string>
Include dependency graph for IPlugin.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface utl::IPlugin

  Interface for plugins.

Namespaces

- namespace utl

Macros

- #define PLUGIN_EXPORT

Enumerations

- enum class utl::PluginType : uint8_t { utl::AUDIO = 0 , utl::NETWORK = 1 , utl::RENDERER = 2 , utl::SHADER_IR = 3 , utl::SHADER_FRONTEND = 4 , utl::WINDOW = 5 , utl::UNDEFINED = 255 }
- enum class utl::PluginPlatform : uint8_t { utl::LINUX = 0 , utl::MACOSX = 1 , utl::WINDOWS = 2 , utl::ALL = 255 }

43.81.1 Detailed Description

This file contains the plugin interface.
Definition in file IPlugin.hpp.

43.81.2 Macro Definition Documentation

43.81.2.1 PLUGIN_EXPORT

#define PLUGIN_EXPORT
Definition at line 15 of file IPlugin.hpp.

43.82 IPlugin.hpp

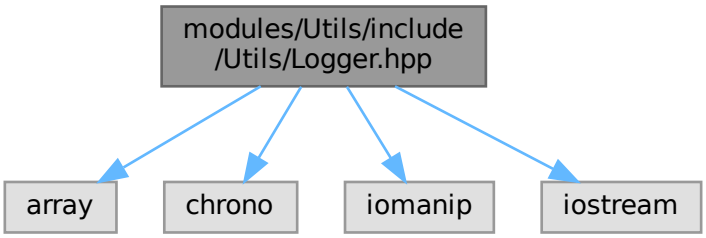
Go to the documentation of this file.

```
00001 ///
00002 /// @file IPlugin.hpp
00003 /// @brief This file contains the plugin interface
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #include <cstdint>
00010 #include <string>
00011
00012 #if defined(_WIN32) || defined(_WIN64)
00013 #define PLUGIN_EXPORT __declspec(dllexport)
00014 #else
00015 #define PLUGIN_EXPORT
00016 #endif
00017
00018 namespace utl
00019 {
00020
00021     enum class PluginType : uint8_t
00022     {
00023         AUDIO = 0,
00024         NETWORK = 1,
00025         RENDERER = 2,
00026         SHADER_IR = 3,
00027         SHADER_FRONTEND = 4,
00028         WINDOW = 5,
00029         UNDEFINED = 255,
00030     };
00031
00032     enum class PluginPlatform : uint8_t
00033     {
00034         LINUX = 0,
00035         MACOSX = 1,
00036         WINDOWS = 2,
00037         ALL = 255
00038     };
00039
00040     ///
00041     /// @interface IPlugin
00042     /// @brief Interface for plugins
00043     /// @namespace utl
00044     ///
00045     class IPlugin
00046     {
00047
00048         public:
00049             virtual ~IPlugin() = default;
00050
00051             ///
00052             /// @return The name of the plugin
00053             /// @brief Get the name of the plugin
00054             ///
00055             [[nodiscard]] virtual std::string getName() const = 0;
00056
00057             ///
00058             /// @return The type of the plugin
00059             /// @brief Get the type of the plugin
00060             ///
00061             [[nodiscard]] virtual PluginType getType() const = 0;
00062
00063             ///
00064             /// @return The handled platform of the plugin
00065             /// @brief Get the handled platform of the plugin
00066             ///
```

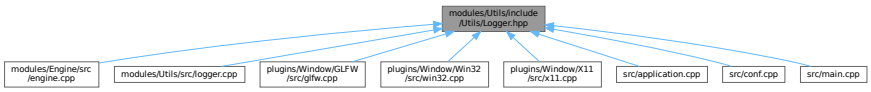
```
00067             [[nodiscard]] virtual PluginPlatform getPlatform() const = 0;
00068
00069     }; // interface IPlugin
00070 } // namespace utl
```

43.83 modules/Utils/include/Utils/Logger.hpp File Reference

This file contains the Logger class.
#include <array>
#include <chrono>
#include <iomanip>
#include <iostream>
Include dependency graph for Logger.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class utl::Logger
  Class for logging.

Namespaces

- namespace utl

Enumerations

- enum class utl::LogLevel : uint8_t { utl::INFO , utl::WARNING }

43.83.1 Detailed Description

This file contains the Logger class.
Definition in file Logger.hpp.

43.84 Logger.hpp

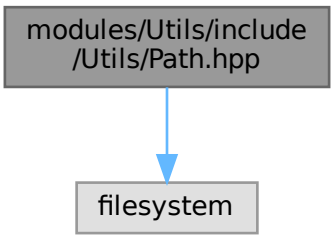
Go to the documentation of this file.

```
00001 ///
00002 /// @file Logger.hpp
00003 /// @brief This file contains the Logger class
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #include <array>
00010 #include <chrono>
00011 #include <iomanip>
00012 #include <iostream>
00013
00014 namespace utl
00015 {
00016
00017     enum class LogLevel : uint8_t
00018     {
00019         INFO,
00020         WARNING
00021     };
00022
00023     ///
00024     /// @class Logger
00025     /// @brief Class for logging
00026     /// @namespace utl
00027     ///
00028     class Logger
00029     {
00030
00031         private:
00032             template <typename E> static constexpr auto to_underlying(E e) noexcept
00033             {
00034                 return static_cast<std::underlying_type_t<E>>(e);
00035             }
00036
00037         public:
00038             Logger(const Logger &) = delete;
00039             Logger &operator=(const Logger &) = delete;
00040             Logger(Logger &&) = delete;
00041             Logger &operator=(Logger &&) = delete;
00042
00043             ///
00044             /// @brief Initialize the logger
00045             ///
00046             static void init();
00047
00048             ///
00049             /// @tparam Func Function to be measured
00050             /// @param message Message to be logged
00051             /// @param func Function to be measured
00052             /// @brief Log the execution time of a function
00053             ///
00054             template <typename Func> static void logExecutionTime(const std::string &message, Func &&func)
00055             {
00056                 const auto start = std::chrono::high_resolution_clock::now();
00057                 func();
00058                 const auto end = std::chrono::high_resolution_clock::now();
00059                 const auto duration = std::chrono::duration<float, std::milli>(end - start).count();
00060
00061                 std::cout « getColorForDuration(duration)
00062                         « formatLogMessage(LogLevel::INFO, message + " took " + std::to_string(duration) + " ms")
00063                         « LOG_LEVEL_COLOR[to_underlying(ColorIndex::COLOR_RESET)];
00064             }
00065
00066             ///
00067             /// @param message Message to be logged
00068             /// @param logLevel Log level of the message
00069             /// @brief Log a message with a specific log level
00070             ///
00071             static void log(const std::string &message, const LogLevel &logLevel)
00072             {
00073                 std::cout « (logLevel == LogLevel::INFO ?
    LOG_LEVEL_COLOR[to_underlying(ColorIndex::COLOR_INFO)]
00074                                                 : LOG_LEVEL_COLOR[to_underlying(ColorIndex::COLOR_WARNING)])
00075                         « formatLogMessage(logLevel, message)
00076                         « LOG_LEVEL_COLOR[to_underlying(ColorIndex::COLOR_RESET)];
00077             }
00078
00079         private:
00080             enum class ColorIndex : uint8_t
00081             {
00082                 COLOR_ERROR = 0,
00083                 COLOR_INFO = 1,
00084                 COLOR_WARNING = 2,
00085                 COLOR_RESET = 3
00086             };
```

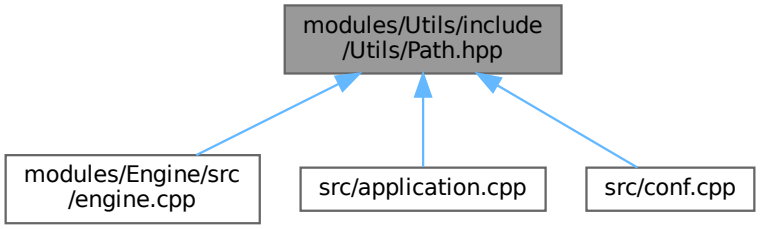
```
00087
00088           static constexpr std::array<const char *, 4> LOG_LEVEL_COLOR = {
00089               "\033[31m", // ERROR/slow execution
00090               "\033[32m", // INFO/fast execution
00091               "\033[33m", // WARNING/medium execution
00092               "\033[0m\n" // RESET + newline
00093           };
00094
00095           static constexpr std::array<const char *, 2> LOG_LEVEL_STRING = {"INFO", "WARNING"};
00096
00097           Logger() = default;
00098           ~Logger() = default;
00099
00100          [[nodiscard]] static const char *getColorForDuration(const float duration)
00101          {
00102              return duration < 20.0F ? LOG_LEVEL_COLOR[to_underlying(ColorIndex::COLOR_INFO)]
00103                                      : (duration < 90.0F ?
      LOG_LEVEL_COLOR[to_underlying(ColorIndex::COLOR_WARNING)]
00104                                                          : LOG_LEVEL_COLOR[to_underlying(ColorIndex::COLOR_ERROR)]);
00105          }
00106
00107          [[nodiscard]] static std::string formatLogMessage(LogLevel level, const std::string &message)
00108          {
00109              using namespace std::chrono;
00110
00111              const auto now = system_clock::now();
00112              const auto inTimeT = system_clock::to_time_t(now);
00113              const auto ms = duration_cast<milliseconds>(now.time_since_epoch()) % 1000;
00114
00115              std::ostringstream ss;
00116              ss « "[" « std::put_time(std::localtime(&inTimeT), "%Y-%m-%d %H:%M:%S");
00117              ss « ":" « std::setfill('0') « std::setw(3) « ms.count() « "] ";
00118              ss « "[" « LOG_LEVEL_STRING[static_cast<uint8_t>(level)] « "] " « message;
00119
00120              return ss.str();
00121          }
00122
00123      }; // class Logger
00124
00125 } // namespace utl
```

43.85 modules/Utils/include/Utils/Path.hpp File Reference

This file contains Path resolution utilities.
#include <filesystem>
Include dependency graph for Path.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class utl::Path

  Class for path resolution utilities.

Namespaces

- namespace utl

43.85.1 Detailed Description

This file contains Path resolution utilities.
Definition in file Path.hpp.

43.86 Path.hpp

Go to the documentation of this file.

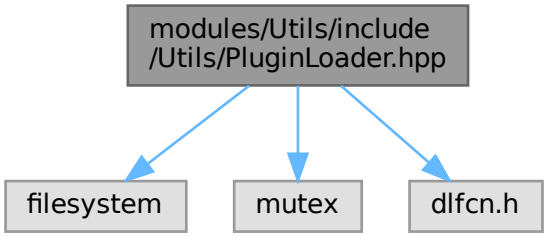
```
00001 ///
00002 /// @file Path.hpp
00003 /// @brief This file contains Path resolution utilities
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #ifdef _WIN32
00010 #include <windows.h>
00011 #elifdef __APPLE__
00012 #include <mach-o/dyld.h>
00013 #elif __linux__
00014 #include <unistd.h>
00015 #endif
00016
00017 #include <filesystem>
00018
00019 namespace utl
00020 {
00021
00022     namespace fs = std::filesystem;
00023
00024     ///
00025     /// @class Path
00026     /// @brief Class for path resolution utilities
00027     /// @namespace utl
00028     ///
00029     class Path
00030     {
00031
00032         public:
00033             Path() = default;
00034             ~Path() = default;
00035
00036             Path(const Path &) = delete;
00037             Path &operator=(const Path &) = delete;
```

```
00038           Path(Path &&) = delete;
00039           Path &operator=(Path &&) = delete;
00040
00041           /// @param path Path to be normalized
00042           /// @return Normalized path
00043           /// @brief Normalize a path (resolve symlinks, relative paths, etc.)
00044           static fs::path normalize(const fs::path &path)
00045           {
00046               try
00047               {
00048                   return fs::weakly_canonical(path);
00049               }
00050               catch (...)
00051               {
00052                   return fs::absolute(path);
00053               }
00054           }
00055
00056           ///
00057           /// @param path Path to be checked
00058           /// @return True if the file exists
00059           /// @brief Check if a file exists
00060           ///
00061           static bool existsFile(const fs::path &path) { return fs::exists(path) && fs::is_regular_file(path); }
00062
00063           ///
00064           /// @param path Path to be checked
00065           /// @return True if the directory exists
00066           /// @brief Check if a directory exists
00067           ///
00068           static bool existsDir(const fs::path &path) { return fs::exists(path) && fs::is_directory(path); }
00069
00070           ///
00071           /// @param path Path to get the parent directory of
00072           /// @return Parent directory of the path
00073           /// @brief Get the parent directory of a path
00074           ///
00075           static fs::path parentDir(const fs::path &path) { return normalize(path).parent_path(); }
00076
00077           ///
00078           /// @tparam Paths Variadic template for paths
00079           /// @param paths Paths to be joined
00080           /// @return Joined path
00081           /// @brief Join multiple paths
00082           ///
00083           template <typename... Paths> static fs::path join(const Paths &...paths)
00084           {
00085               return normalize((fs::path(paths) / ...));
00086           }
00087
00088           ///
00089           /// @return Directory of the executable
00090           /// @brief Get the directory of the executable
00091           ///
00092           static fs::path executableDir()
00093           {
00094 #ifdef _WIN32
00095               char buffer[MAX_PATH];
00096               GetModuleFileNameA(NULL, buffer, MAX_PATH);
00097               return fs::path(buffer).parent_path();
00098 #elifdef ___APPLE___
00099               char buffer[1024];
00100               uint32_t size = sizeof(buffer);
00101               if (_NSGetExecutablePath(buffer, &size) == 0)
00102               {
00103                   return fs::path(buffer).parent_path();
00104               }
00105               return fs::current_path();
00106 #else
00107               char buffer[1024];
00108               if (const ssize_t count = readlink("/proc/self/exe", buffer, sizeof(buffer)); count != -1)
00109               {
00110                   return fs::path(std::string(buffer, count)).parent_path();
00111               }
00112               return fs::current_path();
00113 #endif
00114           }
00115
00116           ///
00117           /// @param relativePath Relative path to be resolved
00118           /// @return Resolved path relative to the executable directory
00119           /// @brief Resolve a relative path to the executable directory
00120           ///
00121           static fs::path resolveRelativeToExe(const fs::path &relativePath)
00122           {
00123               return normalize(executableDir() / relativePath);
00124           }
```

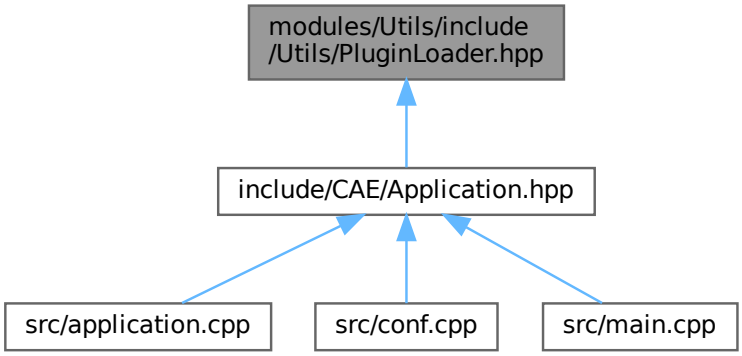
```
00125
00126             ///
00127             /// @param relativePath Relative path to be resolved
00128             /// @return Resolved path relative to the user cwd
00129             /// @brief
00130             ///
00131             static fs::path resolveRelativeToCwd(const fs::path &relativePath)
00132             {
00133                 return normalize(fs::current_path() / relativePath);
00134             }
00135
00136     }; // class Path
00137
00138 } // namespace utl
```

43.87 modules/Utils/include/Utils/PluginLoader.hpp File Reference

Modern, cross-platform plugin loader.
#include <filesystem>
#include <mutex>
#include <dlfcn.h>
Include dependency graph for PluginLoader.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct utl::SharedLib

RAII wrapper for shared libraries.

- class utl::PluginLoader

  Modern, type-safe plugin loader.

Namespaces

- namespace utl

Macros

- #define PLUGINS_PREFIX "lib"

Typedefs

- using utl::LibHandle
- using utl::EntryPointFn = IPlugin *(*)()

43.87.1 Detailed Description

Modern, cross-platform plugin loader.
Definition in file PluginLoader.hpp.

43.87.2 Macro Definition Documentation

43.87.2.1 PLUGINS_PREFIX

#define PLUGINS_PREFIX "lib"
Definition at line 23 of file PluginLoader.hpp.
Referenced by utl::PluginLoader::loadPlugin().

43.88 PluginLoader.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file PluginLoader.hpp
00003 /// @brief Modern, cross-platform plugin loader
00004 /// @namespace utl
00005 ///
00006
00007 #pragma once
00008
00009 #include <filesystem>
00010 #include <mutex>
00011
00012 #ifdef _WIN32
00013 #include <windows.h>
00014 #define PLUGINS_EXTENSION ".dll"
00015 #define PLUGINS_PREFIX ""
00016 #else
00017 #include <dlfcn.h>
00018 #ifdef __APPLE__
00019 #define PLUGINS_EXTENSION ".dylib"
00020 #elif __linux__
00021 #define PLUGINS_EXTENSION ".so"
00022 #endif
00023 #define PLUGINS_PREFIX "lib"
00024 #endif
00025
00026 namespace utl
00027 {
00028
00029     using LibHandle =
00030 #ifdef _WIN32
00031         HMODULE;
00032 #else
00033         void *;
00034 #endif
00035
00036     ///
00037     /// @struct SharedLib
00038     /// @brief RAII wrapper for shared libraries
00039     /// @namespace utl
```

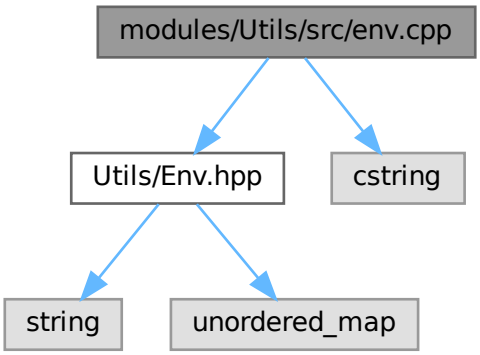
```
00040     ///
00041     struct SharedLib
00042     {
00043         LibHandle handle = nullptr;
00044
00045         explicit SharedLib(const LibHandle h = nullptr) : handle(h) {}
00046         ~SharedLib() { close(); }
00047
00048         SharedLib(const SharedLib &) = delete;
00049         SharedLib &operator=(const SharedLib &) = delete;
00050         SharedLib(SharedLib &&other) noexcept : handle(other.handle) { other.handle = nullptr; }
00051         SharedLib &operator=(SharedLib &&other) noexcept
00052         {
00053             if (this != &other)
00054             {
00055                 close();
00056                 handle = other.handle;
00057                 other.handle = nullptr;
00058             }
00059             return *this;
00060         }
00061
00062         void close()
00063         {
00064             if (handle == nullptr)
00065             {
00066                 return;
00067             }
00068 #ifdef _WIN32
00069             FreeLibrary(handle);
00070 #else
00071             dlclose(handle);
00072 #endif
00073             handle = nullptr;
00074         }
00075
00076         explicit operator bool() const { return handle != nullptr; }
00077     };
00078
00079     using EntryPointFn = IPlugin *(*)();
00080
00081     ///
00082     /// @class PluginLoader
00083     /// @brief Modern, type-safe plugin loader
00084     /// @namespace utl
00085     ///
00086     class PluginLoader
00087     {
00088         public:
00089             PluginLoader() = default;
00090             ~PluginLoader() = default;
00091
00092             PluginLoader(const PluginLoader &) = delete;
00093             PluginLoader &operator=(const PluginLoader &) = delete;
00094             PluginLoader(PluginLoader &&) = delete;
00095             PluginLoader &operator=(PluginLoader &&) = delete;
00096
00097             ///
00098             /// @tparam T Expected plugin interface (must derive from IPlugin)
00099             /// @param path Path to the dynamic library
00100             /// @param pluginPrefix Expected prefix for plugin filenames
00101             /// @return shared_ptr<T> instance
00102             /// @brief Load a plugin of type T
00103             ///
00104             template <std::derived_from<IPlugin> T>
00105             std::shared_ptr<T> loadPlugin(const std::string &path, const std::string_view &pluginPrefix = "")
00106             {
00107                 std::scoped_lock lock(m_mutex);
00108
00109                 try
00110                 {
00111                     validatePluginPath(path, PLUGINS_PREFIX + std::string(pluginPrefix));
00112
00113                     if (m_plugins.contains(path))
00114                     {
00115                         return nullptr;
00116                     }
00117
00118                     SharedLib lib = loadLibrary(path);
00119                     const EntryPointFn entry = getEntryPoint(lib, path);
00120
00121                     std::unique_ptr<IPlugin> plugin(entry());
00122                     if (!plugin)
00123                     {
00124                         throw std::runtime_error("EntryPoint returned null");
00125                     }
00126
```

```
00127                 T *typed = dynamic_cast<T *>(plugin.get());
00128                 if (!typed)
00129                 {
00130                     throw std::runtime_error("Plugin type mismatch");
00131                 }
00132
00133                 auto [it, inserted] = m_plugins.emplace(path, std::move(plugin));
00134                 if (!inserted)
00135                 {
00136                     throw std::runtime_error("Failed to store plugin");
00137                 }
00138
00139                 m_handles[path] = std::move(lib);
00140
00141                 std::shared_ptr<IPlugin> baseShared(it->second.get(), [](IPlugin *) {});
00142                 return std::shared_ptr<T>(baseShared, typed);
00143             }
00144             catch (const std::exception &e)
00145             {
00146                 return nullptr;
00147             }
00148         }
00149
00150     private:
00151         std::mutex m_mutex;
00152         std::unordered_map<std::string, SharedLib> m_handles;
00153         std::unordered_map<std::string, std::unique_ptr<IPlugin>> m_plugins;
00154
00155         ///
00156         /// @param path Path to the dynamic library
00157         /// @return Loaded SharedLib
00158         /// @brief Load a shared library
00159         ///
00160         static SharedLib loadLibrary(const std::string &path)
00161         {
00162             const LibHandle handle =
00163 #ifdef _WIN32
00164                 LoadLibraryA(path.c_str());
00165 #else
00166                 dlopen(path.c_str(), RTLD_LAZY);
00167 #endif
00168             if (handle == nullptr)
00169             {
00170                 const std::string dlErrorMsg =
00171 #ifdef _WIN32
00172                     "LoadLibraryA failed";
00173 #else
00174                     dlerror();
00175 #endif
00176                 throw std::runtime_error("Cannot load library: " + dlErrorMsg + "\nWith path: " + path);
00177             }
00178             return SharedLib(handle);
00179         }
00180
00181         ///
00182         /// @param lib Loaded SharedLib
00183         /// @param path Path to the dynamic library
00184         /// @return EntryPoint function pointer
00185         ///
00186         static EntryPointFn getEntryPoint(SharedLib &lib, const std::string &path)
00187         {
00188             const auto entry =
00189 #ifdef _WIN32
00190                 reinterpret_cast<EntryPointFn>(GetProcAddress(lib.handle, "entryPoint"));
00191 #else
00192                 reinterpret_cast<EntryPointFn>(dlsym(lib.handle, "entryPoint"));
00193 #endif
00194             if (entry == nullptr)
00195             {
00196                 throw std::runtime_error("EntryPoint not found in plugin: " + path);
00197             }
00198             return entry;
00199         }
00200
00201         ///
00202         /// @param path Path to the dynamic library
00203         /// @param pluginPrefix Expected prefix for plugin filenames
00204         /// @brief Validate the plugin path
00205         ///
00206         static void validatePluginPath(const std::filesystem::path &path, const std::string_view &pluginPrefix)
00207         {
00208             const std::string filename = path.filename().string();
00209
00210             if (path.extension() != PLUGINS_EXTENSION)
00211             {
00212                 throw std::runtime_error("Invalid plugin extension: " + filename +
00213                                          " (expected " PLUGINS_EXTENSION ")");
```

```
00214             }
00215
00216             if (!filename.starts_with(pluginPrefix))
00217             {
00218                 throw std::runtime_error("Invalid plugin name: " + filename + " (plugins must start with '" +
00219                                          std::string(pluginPrefix) + "')");
00220             }
00221         }
00222
00223     }; // class PluginLoader
00224
00225 } // namespace utl
```

43.89 modules/Utils/src/env.cpp File Reference

#include "Utils/Env.hpp"
#include <cstring>
Include dependency graph for env.cpp:



43.90 env.cpp

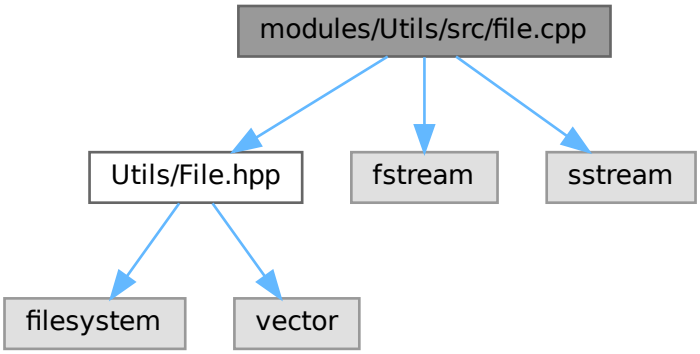
Go to the documentation of this file.

```
00001 #include "Utils/Env.hpp"
00002
00003 #ifdef _WIN32
00004 #include <windows.h>
00005 #endif
00006
00007 #include <cstring>
00008
00009 std::unordered_map<std::string, std::string> utl::getEnvMap(const char *const *env)
00010 {
00011     std::unordered_map<std::string, std::string> cpyEnv;
00012
00013 #ifdef _WIN32
00014     const LPCH envStrings = GetEnvironmentStringsA();
00015     if (envStrings == nullptr)
00016     {
00017         return cpyEnv;
00018     }
00019
00020     for (LPCH var = envStrings; *var != 0; var += std::strlen(var) + 1)
00021     {
00022         std::string entry(var);
00023         if (const auto pos = entry.find('='); pos != std::string::npos)
00024         {
00025             cpyEnv.emplace(entry.substr(0, pos), entry.substr(pos + 1));
00026         }
00027     }
00028
00029     FreeEnvironmentStringsA(envStrings);
```

```
00030 #else
00031     for (const char *const *current = env; (current != nullptr) && (*current != nullptr); ++current)
00032     {
00033         std::string entry(*current);
00034         if (const auto pos = entry.find('='); pos != std::string::npos)
00035         {
00036             cpyEnv.emplace(entry.substr(0, pos), entry.substr(pos + 1));
00037         }
00038     }
00039 #endif
00040
00041     return cpyEnv;
00042 }
```

43.91 modules/Utils/src/file.cpp File Reference

#include "Utils/File.hpp"
#include <fstream>
#include <sstream>
Include dependency graph for file.cpp:



43.92 file.cpp

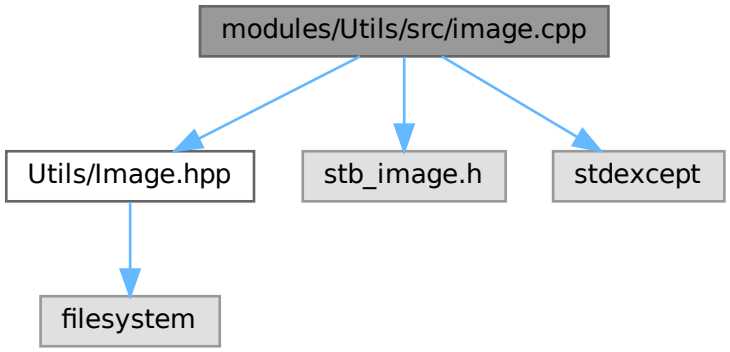
Go to the documentation of this file.

```
00001 #include "Utils/File.hpp"
00002
00003 #include <fstream>
00004 #include <sstream>
00005
00006 std::vector<char> utl::fileToVector(const std::filesystem::path &path)
00007 {
00008     std::ifstream file(path, std::ios::in);
00009     if (!file.is_open())
00010     {
00011         throw std::runtime_error("failed to open file " + path.string());
00012     }
00013     const long int fileSize = file.tellg();
00014     if (fileSize <= 0)
00015     {
00016         throw std::runtime_error("file " + path.string() + " is empty");
00017     }
00018     std::vector<char> buffer(static_cast<long unsigned int>(fileSize));
00019     file.seekg(0, std::ios::beg);
00020     if (!file.read(buffer.data(), fileSize))
00021     {
00022         throw std::runtime_error("failed to read file " + path.string());
00023     }
00024     return buffer;
00025 }
00026
00027 std::string utl::fileToString(const std::filesystem::path &path)
```

```
00028 {
00029     const std::ifstream file(path, std::ios::in);
00030     if (!file.is_open())
00031     {
00032         throw std::runtime_error("Failed to open file: " + path.string());
00033     }
00034     std::stringstream buffer;
00035     buffer « file.rdbuf();
00036     return buffer.str();
00037 }
```

43.93 modules/Utils/src/image.cpp File Reference

#include "Utils/Image.hpp"
#include <stb_image.h>
#include <stdexcept>
Include dependency graph for image.cpp:



Macros

- #define STB_IMAGE_IMPLEMENTATION

43.93.1 Macro Definition Documentation

43.93.1.1 STB_IMAGE_IMPLEMENTATION

#define STB_IMAGE_IMPLEMENTATION
Definition at line 3 of file image.cpp.

43.94 image.cpp

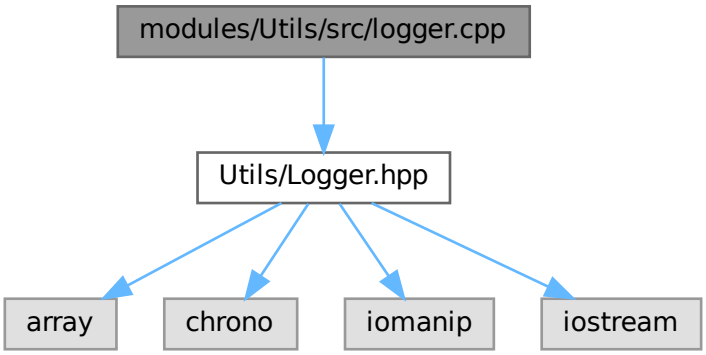
Go to the documentation of this file.

```
00001 #include "Utils/Image.hpp"
00002
00003 #define STB_IMAGE_IMPLEMENTATION
00004 #include <stb_image.h>
00005
00006 #include <stdexcept>
00007
00008 utl::Image::Image(const std::filesystem::path &path, const bool flip)
00009 {
00010     if (flip)
00011     {
00012         stbi_set_flip_vertically_on_load(1);
00013     }
00014     pixels = stbi_load(path.string().c_str(), &width, &height, &channels, STBI_rgb_alpha);
00015     if (pixels == nullptr)
```

```
00016     {
00017         throw std::runtime_error("Failed to load image: " + path.string());
00018     }
00019 }
00020
00021 utl::Image::~Image() { stbi_image_free(pixels); }
```

43.95 modules/Utils/src/logger.cpp File Reference

#include "Utils/Logger.hpp"
Include dependency graph for logger.cpp:



43.96 logger.cpp

Go to the documentation of this file.

```
00001 #include "Utils/Logger.hpp"
00002
00003 #ifdef _WIN32
00004 #include <windows.h>
00005 #endif
00006
00007 void utl::Logger::init()
00008 {
00009 #ifdef _WIN32
00010     const HANDLE hOut = GetStdHandle(STD_OUTPUT_HANDLE);
00011     DWORD dwMode = 0;
00012     if (hOut != INVALID_HANDLE_VALUE && GetConsoleMode(hOut, &dwMode))
00013     {
00014         SetConsoleMode(hOut, dwMode | ENABLE_VIRTUAL_TERMINAL_PROCESSING);
00015     }
00016 #endif
00017 }
```

43.97 plugins/Audio/ALSA/include/ALSA/ALSA.hpp File Reference

43.98 ALSA.hpp

Go to the documentation of this file.

```
00001
```

43.99 plugins/Audio/Core/include/Core/Core.hpp File Reference

43.100 Core.hpp

Go to the documentation of this file.
00001

43.101 plugins/Audio/OpenAL/include/OpenAL/OpenAL.hpp File Reference

43.102 OpenAL.hpp

Go to the documentation of this file.
00001

43.103 plugins/Audio/XAudio2/include/XAudio2/XAudio2.hpp File Reference

43.104 XAudio2.hpp

Go to the documentation of this file.
00001

43.105 plugins/Model/Assimp/include/Assimp/Assimp.hpp File Reference

43.106 Assimp.hpp

Go to the documentation of this file.
00001

43.107 plugins/Audio/ALSA/src/entrypoint.cpp File Reference

43.108 entrypoint.cpp

Go to the documentation of this file.
00001

43.109 plugins/Audio/Core/src/entrypoint.cpp File Reference

43.110 entrypoint.cpp

Go to the documentation of this file.
00001

43.111 plugins/Audio/OpenAL/src/entrypoint.cpp File Reference

43.112 entrypoint.cpp

Go to the documentation of this file.
00001

43.113 plugins/Audio/XAudio2/src/entrypoint.cpp File Reference

43.114 entrypoint.cpp

Go to the documentation of this file.
00001

43.115 plugins/Input/Cocoa/src/entrypoint.cpp File Reference

43.116 entrypoint.cpp

Go to the documentation of this file.
00001

43.117 plugins/Input/Win32/src/entrypoint.cpp File Reference

43.118 entrypoint.cpp

Go to the documentation of this file.
00001

43.119 plugins/Input/X11/src/entrypoint.cpp File Reference

43.120 entrypoint.cpp

Go to the documentation of this file.
00001

43.121 plugins/Model/Assimp/src/entrypoint.cpp File Reference

43.122 entrypoint.cpp

Go to the documentation of this file.
00001

43.123 plugins/Network/Asio/src/entrypoint.cpp File Reference

43.124 entrypoint.cpp

Go to the documentation of this file.
00001

43.125 plugins/Network/Posix/src/entrypoint.cpp File Reference

43.126 entrypoint.cpp

Go to the documentation of this file.
00001

43.127 plugins/Network/WinSock/src/entrypoint.cpp File Reference

43.128 entrypoint.cpp

Go to the documentation of this file.
00001

43.129 plugins/Renderer/DirectX12/src/entrypoint.cpp File Reference

43.130 entrypoint.cpp

Go to the documentation of this file.
00001

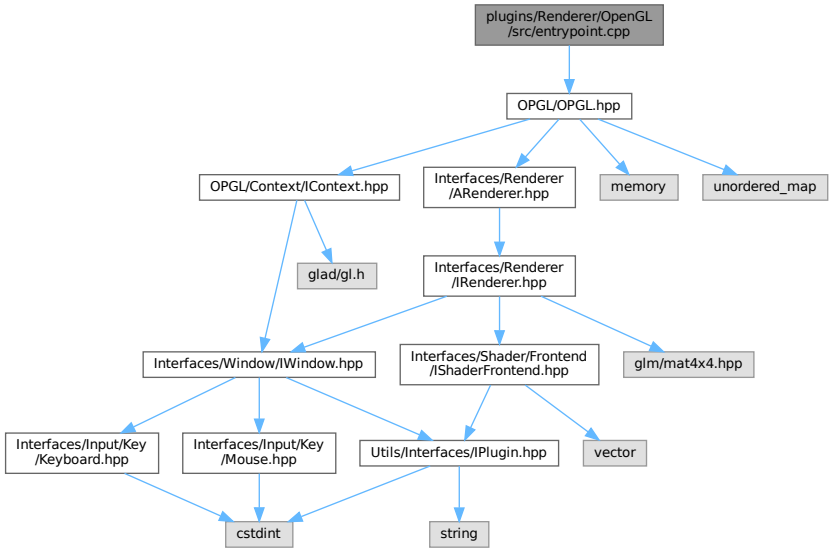
43.131 plugins/Renderer/Metal/src/entrypoint.cpp File Reference

43.132 entrypoint.cpp

Go to the documentation of this file.
00001

43.133 plugins/Renderer/OpenGL/src/entrypoint.cpp File Reference

#include ”OPGL/OPGL.hpp”
Include dependency graph for entrypoint.cpp:



Functions

- PLUGIN_EXPORT cae::IRenderer * entryPoint ()

43.133.1 Function Documentation

43.133.1.1 entryPoint()

PLUGIN_EXPORT cae::IRenderer * entryPoint ()
Definition at line 5 of file entrypoint.cpp.

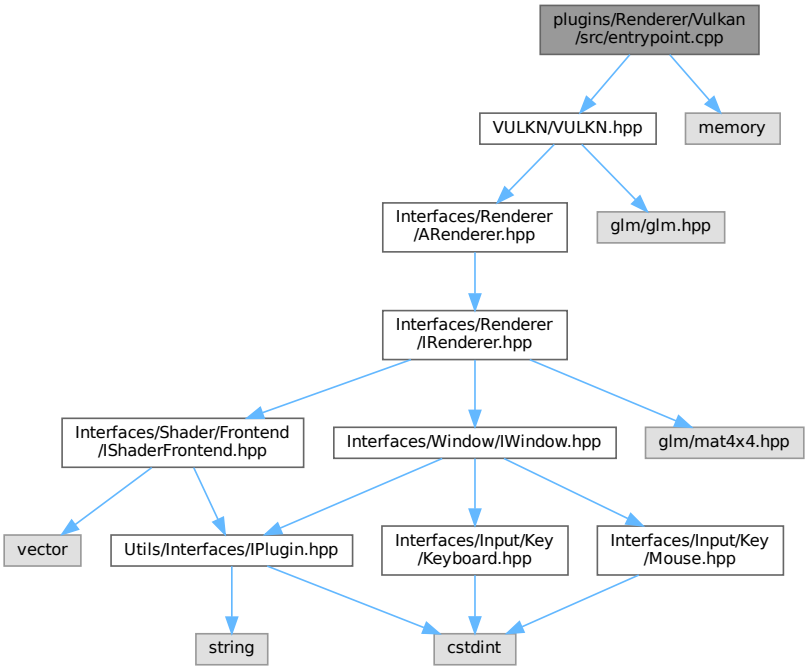
43.134 entrypoint.cpp

Go to the documentation of this file.
```
00001 #include ”OPGL/OPGL.hpp”
00002
00003 extern ”C”
```

```
00004 {
00005     PLUGIN_EXPORT cae::IRenderer *entryPoint() { return std::make_unique<cae::OPGL>().release(); }
00006 }
```

43.135 plugins/Renderer/Vulkan/src/entrypoint.cpp File Reference

#include ”VULKN/VULKN.hpp”
#include <memory>
Include dependency graph for entrypoint.cpp:



Functions

- PLUGIN_EXPORT cae::IRenderer * entryPoint ()

43.135.1 Function Documentation

43.135.1.1 entryPoint()

PLUGIN_EXPORT cae::IRenderer * entryPoint ()
Definition at line 7 of file entrypoint.cpp.

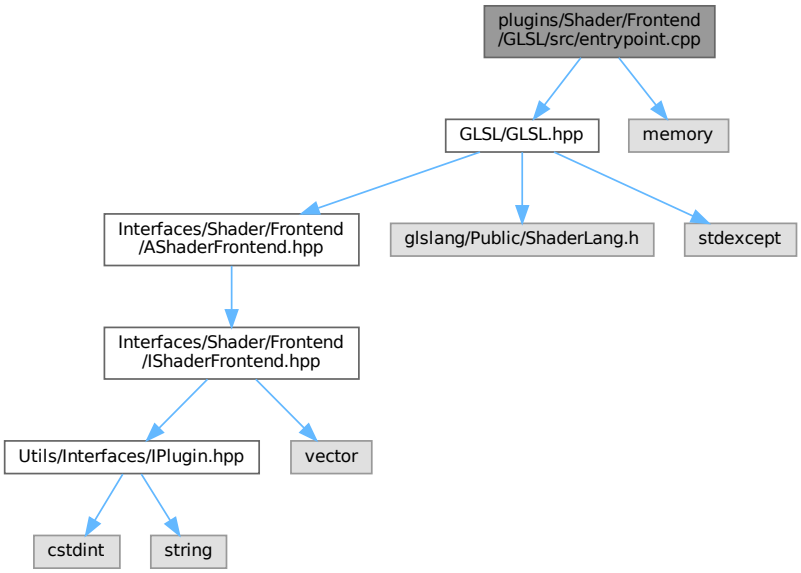
43.136 entrypoint.cpp

Go to the documentation of this file.

```
00001 #include ”VULKN/VULKN.hpp”
00002
00003 #include <memory>
00004
00005 extern ”C”
00006 {
00007     PLUGIN_EXPORT cae::IRenderer *entryPoint() { return std::make_unique<cae::VULKN>().release(); }
00008 }
```

43.137 plugins/Shader/Frontend/GLSL/src/entrypoint.cpp File Reference

#include ”GLSL/GLSL.hpp”
#include <memory>
Include dependency graph for entrypoint.cpp:



Functions

- PLUGIN_EXPORT cae::IShaderFrontend * entryPoint ()

43.137.1 Function Documentation

43.137.1.1 entryPoint()

PLUGIN_EXPORT cae::IShaderFrontend * entryPoint ()
Definition at line 7 of file entrypoint.cpp.

43.138 entrypoint.cpp

Go to the documentation of this file.

```
00001 #include ”GLSL/GLSL.hpp”
00002
00003 #include <memory>
00004
00005 extern ”C”
00006 {
00007     PLUGIN_EXPORT cae::IShaderFrontend *entryPoint() { return std::make_unique<cae::GLSL>().release(); }
00008 }
```

43.139 plugins/Shader/Frontend/HLSL/src/entrypoint.cpp File Reference

43.140 entrypoint.cpp

Go to the documentation of this file.

00001

43.141 plugins/Shader/Frontend/MSL/src/entrypoint.cpp File Reference

43.142 entrypoint.cpp

Go to the documentation of this file.
00001

43.143 plugins/Shader/Frontend/WGSL/src/entrypoint.cpp File Reference

43.144 entrypoint.cpp

Go to the documentation of this file.
00001

43.145 plugins/Shader/IR/DXC/src/entrypoint.cpp File Reference

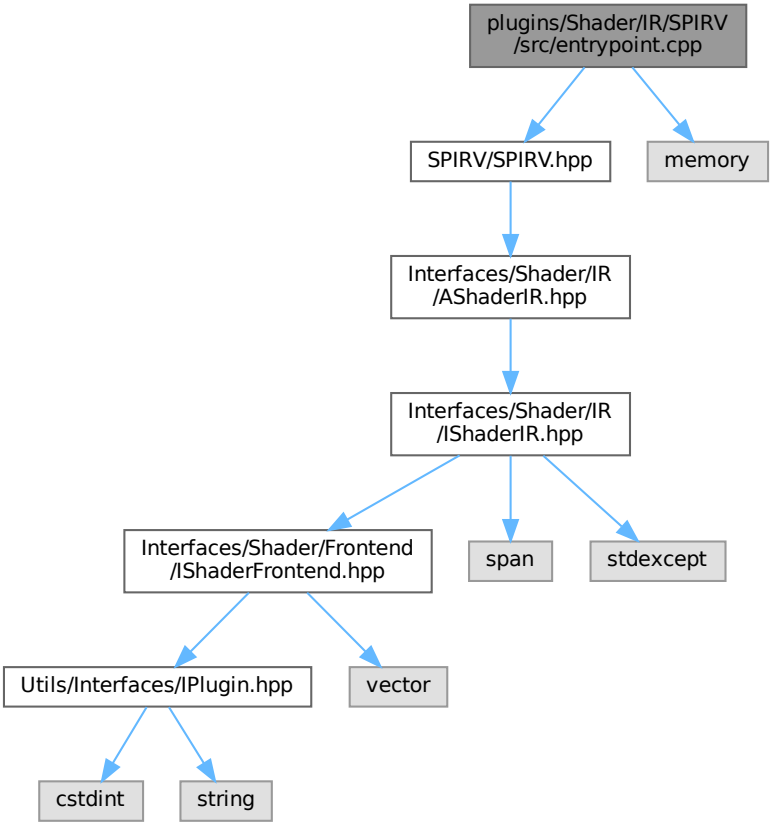
43.146 entrypoint.cpp

Go to the documentation of this file.
00001

43.147 plugins/Shader/IR/SPIRV/src/entrypoint.cpp File Reference

#include ”SPIRV/SPIRV.hpp”
#include <memory>

Include dependency graph for entrypoint.cpp:



Functions

- PLUGIN_EXPORT cae::IShaderIR * entryPoint ()

43.147.1 Function Documentation

43.147.1.1 entryPoint()

PLUGIN_EXPORT cae::IShaderIR * entryPoint ()
Definition at line 7 of file entrypoint.cpp.

43.148 entrypoint.cpp

Go to the documentation of this file.

```
00001 #include "SPIRV/SPIRV.hpp"
00002
00003 #include <memory>
00004
00005 extern "C"
00006 {
00007     PLUGIN_EXPORT cae::IShaderIR *entryPoint() { return std::make_unique<cae::SPIRV>().release(); }
00008 }
```

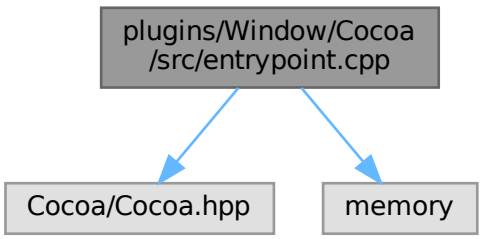
43.149 plugins/UI/Imgui/src/entrypoint.cpp File Reference

43.150 entrypoint.cpp

Go to the documentation of this file.
00001

43.151 plugins/Window/Cocoa/src/entrypoint.cpp File Reference

#include ”Cocoa/Cocoa.hpp”
#include <memory>
Include dependency graph for entrypoint.cpp:



Functions

- PLUGIN_EXPORT cae::IWindow * entryPoint ()

43.151.1 Function Documentation

43.151.1.1 entryPoint()

PLUGIN_EXPORT cae::IWindow * entryPoint ()
Definition at line 7 of file entrypoint.cpp.

43.152 entrypoint.cpp

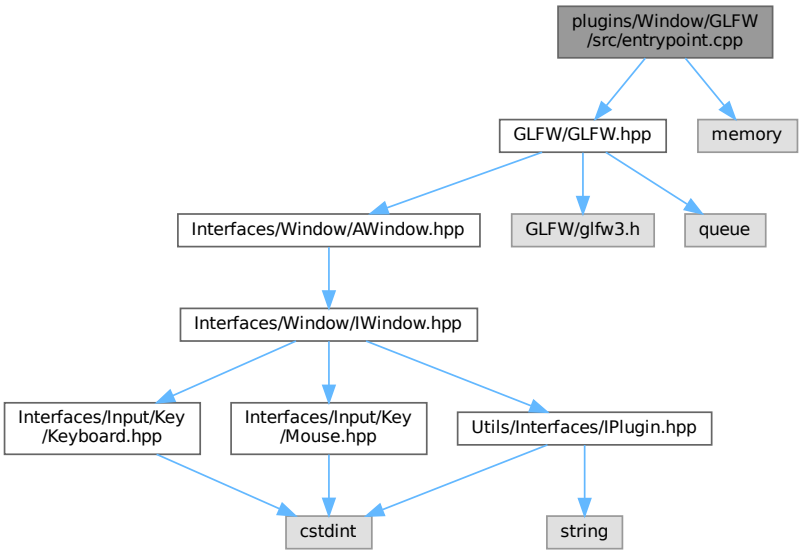
Go to the documentation of this file.

```
00001 #include ”Cocoa/Cocoa.hpp”
00002
00003 #include <memory>
00004
00005 extern ”C”
00006 {
00007     PLUGIN_EXPORT cae::IWindow *entryPoint() { return std::make_unique<cae::Cocoa>().release(); }
00008 }
```

43.153 plugins/Window/GLFW/src/entrypoint.cpp File Reference

#include ”GLFW/GLFW.hpp”
#include <memory>

Include dependency graph for entrypoint.cpp:



Functions

- PLUGIN_EXPORT cae::IWindow * entryPoint ()

43.153.1 Function Documentation

43.153.1.1 entryPoint()

PLUGIN_EXPORT cae::IWindow * entryPoint ()
Definition at line 7 of file entrypoint.cpp.

43.154 entrypoint.cpp

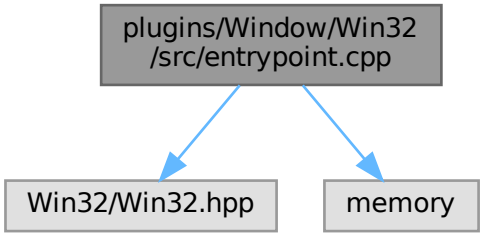
Go to the documentation of this file.

```
00001 #include "GLFW/GLFW.hpp"
00002
00003 #include <memory>
00004
00005 extern "C"
00006 {
00007     PLUGIN_EXPORT cae::IWindow *entryPoint() { return std::make_unique<cae::GLFW>().release(); }
00008 }
```

43.155 plugins/Window/Win32/src/entrypoint.cpp File Reference

#include "Win32/Win32.hpp"
#include <memory>

Include dependency graph for entrypoint.cpp:



Functions

- PLUGIN_EXPORT cae::IWindow * entryPoint ()

43.155.1 Function Documentation

43.155.1.1 entryPoint()

PLUGIN_EXPORT cae::IWindow * entryPoint ()
Definition at line 7 of file entrypoint.cpp.

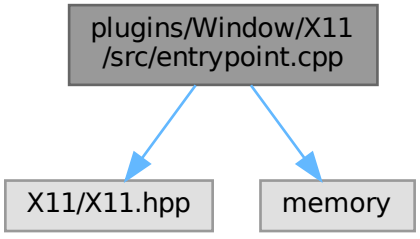
43.156 entrypoint.cpp

Go to the documentation of this file.

```
00001 #include "Win32/Win32.hpp"
00002
00003 #include <memory>
00004
00005 extern "C"
00006 {
00007     PLUGIN_EXPORT cae::IWindow *entryPoint() { return std::make_unique<cae::Win32>().release(); }
00008 }
```

43.157 plugins/Window/X11/src/entrypoint.cpp File Reference

#include "X11/X11.hpp"
#include <memory>
Include dependency graph for entrypoint.cpp:



Functions

- PLUGIN_EXPORT cae::IWindow * entryPoint ()

43.157.1 Function Documentation

43.157.1.1 entryPoint()

PLUGIN_EXPORT cae::IWindow * entryPoint ()
Definition at line 7 of file entrypoint.cpp.

43.158 entrypoint.cpp

Go to the documentation of this file.

```
00001 #include "X11/X11.hpp"
00002
00003 #include <memory>
00004
00005 extern "C"
00006 {
00007     PLUGIN_EXPORT cae::IWindow *entryPoint() { return std::make_unique<cae::X11>().release(); }
00008 }
```

00001

43.195 plugins/Network/Posix/include/Posix/Posix.hpp File Reference

43.196 Posix.hpp

Go to the documentation of this file.
00001

43.197 plugins/Network/WinSock/include/WinSock/WinSock.hpp File Reference

43.198 WinSock.hpp

Go to the documentation of this file.
00001

43.199 plugins/Renderer/DirectX12/include/DirectX12/DirectX12.hpp File Reference

43.200 DirectX12.hpp

Go to the documentation of this file.
00001

43.201 plugins/Renderer/Metal/include/Metal/Metal.hpp File Reference

43.202 Metal.hpp

Go to the documentation of this file.
00001

43.203 plugins/Renderer/OpenGL/include/OPGL/Context/↩ EGLContext.hpp File Reference

This file contains the EGLContext_ class declaration.

Namespaces

- namespace cae

43.203.1 Detailed Description

This file contains the EGLContext_ class declaration.
Definition in file EGLContext.hpp.

43.204 EGLContext.hpp

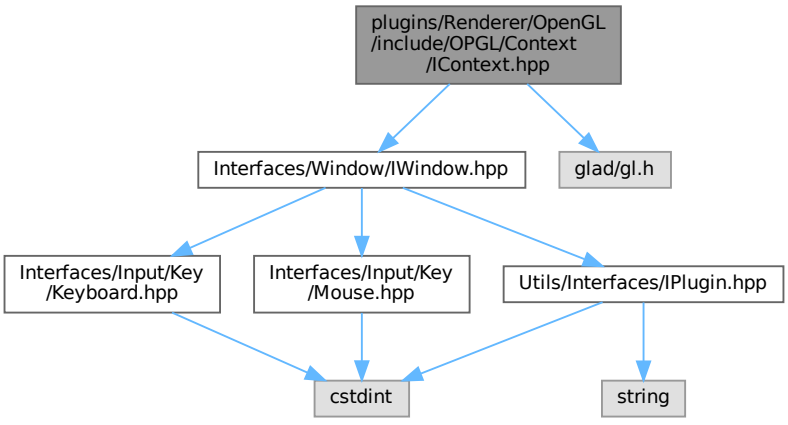
Go to the documentation of this file.

```
00001 ///
00002 /// @file EGLContext.hpp
00003 /// @brief This file contains the EGLContext_ class declaration
00004 /// @namespace cae
00005 ///
```

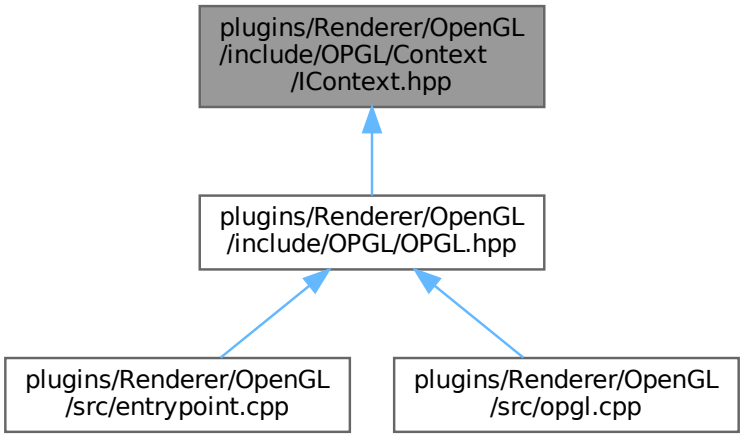
```
00006
00007 #pragma once
00008
00009 #ifdef ___linux___
00010
00011 #include "OPGL/Context/IContext.hpp"
00012
00013 #include <EGL/egl.h>
00014
00015 namespace cae
00016 {
00017
00018     ///
00019     /// @class EGLContext_
00020     /// @brief Implementation of IContext for Linux using EGL
00021     /// @namespace cae
00022     ///
00023     class EGLContext_ final : public IContext
00024     {
00025         public:
00026             explicit EGLContext_() = default;
00027             ~EGLContext_() override;
00028
00029             void initialize(const NativeWindowHandle &window) override;
00030
00031             void swapBuffers() override;
00032
00033             void setVSyncEnabled(bool enabled) override;
00034             [[nodiscard]] bool isVSyncEnabled() const override
00035             {
00036                 // EGL does not provide a direct way to query VSync status
00037                 return false;
00038             }
00039
00040         private:
00041             EGLDisplay m_display = EGL_NO_DISPLAY;
00042             EGLSurface m_surface = EGL_NO_SURFACE;
00043             EGLContext m_context = EGL_NO_CONTEXT;
00044
00045     }; // class EGLContext_
00046
00047 } // namespace cae
00048
00049 #endif
```

43.205 plugins/Renderer/OpenGL/include/OPGL/Context/↩ IContext.hpp File Reference

This file contains the IContext interface.
#include "Interfaces/Window/IWindow.hpp"
#include <glad/gl.h>
Include dependency graph for IContext.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- interface cae::IContext

  Interface for OpenGL context.

Namespaces

- namespace cae

43.205.1 Detailed Description

This file contains the IContext interface.
Definition in file IContext.hpp.

43.206 IContext.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file IContext.hpp
00003 /// @brief This file contains the IContext interface
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Window/IWindow.hpp"
00010
00011 #include <glad/gl.h>
00012
00013 namespace cae
00014 {
00015     ///
00016     /// @interface IContext
00017     /// @brief Interface for OpenGL context
00018     /// @namespace cae
00019     ///
00020     class IContext
00021     {
00022         public:
00023             virtual ~IContext() = default;
00024
00025             ///
00026             /// @param window The native window handle
00027             /// @brief Initialize the OpenGL context with the given window
```

```
00028           ///
00029           virtual void initialize(const NativeWindowHandle &window) = 0;
00030
00031           ///
00032           /// @brief Swap the front and back buffers
00033           ///
00034           virtual void swapBuffers() = 0;
00035
00036           ///
00037           /// @param enabled Whether VSync should be enabled
00038           /// @brief Enable or disable VSync
00039           ///
00040           virtual void setVSyncEnabled(bool enabled) = 0;
00041
00042           ///
00043           /// @return  Whether VSync is enabled
00044           /// @brief Check if VSync is enabled
00045           ///
00046           [[nodiscard]] virtual bool isVSyncEnabled() const = 0;
00047
00048           GladGLContext gl{nullptr};
00049
00050       }; // interface IContext
00051
00052 } // namespace cae
```

43.207 plugins/Renderer/OpenGL/include/OPGL/Context/↩ NSGLContext.hpp File Reference

This file contains the NSGLContext class declaration.

Namespaces

- namespace cae

43.207.1 Detailed Description

This file contains the NSGLContext class declaration.
Definition in file NSGLContext.hpp.

43.208 NSGLContext.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file NSGLContext.hpp
00003 /// @brief This file contains the NSGLContext class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #ifdef ___APPLE___
00010
00011 #include "OPGL/Context/IContext.hpp"
00012
00013 namespace cae
00014 {
00015
00016     ///
00017     /// @class NSGLContext
00018     /// @brief Implementation of IContext for macOS using NSGL
00019     /// @namespace cae
00020     ///
00021     class NSGLContext final : public IContext
00022     {
00023         public:
00024             NSGLContext() = default;
00025             ~NSGLContext() override;
00026
00027             void initialize(const NativeWindowHandle &window) override;
00028
00029             void swapBuffers() override;
00030
00031             void setVSyncEnabled(bool enabled) override;
00032             [[nodiscard]] bool isVSyncEnabled() const override;
00033
```

```
00034         private:
00035             void *m_context = nullptr;
00036
00037     }; // class NSGLContext
00038
00039 } // namespace cae
00040
00041 #endif
```

43.209 plugins/Renderer/OpenGL/include/OPGL/Context/WGLContext.hpp File Reference

This file contains the WGLContext class declaration.

Namespaces

- namespace cae

43.209.1 Detailed Description

This file contains the WGLContext class declaration.
Definition in file WGLContext.hpp.

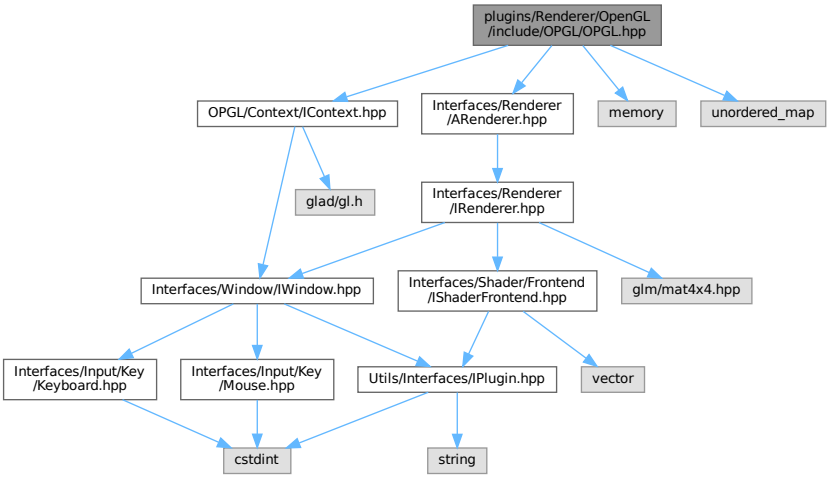
43.210 WGLContext.hpp

Go to the documentation of this file.

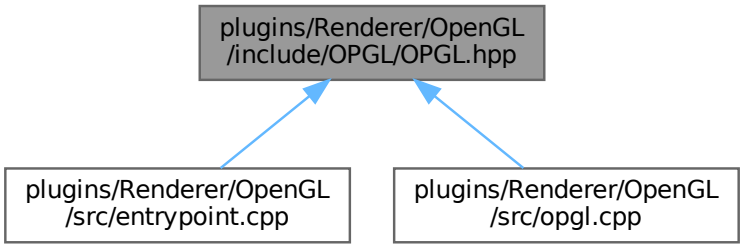
```
00001 ///
00002 /// @file WGLContext.hpp
00003 /// @brief This file contains the WGLContext class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #ifdef _WIN32
00010
00011 #include "OPGL/Context/IContext.hpp"
00012
00013 #include <Windows.h>
00014
00015 namespace cae
00016 {
00017
00018     ///
00019     /// @class WGLContext
00020     /// @brief Implementation of IContext for Windows using WGL
00021     /// @namespace cae
00022     ///
00023     class WGLContext final : public IContext
00024     {
00025         public:
00026             WGLContext() = default;
00027             ~WGLContext() override;
00028
00029             void initialize(const NativeWindowHandle &window) override;
00030             void swapBuffers() override;
00031             void setVSyncEnabled(bool enabled) override;
00032             [[nodiscard]] bool isVSyncEnabled() const override
00033             {
00034                 // WGL does not provide a direct way to query VSync status
00035                 return false;
00036             }
00037
00038         private:
00039             HWND m_hwnd = nullptr;
00040             HDC m_hdc = nullptr;
00041             HGLRC m_hglrc = nullptr;
00042
00043     }; // class WGLContext
00044
00045 } // namespace cae
00046
00047 #endif
```

43.211 plugins/Renderer/OpenGL/include/OPGL/OPGL.hpp File Reference

This file contains the OPGL class declaration.
#include "OPGL/Context/IContext.hpp"
#include "Interfaces/Renderer/ARenderer.hpp"
#include <memory>
#include <unordered_map>
Include dependency graph for OPGL.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::Mesh
- class cae::OPGL

  Class for the OpenGL plugin.

Namespaces

- namespace cae

43.211.1 Detailed Description

This file contains the OPGL class declaration.
Definition in file OPGL.hpp.

43.212 OPGL.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file OPGL.hpp
00003 /// @brief This file contains the OPGL class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "OPGL/Context/IContext.hpp"
00010
00011 #include "Interfaces/Renderer/ARenderer.hpp"
00012
00013 #include <memory>
00014 #include <unordered_map>
00015
00016 namespace cae
00017 {
00018
00019     struct Mesh
00020     {
00021         GLuint vao = 0;
00022         GLuint vbo = 0;
00023         GLuint ebo = 0;
00024         GLsizei vertexCount = 0;
00025     };
00026
00027     ///
00028     /// @class OPGL
00029     /// @brief Class for the OpenGL plugin
00030     /// @namespace cae
00031     ///
00032     class OPGL final : public ARenderer
00033     {
00034         public:
00035             OPGL() = default;
00036             ~OPGL() override = default;
00037
00038             OPGL(const OPGL &) = delete;
00039             OPGL &operator=(const OPGL &) = delete;
00040             OPGL(OPGL &&) = delete;
00041             OPGL &operator=(OPGL &&) = delete;
00042
00043             [[nodiscard]] std::string getName() const override { return "OpenGL"; }
00044             [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::RENDERER; }
00045             [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::ALL; }
00046
00047             void setVSyncEnabled(const bool enabled) override { m_context->setVSyncEnabled(enabled); }
00048             void setClearColor(const Color &color) override
00049             {
00050                 auto &gl = m_context->gl;
00051                 gl.ClearColor(color.r, color.g, color.b, color.a);
00052             }
00053
00054             [[nodiscard]] bool isVSyncEnabled() const override { return m_context->isVSyncEnabled(); }
00055
00056             void initialize(const NativeWindowHandle &nativeWindowHandle, const Color &clearColor) override;
00057             void createPipeline(const ShaderID &id, const ShaderIRModule &vertex,
00058                                 const ShaderIRModule &fragment) override;
00059             void draw(const WindowSize &windowSize, const ShaderID &shaderId, glm::mat4 mvp) override;
00060             void createMesh(const std::vector<float> &vertices) override;
00061
00062         private:
00063             std::unique_ptr<IContext> m_context;
00064             std::unordered_map<ShaderID, GLuint> m_programs;
00065             Mesh m_mesh;
00066
00067             GLuint m_ubo;
00068             static GLuint createGLShader(GLenum type, const ShaderIRModule &data, const GladGLContext &gl);
00069
00070     }; // class OPGL
00071
00072 } // namespace cae
```

43.213 plugins/Renderer/OpenGL/src/context/EGLContext.cpp File Reference

43.214 EGLContext.cpp

Go to the documentation of this file.

```
00001 #ifdef __linux__
00002
00003 #include "OPGL/Context/EGLContext.hpp"
00004
00005 #include <stdexcept>
00006
00007 cae::EGLContext_::~EGLContext_()
00008 {
00009     if (m_display != EGL_NO_DISPLAY)
00010     {
00011         eglMakeCurrent(m_display, EGL_NO_SURFACE, EGL_NO_SURFACE, EGL_NO_CONTEXT);
00012         if (m_context != EGL_NO_CONTEXT)
00013         {
00014             eglDestroyContext(m_display, m_context);
00015         }
00016         if (m_surface != EGL_NO_SURFACE)
00017         {
00018             eglDestroySurface(m_display, m_surface);
00019         }
00020         eglTerminate(m_display);
00021     }
00022 }
00023
00024 void cae::EGLContext_::initialize(const NativeWindowHandle &window)
00025 {
00026     if (eglBindAPI(EGL_OPENGL_API) == EGL_FALSE)
00027     {
00028         throw std::runtime_error("Failed to bind OpenGL API");
00029     }
00030
00031     m_display = eglGetDisplay(window.display);
00032     if (m_display == EGL_NO_DISPLAY)
00033     {
00034         throw std::runtime_error("Failed to get EGL display");
00035     }
00036
00037     if (eglInitialize(m_display, nullptr, nullptr) == EGL_FALSE)
00038     {
00039         throw std::runtime_error("Failed to initialize EGL");
00040     }
00041
00042     constexpr EGLint configAttribs[] = {EGL_RED_SIZE,
00043                                         8,
00044                                         EGL_GREEN_SIZE,
00045                                         8,
00046                                         EGL_BLUE_SIZE,
00047                                         8,
00048                                         EGL_DEPTH_SIZE,
00049                                         24,
00050                                         EGL_SURFACE_TYPE,
00051                                         EGL_WINDOW_BIT,
00052                                         EGL_RENDERABLE_TYPE,
00053                                         EGL_OPENGL_BIT,
00054                                         EGL_NONE};
00055
00056     EGLConfig config = nullptr;
00057     EGLint numConfigs = 0;
00058     if (eglChooseConfig(m_display, configAttribs, &config, 1, &numConfigs) == EGL_FALSE || numConfigs == 0)
00059     {
00060         throw std::runtime_error("Failed to choose EGL config");
00061     }
00062
00063     m_surface =
00064         eglCreateWindowSurface(m_display, config, reinterpret_cast<EGLNativeWindowType>(window.window), nullptr);
00065     if (m_surface == EGL_NO_SURFACE)
00066     {
00067         throw std::runtime_error("Failed to create EGL surface");
00068     }
00069
00070     constexpr EGLint contextAttribs[] = {EGL_CONTEXT_CLIENT_VERSION, 3, EGL_NONE};
00071     m_context = eglCreateContext(m_display, config, EGL_NO_CONTEXT, contextAttribs);
00072     if (m_context == EGL_NO_CONTEXT)
00073     {
00074         throw std::runtime_error("Failed to create EGL context");
00075     }
00076
00077     if (eglMakeCurrent(m_display, m_surface, m_surface, m_context) == EGL_FALSE)
00078     {
```

```
00079         throw std::runtime_error("Failed to make EGL context current");
00080     }
00081
00082     eglSwapInterval(m_display, 0);
00083
00084     if (gladLoadGLContext(&gl, eglGetProcAddress) == 0)
00085     {
00086         throw std::runtime_error("Failed to initialize GLAD");
00087     }
00088 }
00089
00090 void cae::EGLContext_::swapBuffers()
00091 {
00092     if (m_display != EGL_NO_DISPLAY && m_surface != EGL_NO_SURFACE)
00093     {
00094         eglSwapBuffers(m_display, m_surface);
00095     }
00096 }
00097
00098 void cae::EGLContext_::setVSyncEnabled(const bool enabled)
00099 {
00100     if (m_display != EGL_NO_DISPLAY)
00101     {
00102         eglSwapInterval(m_display, enabled ? 1 : 0);
00103     }
00104 }
00105
00106 #endif
```

43.215 plugins/Renderer/OpenGL/src/context/WGLContext.cpp File Reference

43.216 WGLContext.cpp

Go to the documentation of this file.

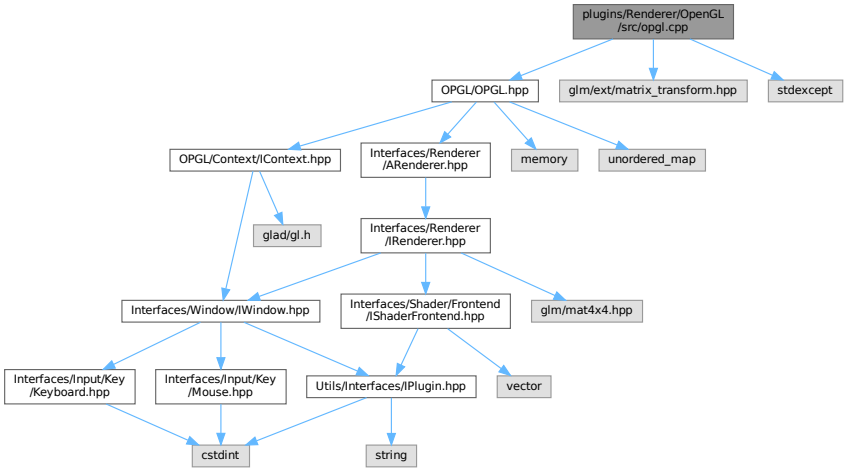
```
00001 #ifdef _WIN32
00002
00003 #include "OPGL/Context/WGLContext.hpp"
00004
00005 #include "Utils/Logger.hpp"
00006
00007 #include <stdexcept>
00008
00009 typedef HGLRC(WINAPI *PFNWGLCREATECONTEXTATTRIBSARBPROC)(HDC, HGLRC, const int *);
00010 typedef const char *(WINAPI *PFNWGLGETEXTENSIONSSTRINGARBPROC)(HDC);
00011
00012 #ifndef WGL_CONTEXT_MAJOR_VERSION_ARB
00013 #define WGL_CONTEXT_MAJOR_VERSION_ARB 0x2091
00014 #endif
00015 #ifndef WGL_CONTEXT_MINOR_VERSION_ARB
00016 #define WGL_CONTEXT_MINOR_VERSION_ARB 0x2092
00017 #endif
00018 #ifndef WGL_CONTEXT_FLAGS_ARB
00019 #define WGL_CONTEXT_FLAGS_ARB 0x2094
00020 #endif
00021 #ifndef WGL_CONTEXT_FORWARD_COMPATIBLE_BIT_ARB
00022 #define WGL_CONTEXT_FORWARD_COMPATIBLE_BIT_ARB 0x0002
00023 #endif
00024 #ifndef WGL_CONTEXT_PROFILE_MASK_ARB
00025 #define WGL_CONTEXT_PROFILE_MASK_ARB 0x9126
00026 #endif
00027 #ifndef WGL_CONTEXT_CORE_PROFILE_BIT_ARB
00028 #define WGL_CONTEXT_CORE_PROFILE_BIT_ARB 0x00000001
00029 #endif
00030
00031 static HMODULE g_opengl32 = nullptr;
00032
00033 static void *win32GetGLProc(const char *name)
00034 {
00035     auto *proc = (void *)wglGetProcAddress(name);
00036
00037     if (proc == nullptr || proc == reinterpret_cast<void *>(0x1) || proc == reinterpret_cast<void *>(0x2) ||
00038         proc == reinterpret_cast<void *>(0x3) || proc == reinterpret_cast<void *>(-1))
00039     {
00040         if (g_opengl32 == nullptr)
00041         {
00042             g_opengl32 = LoadLibraryA("opengl32.dll");
00043         }
00044
00045         proc = (void *)GetProcAddress(g_opengl32, name);
00046     }
00047
```

```
00048     return proc;
00049 }
00050
00051 cae::WGLContext::~WGLContext()
00052 {
00053     if (m_hglrc != nullptr)
00054     {
00055         wglMakeCurrent(nullptr, nullptr);
00056         wglDeleteContext(m_hglrc);
00057     }
00058     if ((m_hdc != nullptr) && (m_hwnd != nullptr))
00059     {
00060         ReleaseDC(m_hwnd, m_hdc);
00061     }
00062 }
00063
00064 void cae::WGLContext::initialize(const NativeWindowHandle &window)
00065 {
00066     m_hwnd = static_cast<HWND>(window.window);
00067     m_hdc = GetDC(m_hwnd);
00068     if (m_hdc == nullptr)
00069     {
00070         throw std::runtime_error("Failed to get HDC from HWND");
00071     }
00072
00073     PIXELFORMATDESCRIPTOR pfd{};
00074     pfd.nSize = sizeof(pfd);
00075     pfd.nVersion = 1;
00076     pfd.dwFlags = PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
00077     pfd.iPixelType = PFD_TYPE_RGBA;
00078     pfd.cColorBits = 32;
00079     pfd.cDepthBits = 24;
00080     pfd.iLayerType = PFD_MAIN_PLANE;
00081
00082     const int pf = ChoosePixelFormat(m_hdc, &pfd);
00083     if (pf == 0)
00084     {
00085         throw std::runtime_error("Failed to choose pixel format");
00086     }
00087     if (SetPixelFormat(m_hdc, pf, &pfd) == 0)
00088     {
00089         throw std::runtime_error("Failed to set pixel format");
00090     }
00091
00092     const HGLRC tempContext = wglCreateContext(m_hdc);
00093     if (tempContext == nullptr)
00094     {
00095         throw std::runtime_error("Failed to create temporary WGL context");
00096     }
00097     if (wglMakeCurrent(m_hdc, tempContext) == 0)
00098     {
00099         throw std::runtime_error("Failed to make temporary context current");
00100     }
00101
00102     const auto wglCreateContextAttribsARB =
00103         reinter-
    pret_cast<PFNWGLCREATECONTEXTATTRIBSARBPROC>(wglGetProcAddress("wglCreateContextAttribsARB"));
00104
00105     if (wglCreateContextAttribsARB != nullptr)
00106     {
00107         const int attribs[] = {WGL_CONTEXT_MAJOR_VERSION_ARB,
00108                                4,
00109                                WGL_CONTEXT_MINOR_VERSION_ARB,
00110                                6,
00111                                WGL_CONTEXT_FLAGS_ARB,
00112                                WGL_CONTEXT_FORWARD_COMPATIBLE_BIT_ARB,
00113                                WGL_CONTEXT_PROFILE_MASK_ARB,
00114                                WGL_CONTEXT_CORE_PROFILE_BIT_ARB,
00115                                0};
00116
00117         const HGLRC modernContext = wglCreateContextAttribsARB(m_hdc, nullptr, attribs);
00118         if (modernContext == nullptr)
00119         {
00120             throw std::runtime_error("Failed to create modern WGL context");
00121         }
00122
00123         wglMakeCurrent(nullptr, nullptr);
00124         wglDeleteContext(tempContext);
00125
00126         m_hglrc = modernContext;
00127         if (wglMakeCurrent(m_hdc, m_hglrc) == 0)
00128         {
00129             throw std::runtime_error("Failed to make modern WGL context current");
00130         }
00131     }
00132     else
00133     {
```

```
00134         utl::Logger::log("wglCreateContextAttribsARB not available, using legacy context\n", utl::LogLevel::WARNING);
00135         m_hglrc = tempContext;
00136     }
00137
00138     if (wglGetCurrentContext() != m_hglrc)
00139     {
00140         throw std::runtime_error("Current WGL context is not the one just created");
00141     }
00142     if (const int version = gladLoadGLContext(&gl, reinterpret_cast<GLADloadfunc>(win32GetGLProc)); version == 0)
00143     {
00144         throw std::runtime_error("Failed to initialize GLAD MX (Windows)");
00145     }
00146     if (gl.Enable != nullptr)
00147     {
00148         gl.Enable(GL_DEBUG_OUTPUT);
00149 #ifdef CAE_DEBUG
00150         gl.DebugMessageCallback([](GLenum source, GLenum type, GLuint id, GLenum severity, GLsizei length,
00151                                    const GLchar *message, const void *userParam)
00152                                 { utl::Logger::log("[GL DEBUG] " + std::string(message), utl::LogLevel::WARNING); },
00153                                 nullptr);
00154 #endif
00155     }
00156 }
00157
00158 void cae::WGLContext::swapBuffers()
00159 {
00160     if (m_hdc != nullptr)
00161     {
00162         SwapBuffers(m_hdc);
00163     }
00164 }
00165
00166 void cae::WGLContext::setVSyncEnabled(const bool enabled)
00167 {
00168     using PFNWGLSWAPINTERVALEXTPROC = BOOL(WINAPI *)(int);
00169     static auto wglSwapIntervalEXT =
00170         reinterpret_cast<PFNWGLSWAPINTERVALEXTPROC>(wglGetProcAddress("wglSwapIntervalEXT"));
00171     if (wglSwapIntervalEXT != nullptr)
00172     {
00173         wglSwapIntervalEXT(enabled ? 1 : 0);
00174     }
00175 }
00176
00177 #endif
```

43.217 plugins/Renderer/OpenGL/src/opgl.cpp File Reference

#include "OPGL/OPGL.hpp"
#include <glm/ext/matrix_transform.hpp>
#include <stdexcept>
Include dependency graph for opgl.cpp:



43.218 opgl.cpp

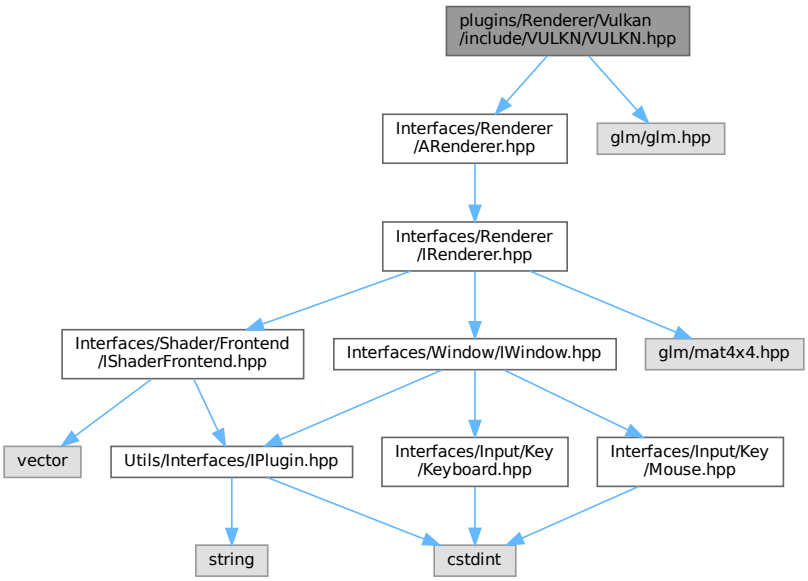
Go to the documentation of this file.

```
00001 #include "OPGL/OPGL.hpp"
00002
00003 #ifdef __linux__
00004 #include "OPGL/Context/EGLContext.hpp"
00005 #elifdef _WIN32
00006 #include "OPGL/Context/WGLContext.hpp"
00007 #elifdef __APPLE__
00008 #include "OPGL/Context/NSGLContext.hpp"
00009 #endif
00010
00011 #include <glm/ext/matrix_transform.hpp>
00012
00013 #include <stdexcept>
00014
00015 void cae::OPGL::initialize(const NativeWindowHandle &nativeWindowHandle, const Color &clearColor)
00016 {
00017 #ifdef __linux__
00018     m_context = std::make_unique<EGLContext_>();
00019 #elifdef _WIN32
00020     m_context = std::make_unique<WGLContext>();
00021 #elifdef __APPLE__
00022     m_context = std::make_unique<NSGLContext>();
00023 #endif
00024
00025     m_context->initialize(nativeWindowHandle);
00026     const auto &gl = m_context->gl;
00027
00028     gl.Enable(GL_DEPTH_TEST);
00029     gl.ClearColor(clearColor.r, clearColor.g, clearColor.b, clearColor.a);
00030
00031     gl.GenBuffers(1, &m_ubo);
00032     gl.BindBuffer(GL_UNIFORM_BUFFER, m_ubo);
00033     gl.BufferData(GL_UNIFORM_BUFFER, sizeof(glm::mat4), nullptr, GL_DYNAMIC_DRAW);
00034     gl.BindBufferBase(GL_UNIFORM_BUFFER, 0, m_ubo);
00035     gl.BindBuffer(GL_UNIFORM_BUFFER, 0);
00036 }
00037
00038 void cae::OPGL::draw(const WindowSize &windowSize, const ShaderID &shaderId, const glm::mat4 mvp)
00039 {
00040     const auto &gl = m_context->gl;
00041     gl.Viewport(0, 0, windowSize.width, windowSize.height);
00042     gl.Clear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
00043
00044     gl.UseProgram(m_programs.at(shaderId));
00045     gl.BindBuffer(GL_UNIFORM_BUFFER, m_ubo);
00046     gl.BufferSubData(GL_UNIFORM_BUFFER, 0, sizeof(glm::mat4), &mvp);
00047
00048     // binding = 0 car dans le shader: layout(binding = 0)
00049     gl.BindVertexArray(m_mesh.vao);
00050     gl.DrawArrays(GL_TRIANGLES, 0, m_mesh.vertexCount);
00051     gl.BindVertexArray(0);
00052
00053     m_context->swapBuffers();
00054 }
00055
00056 void cae::OPGL::createPipeline(const ShaderID &id, const ShaderIRModule &vertex, const ShaderIRModule &fragment)
00057 {
00058     const auto &gl = m_context->gl;
00059     const GLuint program = gl.CreateProgram();
00060
00061     const GLuint vs = createGLShader(GL_VERTEX_SHADER, vertex, gl);
00062     const GLuint fs = createGLShader(GL_FRAGMENT_SHADER, fragment, gl);
00063
00064     gl.AttachShader(program, vs);
00065     gl.AttachShader(program, fs);
00066     gl.LinkProgram(program);
00067
00068     GLint success = 0;
00069     gl.GetProgramiv(program, GL_LINK_STATUS, &success);
00070     if (success == 0)
00071     {
00072         char log[512];
00073         gl.GetProgramInfoLog(program, 512, nullptr, log);
00074         throw std::runtime_error(log);
00075     }
00076
00077     gl.DeleteShader(vs);
00078     gl.DeleteShader(fs);
00079
00080     m_programs[id] = program;
00081 }
00082
00083 void cae::OPGL::createMesh(const std::vector<float> &vertices)
```

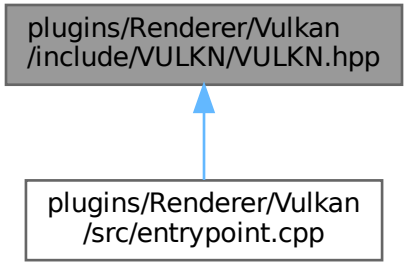
```
00084 {
00085     const auto &gl = m_context->gl;
00086     Mesh mesh{};
00087     mesh.vertexCount = static_cast<GLsizei>(vertices.size() / 5);
00088
00089     gl.GenVertexArrays(1, &mesh.vao);
00090     gl.GenBuffers(1, &mesh.vbo);
00091
00092     gl.BindVertexArray(mesh.vao);
00093     gl.BindBuffer(GL_ARRAY_BUFFER, mesh.vbo);
00094     gl.BufferData(GL_ARRAY_BUFFER, vertices.size() * sizeof(float), vertices.data(), GL_STATIC_DRAW);
00095
00096     gl.VertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), static_cast<void *>(0));
00097     gl.EnableVertexAttribArray(0);
00098
00099     gl.VertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * sizeof(float), reinterpret_cast<void *>(3 * sizeof(float)));
00100     gl.EnableVertexAttribArray(1);
00101
00102     gl.BindBuffer(GL_ARRAY_BUFFER, 0);
00103     gl.BindVertexArray(0);
00104
00105     m_mesh = mesh;
00106 }
00107
00108 GLuint cae::OPGL::createGLShader(const GLenum type, const ShaderIRModule &data, const GladGLContext &gl)
00109 {
00110     const GLuint shader = gl.CreateShader(type);
00111
00112     gl.ShaderBinary(1, &shader, GL_SHADER_BINARY_FORMAT_SPIR_V, data.spirv.data(),
00113                     static_cast<GLsizei>(data.spirv.size() * sizeof(uint32_t)));
00114
00115     gl.SpecializeShader(shader, data.entryPoint.c_str(), 0, nullptr, nullptr);
00116
00117     return shader;
00118 }
```

43.219 plugins/Renderer/Vulkan/include/VULKN/VULKN.hpp File Reference

This file contains the VULKN class declaration.
#include ”Interfaces/Renderer/ARenderer.hpp”
#include <glm/glm.hpp>
Include dependency graph for VULKN.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class cae::VULKN

  Class for the Vulkan plugin.

Namespaces

- namespace cae

43.219.1 Detailed Description

This file contains the VULKN class declaration.
Definition in file VULKN.hpp.

43.220 VULKN.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file VULKN.hpp
00003 /// @brief This file contains the VULKN class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Renderer/ARenderer.hpp"
00010
00011 #include <glm/glm.hpp>
00012
00013 namespace cae
00014 {
00015
00016     ///
00017     /// @class VULKN
00018     /// @brief Class for the Vulkan plugin
00019     /// @namespace cae
00020     ///
00021     class VULKN final : public ARenderer
00022     {
00023
00024         public:
00025             VULKN() = default;
00026             ~VULKN() override = default;
00027
00028             VULKN(const VULKN &) = delete;
00029             VULKN &operator=(const VULKN &) = delete;
00030             VULKN(VULKN &&) = delete;
00031             VULKN &operator=(VULKN &&) = delete;
00032
00033             [[nodiscard]] std::string getName() const override { return "Vulkan"; }
00034             [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::RENDERER; }
```

```
00035           [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::ALL; }
00036
00037           void setVSyncEnabled(bool enabled) override {}
00038           void setClearColor(const Color &color) override {}
00039
00040           [[nodiscard]] bool isVSyncEnabled() const override { return false; }
00041
00042           void initialize(const NativeWindowHandle &nativeWindowHandle, const Color &clearColor) override {}
00043           void createPipeline(const ShaderID &id, const ShaderIRModule &vertex,
00044                               const ShaderIRModule &fragment) override
00045           {
00046           }
00047           void draw(const WindowSize &windowSize, const ShaderID &shaderId, glm::mat4 mvp) override {}
00048           void createMesh(const std::vector<float> &vertices) override {}
00049
00050       }; // class VULKN
00051
00052 } // namespace cae
```

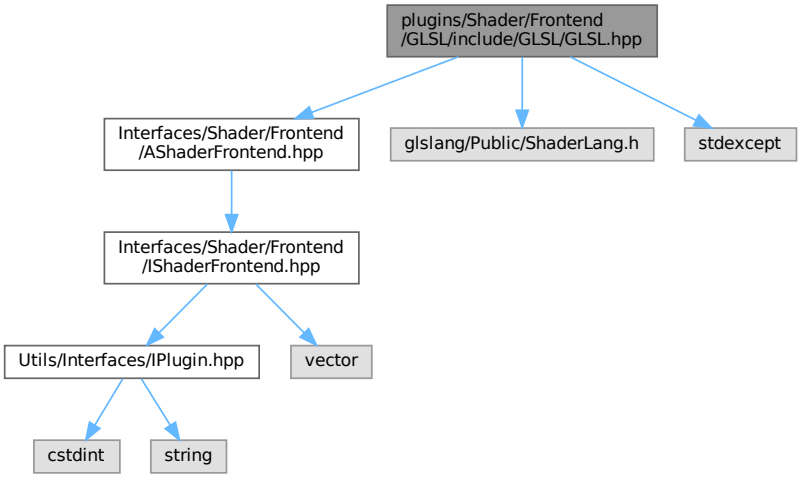
43.221 plugins/Renderer/Vulkan/src/VULKN.cpp File Reference

43.222 VULKN.cpp

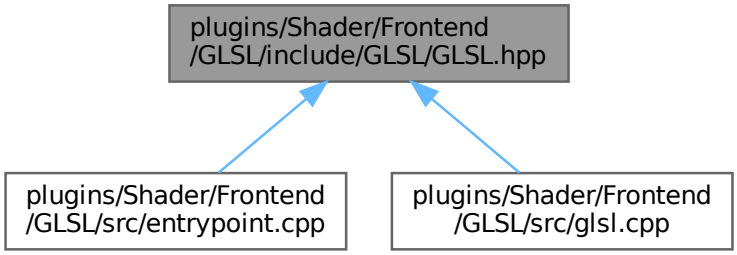
Go to the documentation of this file.
00001

43.223 plugins/Shader/Frontend/GLSL/include/GLSL/GLSL.hpp File Reference

This file contains the GLSL class declaration.
#include ”Interfaces/Shader/Frontend/AShaderFrontend.hpp”
#include <glslang/Public/ShaderLang.h>
#include <stdexcept>
Include dependency graph for GLSL.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class cae::GLSL

  Class for the GLSL plugin.

Namespaces

- namespace cae

Variables

- constexpr auto cae::VERSION = 450

43.223.1 Detailed Description

This file contains the GLSL class declaration.
Definition in file GLSL.hpp.

43.224 GLSL.hpp

Go to the documentation of this file.

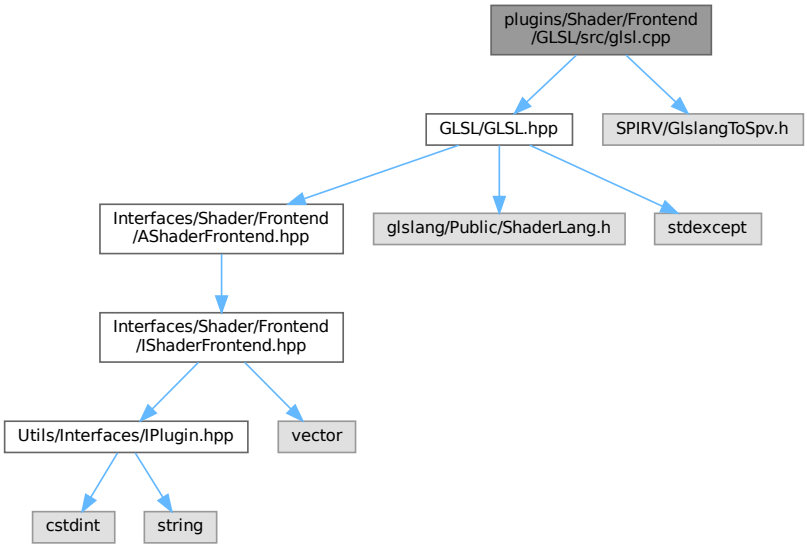
```
00001 ///
00002 /// @file GLSL.hpp
00003 /// @brief This file contains the GLSL class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Shader/Frontend/AShaderFrontend.hpp"
00010
00011 #include <glslang/Public/ShaderLang.h>
00012
00013 #include <stdexcept>
00014
00015 namespace cae
00016 {
00017
00018     constexpr auto VERSION = 450;
00019
00020     ///
00021     /// @class GLSL
00022     /// @brief Class for the GLSL plugin
00023     /// @namespace cae
00024     ///
00025     class GLSL final : public AShaderFrontend
00026     {
00027
00028         public:
00029             GLSL() = default;
```

```
00030           ~GLSL() override = default;
00031
00032           GLSL(const GLSL &) = delete;
00033           GLSL &operator=(const GLSL &) = delete;
00034           GLSL(GLSL &&) = delete;
00035           GLSL &operator=(GLSL &&) = delete;
00036
00037           [[nodiscard]] std::string getName() const override { return "GLSL"; }
00038           [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::SHADER_FRONTEND; }
00039           [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::ALL; }
00040
00041           [[nodiscard]] ShaderSourceType sourceType() const override { return ShaderSourceType::GLSL; }
00042
00043           ShaderIRModule compile(const ShaderSourceDesc &desc) override
00044           {
00045               ShaderIRModule ir;
00046               ir.id = desc.id;
00047               ir.stage = desc.stage;
00048               ir.entryPoint = "main";
00049               ir.spirv = compileGLSLtoSPIRV(desc.source, desc.stage);
00050               return ir;
00051           }
00052
00053       private:
00054           static EShLanguage shaderStageToESh(const ShaderStage stage)
00055           {
00056               switch (stage)
00057               {
00058                   case ShaderStage::VERTEX:
00059                       return EShLangVertex;
00060                   case ShaderStage::FRAGMENT:
00061                       return EShLangFragment;
00062                   case ShaderStage::GEOMETRY:
00063                       return EShLangGeometry;
00064                   case ShaderStage::COMPUTE:
00065                       return EShLangCompute;
00066                   default:
00067                       throw std::runtime_error("Unsupported ShaderStage");
00068               }
00069           }
00070
00071           static std::vector<uint32_t> compileGLSLtoSPIRV(const std::string &src, ShaderStage stage);
00072
00073       }; // class GLSL
00074
00075 } // namespace cae
```

43.225 plugins/Shader/Frontend/GLSL/src/glsl.cpp File Reference

#include "GLSL/GLSL.hpp"
#include <SPIRV/GlslangToSpv.h>

Include dependency graph for glsl.cpp:



43.226 glsl.cpp

Go to the documentation of this file.

```
00001 #include "GLSL/GLSL.hpp"
00002
00003 #include <SPIRV/GlslangToSpv.h>
00004
00005 std::vector<uint32_t> cae::GLSL::compileGLSLtoSPIRV(const std::string &src, const ShaderStage stage)
00006 {
00007     static bool glslangInitialized = false;
00008     if (!glslangInitialized)
00009     {
00010         glslang::InitializeProcess();
00011         glslangInitialized = true;
00012     }
00013
00014     const EShLanguage lang = shaderStageToESh(stage);
00015     glslang::TShader shader(lang);
00016     const char *shaderStrings[1] = {src.c_str()};
00017     shader.setStrings(shaderStrings, 1);
00018
00019     shader.setEnvInput(glslang::EShSourceGlsl, lang, glslang::EShClientVulkan, VERSION);
00020     shader.setEnvClient(glslang::EShClientVulkan, glslang::EShTargetVulkan_1_3);
00021     shader.setEnvTarget(glslang::EShTargetSpv, glslang::EShTargetSpv_1_6);
00022
00023     TBuiltInResource Resources = {};
00024     Resources.maxLights = 32;
00025     Resources.maxClipPlanes = 6;
00026     Resources.maxTextureUnits = 32;
00027     Resources.maxTextureCoords = 32;
00028     Resources.maxVertexAttribs = 64;
00029     Resources.maxVertexUniformComponents = 4096;
00030     Resources.maxVaryingFloats = 64;
00031     Resources.maxVertexTextureImageUnits = 32;
00032     Resources.maxCombinedTextureImageUnits = 80;
00033     Resources.maxTextureImageUnits = 32;
00034     Resources.maxFragmentUniformComponents = 4096;
00035     Resources.maxDrawBuffers = 32;
00036     Resources.maxVertexUniformVectors = 128;
00037     Resources.maxVaryingVectors = 8;
00038     Resources.maxFragmentUniformVectors = 16;
00039     Resources.maxVertexOutputVectors = 16;
00040     Resources.maxFragmentInputVectors = 15;
00041     Resources.minProgramTexelOffset = -8;
00042     Resources.maxProgramTexelOffset = 7;
00043     Resources.maxClipDistances = 8;
00044     Resources.maxComputeWorkGroupCountX = 65535;
00045     Resources.maxComputeWorkGroupCountY = 65535;
```

```
00046     Resources.maxComputeWorkGroupCountZ = 65535;
00047     Resources.maxComputeWorkGroupSizeX = 1024;
00048     Resources.maxComputeWorkGroupSizeY = 1024;
00049     Resources.maxComputeWorkGroupSizeZ = 64;
00050     Resources.maxComputeUniformComponents = 1024;
00051     Resources.maxComputeTextureImageUnits = 16;
00052     Resources.maxComputeImageUniforms = 8;
00053     Resources.maxComputeAtomicCounters = 8;
00054     Resources.maxComputeAtomicCounterBuffers = 1;
00055     Resources.maxVaryingComponents = 60;
00056     Resources.maxVertexOutputComponents = 64;
00057     Resources.maxGeometryInputComponents = 64;
00058     Resources.maxGeometryOutputComponents = 128;
00059     Resources.maxFragmentInputComponents = 128;
00060     Resources.maxImageUnits = 8;
00061     Resources.maxCombinedImageUnitsAndFragmentOutputs = 8;
00062     Resources.maxCombinedShaderOutputResources = 8;
00063
00064     constexpr auto messages = static_cast<EShMessages>(EShMsgSpvRules | EShMsgVulkanRules);
00065
00066     if (!shader.parse(&Resources, VERSION, false, messages))
00067     {
00068         throw std::runtime_error("GLSL parsing failed: " + std::string(shader.getInfoLog()));
00069     }
00070
00071     glslang::TProgram program;
00072     program.addShader(&shader);
00073
00074     if (!program.link(messages))
00075     {
00076         throw std::runtime_error("GLSL linking failed: " + std::string(program.getInfoLog()));
00077     }
00078
00079     std::vector<uint32_t> spirv;
00080     glslang::GlslangToSpv(*program.getIntermediate(lang), spirv);
00081
00082     return spirv;
00083 }
```

43.227 plugins/Shader/Frontend/HLSL/include/HLSL/HLSL.hpp File Reference

43.228 HLSL.hpp

Go to the documentation of this file.

```
00001
```

43.229 plugins/Shader/Frontend/MSL/include/MSL/MSL.hpp File Reference

43.230 MSL.hpp

Go to the documentation of this file.

```
00001
```

43.231 plugins/Shader/Frontend/WGSL/include/WGSL/WGSL.hpp File Reference

43.232 WGSL.hpp

Go to the documentation of this file.

```
00001
```

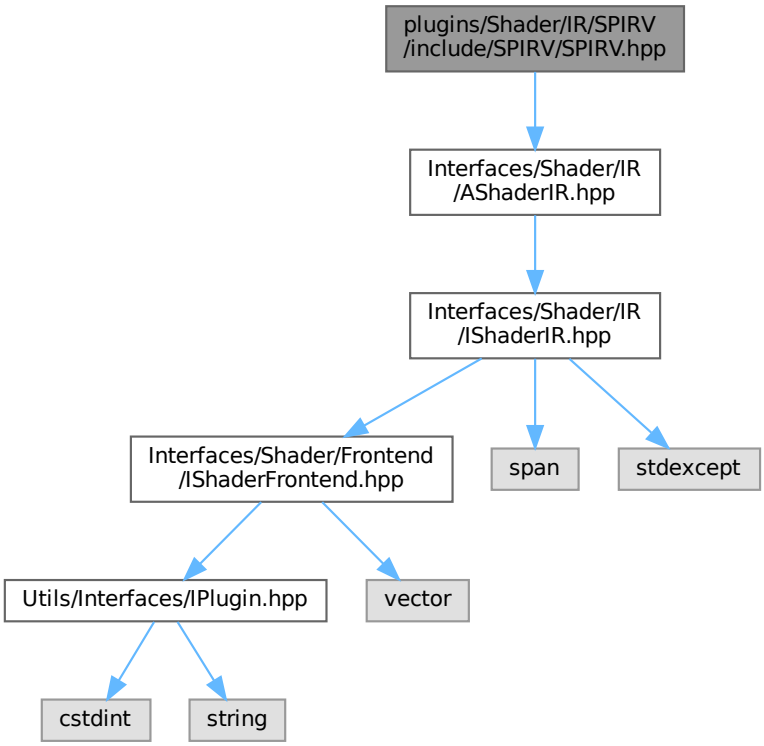
43.233 plugins/Shader/IR/DXC/include/DXC/DXC.hpp File Reference

43.234 DXC.hpp

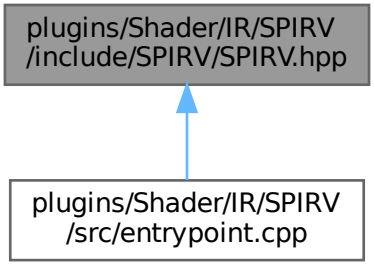
Go to the documentation of this file.
00001

43.235 plugins/Shader/IR/SPIRV/include/SPIRV/SPIRV.hpp File Reference

This file contains the SpirvIR class declaration.
#include "Interfaces/Shader/IR/AShaderIR.hpp"
Include dependency graph for SPIRV.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class cae::SPIRV

  Class for the SPIR-V IR plugin.

Namespaces

- namespace cae

43.235.1 Detailed Description

This file contains the SpirvIR class declaration.
Definition in file SPIRV.hpp.

43.236 SPIRV.hpp

Go to the documentation of this file.

```
00001 ///
00002 /// @file SPIRV.hpp
00003 /// @brief This file contains the SpirvIR class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Shader/IR/AShaderIR.hpp"
00010
00011 namespace cae
00012 {
00013
00014     ///
00015     /// @class SPIRV
00016     /// @brief Class for the SPIR-V IR plugin
00017     /// @namespace cae
00018     ///
00019     class SPIRV final : public AShaderIR
00020     {
00021         public:
00022             SPIRV() = default;
00023             ~SPIRV() override = default;
00024
00025             SPIRV(const SPIRV &) = delete;
00026             SPIRV &operator=(const SPIRV &) = delete;
00027             SPIRV(SPIRV &&) = delete;
00028             SPIRV &operator=(SPIRV &&) = delete;
00029
00030             [[nodiscard]] std::string getName() const override { return "SPIRV"; }
00031             [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::SHADER_IR; }
00032             [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::ALL; }
00033
00034             [[nodiscard]] ShaderSourceType irType() const override { return ShaderSourceType::SPIRV; }
```

```
00035
00036           ShaderIRModule process(const ShaderIRModule &module) override
00037           {
00038               ShaderIRModule out = module;
00039               return out;
00040           }
00041
00042           void optimize(std::span<ShaderIRModule> modules) override {}
00043
00044           ShaderIRModule crossCompile(const ShaderIRModule &module, const ShaderSourceType targetType) override
00045           {
00046               if (targetType == ShaderSourceType::MSL)
00047               {
00048                   ShaderIRModule out = module;
00049                   return out;
00050               }
00051
00052               throw std::runtime_error("Cross-compilation to this target not implemented");
00053           }
00054
00055       }; // class SPIRV
00056
00057 } // namespace cae
```

43.237 plugins/UI/Imgui/include/Imgui/Imgui.hpp File Reference

43.238 Imgui.hpp

Go to the documentation of this file.

```
00001
```

43.239 plugins/Input/Cocoa/include/Cocoa/Cocoa.hpp File Reference

43.240 Cocoa.hpp

Go to the documentation of this file.

```
00001
```

43.241 plugins/Window/Cocoa/include/Cocoa/Cocoa.hpp File Reference

This file contains the hpp class declaration.

Namespaces

- namespace cae

43.241.1 Detailed Description

This file contains the hpp class declaration.
Definition in file Cocoa.hpp.

43.242 Cocoa.hpp

Go to the documentation of this file.

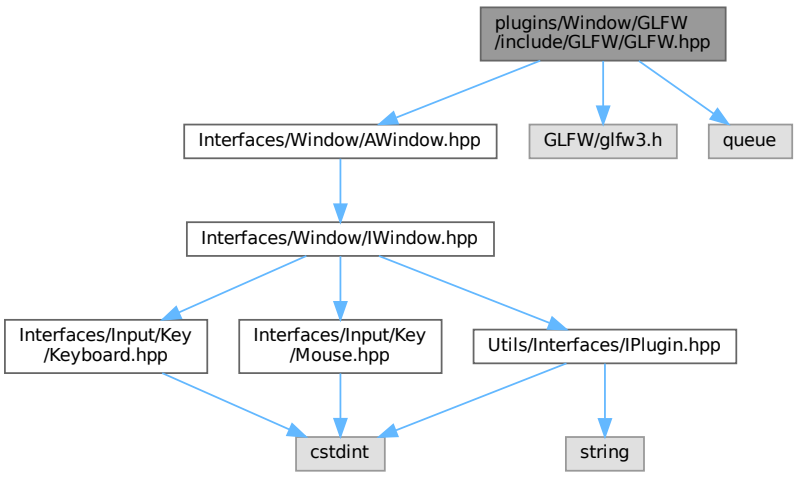
```
00001 ///
00002 /// @file Cocoa.hpp
00003 /// @brief This file contains the hpp class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #ifdef __APPLE__
00010
00011 #include "Interfaces/Window/AWindow.hpp"
00012
00013 #include <queue>
```

```
00014
00015 namespace cae
00016 {
00017
00018     ///
00019     /// @class Cocoa
00020     /// @brief Class for the Cocoa plugin
00021     /// @namespace cae
00022     ///
00023     class Cocoa final : public AWindow
00024     {
00025         public:
00026             Cocoa() = default;
00027             ~Cocoa() override;
00028
00029             Cocoa(const Cocoa &) = delete;
00030             Cocoa &operator=(const Cocoa &) = delete;
00031
00032             [[nodiscard]] std::string getName() const override { return "Cocoa"; }
00033             [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::WINDOW; }
00034             [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::MACOSX; }
00035
00036             bool create(const std::string &name, WindowSize size) override;
00037             void close() override;
00038
00039             [[nodiscard]] NativeWindowHandle getNativeHandle() const override;
00040             [[nodiscard]] WindowSize getWindowSize() const override;
00041
00042             void setIcon(const std::string &path) const override {}
00043
00044             [[nodiscard]] bool shouldClose() const override;
00045             void pollEvents() override;
00046             bool pollEvent(WindowEvent &event) override;
00047
00048             [[nodiscard]] bool wasResized() const override { return m_resized; }
00049             void resetResizedFlag() override { m_resized = false; }
00050
00051         private:
00052             void *m_window = nullptr; // NSWindow*
00053             void *m_view = nullptr;   // NSView*
00054             void *m_app = nullptr;    // NSApplication*
00055
00056             bool m_shouldClose = false;
00057             bool m_resized = false;
00058
00059             std::queue<WindowEvent> m_eventQueue;
00060             WindowSize m_size{};
00061
00062     }; // class Cocoa
00063
00064 } // namespace cae
00065
00066 #endif
```

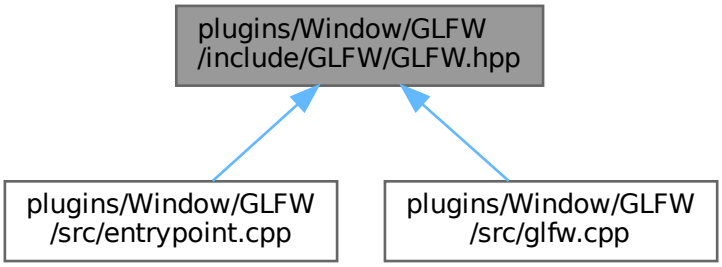
43.243 plugins/Window/GLFW/include/GLFW/GLFW.hpp File Reference

This file contains the GLFW class declaration.
#include "Interfaces/Window/AWindow.hpp"
#include <GLFW/glfw3.h>
#include <queue>

Include dependency graph for GLFW.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class cae::GLFW

  Class for the GLFW plugin.

Namespaces

- namespace cae

43.243.1 Detailed Description

This file contains the GLFW class declaration.
Definition in file GLFW.hpp.

43.244 GLFW.hpp

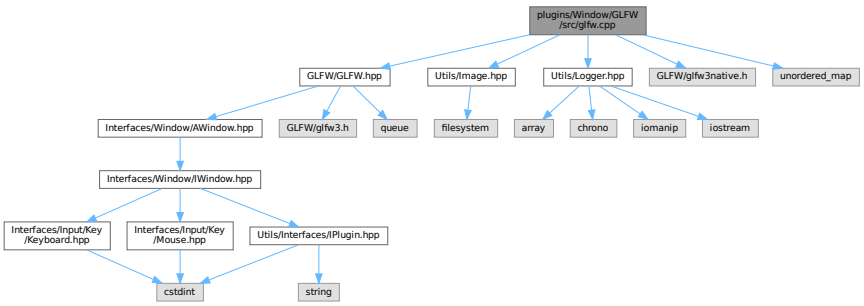
Go to the documentation of this file.

```
00001 ///
00002 /// @file GLFW.hpp
00003 /// @brief This file contains the GLFW class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include "Interfaces/Window/AWindow.hpp"
00010
00011 #ifdef _WIN32
00012 #define GLFW_EXPOSE_NATIVE_WIN32
00013 #elifdef ___linux___
00014 #define GLFW_EXPOSE_NATIVE_X11
00015 #elifdef ___APPLE___
00016 #define GLFW_EXPOSE_NATIVE_COCOA
00017 #endif
00018 #include <GLFW/glfw3.h>
00019
00020 #include <queue>
00021
00022 namespace cae
00023 {
00024
00025     ///
00026     /// @class GLFW
00027     /// @brief Class for the GLFW plugin
00028     /// @namespace cae
00029     ///
00030     class GLFW final : public AWindow
00031     {
00032
00033         public:
00034             GLFW() = default;
00035             ~GLFW() override = default;
00036
00037             GLFW(const GLFW &) = delete;
00038             GLFW &operator=(const GLFW &) = delete;
00039             GLFW(GLFW &&) = delete;
00040             GLFW &operator=(GLFW &&) = delete;
00041
00042             [[nodiscard]] std::string getName() const override { return "GLFW"; }
00043             [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::WINDOW; }
00044             [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::ALL; }
00045
00046             bool create(const std::string &name, WindowSize size) override;
00047             void close() override;
00048
00049             [[nodiscard]] NativeWindowHandle getNativeHandle() const override;
00050             [[nodiscard]] WindowSize getWindowSize() const override;
00051
00052             void setIcon(const std::string &path) const override;
00053
00054             [[nodiscard]] bool shouldClose() const override { return glfwWindowShouldClose(m_window) != 0; }
00055             void pollEvents() override { glfwPollEvents(); }
00056             bool pollEvent(WindowEvent &event) override;
00057
00058             [[nodiscard]] bool wasResized() const override { return m_frameBufferResized; }
00059             void resetResizedFlag() override { m_frameBufferResized = false; }
00060
00061         private:
00062             static void frameBufferResizeCallback(GLFWwindow *window, int width, int height);
00063             static void keyCallback(GLFWwindow *window, int key, int scancode, int action, int mods);
00064             static void mouseButtonCallback(GLFWwindow *window, int button, int action, int mods);
00065             static void cursorPosCallback(GLFWwindow *window, double x, double y);
00066             static void scrollCallback(GLFWwindow *window, double xoffset, double yoffset);
00067
00068             std::queue<WindowEvent> m_eventQueue;
00069
00070             GLFWwindow *m_window = nullptr;
00071             WindowSize m_frameBufferSize{};
00072             bool m_frameBufferResized = false;
00073
00074     }; // class GLFW
00075
00076 } // namespace cae
```

43.245 plugins/Window/GLFW/src/glfw.cpp File Reference

#include "GLFW/GLFW.hpp"
#include "Utils/Image.hpp"
#include "Utils/Logger.hpp"
#include <GLFW/glfw3native.h>

#include <unordered_map>
Include dependency graph for glfw.cpp:



Functions

- static cae::KeyCode translateKey (const int key)

43.245.1 Function Documentation

43.245.1.1 translateKey()

static cae::KeyCode translateKey (
const int key) [static]

Definition at line 10 of file glfw.cpp.
References cae::A, cae::B, cae::Backspace, cae::C, cae::CapsLock, cae::Count, cae::D, cae::Delete, cae::Down, cae::E, cae::End, cae::Enter, cae::Escape, cae::F, cae::F1, cae::F10, cae::F11, cae::F12, cae::F2, cae::F3, cae::F4, cae::F5, cae::F6, cae::F7, cae::F8, cae::F9, cae::G, cae::H, cae::Home, cae::I, cae::Insert, cae::J, cae::K, cae::L, cae::LAlt, cae::LCtrl, cae::Left, cae::LShift, cae::LSuper, cae::M, cae::Menu, cae::N, cae::Num0, cae::Num1, cae::Num2, cae::Num3, cae::Num4, cae::Num5, cae::Num6, cae::Num7, cae::Num8, cae::Num9, cae::O, cae::P, cae::PageDown, cae::PageUp, cae::Pause, cae::PrintScreen, cae::Q, cae::R, cae::RAlt, cae::RCtrl, cae::Right, cae::RShift, cae::RSuper, cae::S, cae::Space, cae::T, cae::Tab, cae::U, cae::Up, cae::V, cae::W, cae::X, cae::Y, and cae::Z.
Referenced by cae::GLFW::keyCallback().
Here is the caller graph for this function:

43.246 glfw.cpp

Go to the documentation of this file.

```
00001 #include "GLFW/GLFW.hpp"
00002
00003 #include "Utils/Image.hpp"
00004 #include "Utils/Logger.hpp"
00005
00006 #include <GLFW/glfw3native.h>
00007
00008 #include <unordered_map>
00009
00010 static cae::KeyCode translateKey(const int key)
```

```
00011 {
00012     switch (key)
00013     {
00014         case GLFW_KEY_A:
00015             return cae::KeyCode::A;
00016         case GLFW_KEY_B:
00017             return cae::KeyCode::B;
00018         case GLFW_KEY_C:
00019             return cae::KeyCode::C;
00020         case GLFW_KEY_D:
00021             return cae::KeyCode::D;
00022         case GLFW_KEY_E:
00023             return cae::KeyCode::E;
00024         case GLFW_KEY_F:
00025             return cae::KeyCode::F;
00026         case GLFW_KEY_G:
00027             return cae::KeyCode::G;
00028         case GLFW_KEY_H:
00029             return cae::KeyCode::H;
00030         case GLFW_KEY_I:
00031             return cae::KeyCode::I;
00032         case GLFW_KEY_J:
00033             return cae::KeyCode::J;
00034         case GLFW_KEY_K:
00035             return cae::KeyCode::K;
00036         case GLFW_KEY_L:
00037             return cae::KeyCode::L;
00038         case GLFW_KEY_M:
00039             return cae::KeyCode::M;
00040         case GLFW_KEY_N:
00041             return cae::KeyCode::N;
00042         case GLFW_KEY_O:
00043             return cae::KeyCode::O;
00044         case GLFW_KEY_P:
00045             return cae::KeyCode::P;
00046         case GLFW_KEY_Q:
00047             return cae::KeyCode::Q;
00048         case GLFW_KEY_R:
00049             return cae::KeyCode::R;
00050         case GLFW_KEY_S:
00051             return cae::KeyCode::S;
00052         case GLFW_KEY_T:
00053             return cae::KeyCode::T;
00054         case GLFW_KEY_U:
00055             return cae::KeyCode::U;
00056         case GLFW_KEY_V:
00057             return cae::KeyCode::V;
00058         case GLFW_KEY_W:
00059             return cae::KeyCode::W;
00060         case GLFW_KEY_X:
00061             return cae::KeyCode::X;
00062         case GLFW_KEY_Y:
00063             return cae::KeyCode::Y;
00064         case GLFW_KEY_Z:
00065             return cae::KeyCode::Z;
00066 
00067         case GLFW_KEY_0:
00068             return cae::KeyCode::Num0;
00069         case GLFW_KEY_1:
00070             return cae::KeyCode::Num1;
00071         case GLFW_KEY_2:
00072             return cae::KeyCode::Num2;
00073         case GLFW_KEY_3:
00074             return cae::KeyCode::Num3;
00075         case GLFW_KEY_4:
00076             return cae::KeyCode::Num4;
00077         case GLFW_KEY_5:
00078             return cae::KeyCode::Num5;
00079         case GLFW_KEY_6:
00080             return cae::KeyCode::Num6;
00081         case GLFW_KEY_7:
00082             return cae::KeyCode::Num7;
00083         case GLFW_KEY_8:
00084             return cae::KeyCode::Num8;
00085         case GLFW_KEY_9:
00086             return cae::KeyCode::Num9;
00087 
00088         case GLFW_KEY_LEFT_SHIFT:
00089             return cae::KeyCode::LShift;
00090         case GLFW_KEY_RIGHT_SHIFT:
00091             return cae::KeyCode::RShift;
00092         case GLFW_KEY_LEFT_CONTROL:
00093             return cae::KeyCode::LCtrl;
00094         case GLFW_KEY_RIGHT_CONTROL:
00095             return cae::KeyCode::RCtrl;
00096         case GLFW_KEY_LEFT_ALT:
00097             return cae::KeyCode::LAlt;
```

```
00098         case GLFW_KEY_RIGHT_ALT:
00099             return cae::KeyCode::RAlt;
00100         case GLFW_KEY_LEFT_SUPER:
00101             return cae::KeyCode::LSuper;
00102         case GLFW_KEY_RIGHT_SUPER:
00103             return cae::KeyCode::RSuper;
00104         case GLFW_KEY_CAPS_LOCK:
00105             return cae::KeyCode::CapsLock;
00106 
00107         case GLFW_KEY_UP:
00108             return cae::KeyCode::Up;
00109         case GLFW_KEY_DOWN:
00110             return cae::KeyCode::Down;
00111         case GLFW_KEY_LEFT:
00112             return cae::KeyCode::Left;
00113         case GLFW_KEY_RIGHT:
00114             return cae::KeyCode::Right;
00115         case GLFW_KEY_HOME:
00116             return cae::KeyCode::Home;
00117         case GLFW_KEY_END:
00118             return cae::KeyCode::End;
00119         case GLFW_KEY_PAGE_UP:
00120             return cae::KeyCode::PageUp;
00121         case GLFW_KEY_PAGE_DOWN:
00122             return cae::KeyCode::PageDown;
00123 
00124         case GLFW_KEY_ENTER:
00125             return cae::KeyCode::Enter;
00126         case GLFW_KEY_BACKSPACE:
00127             return cae::KeyCode::Backspace;
00128         case GLFW_KEY_TAB:
00129             return cae::KeyCode::Tab;
00130         case GLFW_KEY_SPACE:
00131             return cae::KeyCode::Space;
00132         case GLFW_KEY_DELETE:
00133             return cae::KeyCode::Delete;
00134         case GLFW_KEY_INSERT:
00135             return cae::KeyCode::Insert;
00136 
00137         case GLFW_KEY_F1:
00138             return cae::KeyCode::F1;
00139         case GLFW_KEY_F2:
00140             return cae::KeyCode::F2;
00141         case GLFW_KEY_F3:
00142             return cae::KeyCode::F3;
00143         case GLFW_KEY_F4:
00144             return cae::KeyCode::F4;
00145         case GLFW_KEY_F5:
00146             return cae::KeyCode::F5;
00147         case GLFW_KEY_F6:
00148             return cae::KeyCode::F6;
00149         case GLFW_KEY_F7:
00150             return cae::KeyCode::F7;
00151         case GLFW_KEY_F8:
00152             return cae::KeyCode::F8;
00153         case GLFW_KEY_F9:
00154             return cae::KeyCode::F9;
00155         case GLFW_KEY_F10:
00156             return cae::KeyCode::F10;
00157         case GLFW_KEY_F11:
00158             return cae::KeyCode::F11;
00159         case GLFW_KEY_F12:
00160             return cae::KeyCode::F12;
00161 
00162         case GLFW_KEY_ESCAPE:
00163             return cae::KeyCode::Escape;
00164         case GLFW_KEY_PRINT_SCREEN:
00165             return cae::KeyCode::PrintScreen;
00166         case GLFW_KEY_PAUSE:
00167             return cae::KeyCode::Pause;
00168         case GLFW_KEY_MENU:
00169             return cae::KeyCode::Menu;
00170 
00171         default:
00172             return cae::KeyCode::Count;
00173     }
00174 }
00175 
00176 void cae::GLFW::keyCallback(GLFWwindow *window, const int key, int, const int action, int)
00177 {
00178     auto *self = static_cast<GLFW *>(glfwGetWindowUserPointer(window));
00179     if (self == nullptr)
00180     {
00181         return;
00182     }
00183 
00184     WindowEvent e{};
```

```
00185     if (action == GLFW_PRESS)
00186     {
00187         e.type = WindowEventType::KeyDown;
00188     }
00189     else if (action == GLFW_RELEASE)
00190     {
00191         e.type = WindowEventType::KeyUp;
00192     }
00193     else
00194     {
00195         return;
00196     }
00197 
00198     e.key.key = translateKey(key);
00199     self->m_eventQueue.push(e);
00200 }
00201 
00202 void cae::GLFW::mouseButtonCallback(GLFWwindow *window, int button, const int action, int)
00203 {
00204     auto *self = static_cast<GLFW *>(glfwGetWindowUserPointer(window));
00205     if (self == nullptr)
00206     {
00207         return;
00208     }
00209 
00210     WindowEvent e{};
00211     e.type = (action == GLFW_PRESS) ? WindowEventType::MouseButtonDown : WindowEventType::MouseButtonUp;
00212 
00213     e.mouseButton.button = static_cast<MouseButton>(button);
00214     self->m_eventQueue.push(e);
00215 }
00216 
00217 void cae::GLFW::cursorPosCallback(GLFWwindow *window, const double x, const double y)
00218 {
00219     auto *self = static_cast<GLFW *>(glfwGetWindowUserPointer(window));
00220     if (self == nullptr)
00221     {
00222         return;
00223     }
00224 
00225     WindowEvent e{};
00226     e.type = WindowEventType::MouseMove;
00227     e.mouseMove.x = static_cast<int>(x);
00228     e.mouseMove.y = static_cast<int>(y);
00229 
00230     self->m_eventQueue.push(e);
00231 }
00232 
00233 void cae::GLFW::scrollCallback(GLFWwindow *window, const double xoffset, const double yoffset)
00234 {
00235     auto *self = static_cast<GLFW *>(glfwGetWindowUserPointer(window));
00236     if (self == nullptr)
00237     {
00238         return;
00239     }
00240 
00241     WindowEvent e{};
00242     e.type = WindowEventType::MouseScroll;
00243     e.scroll.x = static_cast<float>(xoffset);
00244     e.scroll.y = static_cast<float>(yoffset);
00245 
00246     self->m_eventQueue.push(e);
00247 }
00248 
00249 void cae::GLFW::frameBufferResizeCallback(GLFWwindow *window, const int width, const int height)
00250 {
00251     auto *self = static_cast<GLFW *>(glfwGetWindowUserPointer(window));
00252     if (self == nullptr)
00253     {
00254         return;
00255     }
00256 
00257     self->m_frameBufferResized = true;
00258     self->m_frameBufferSize = {.width = static_cast<uint16_t>(width), .height = static_cast<uint16_t>(height)};
00259 
00260     WindowEvent e{};
00261     e.type = WindowEventType::Resize;
00262     e.resize.w = self->m_frameBufferSize.width;
00263     e.resize.h = self->m_frameBufferSize.height;
00264 
00265     self->m_eventQueue.push(e);
00266 }
00267 
00268 bool cae::GLFW::create(const std::string &name, const WindowSize size)
00269 {
00270     m_window = nullptr;
00271     if (glfwInit() == 0)
```

```
00272     {
00273         utl::Logger::log("Failed to init glfw", utl::LogLevel::WARNING);
00274         return false;
00275     }
00276 #ifdef __APPLE__
00277 #else
00278     glfwWindowHint(GLFW_CLIENT_API, GLFW_NO_API);
00279 #endif
00280
00281     m_window = glfwCreateWindow(size.width, size.height, name.c_str(), nullptr, nullptr);
00282     if (m_window == nullptr)
00283     {
00284         glfwTerminate();
00285         utl::Logger::log("Failed to create GLFW window", utl::LogLevel::WARNING);
00286
00287         return false;
00288     }
00289     glfwSetWindowUserPointer(m_window, this);
00290
00291     glfwSetFramebufferSizeCallback(m_window, frameBufferResizeCallback);
00292     glfwSetKeyCallback(m_window, keyCallback);
00293     glfwSetMouseButtonCallback(m_window, mouseButtonCallback);
00294     glfwSetCursorPosCallback(m_window, cursorPosCallback);
00295     glfwSetScrollCallback(m_window, scrollCallback);
00296 #ifdef __APPLE__
00297     glfwMakeContextCurrent((GLFWwindow *)m_window);
00298 #endif
00299     return true;
00300 }
00301
00302 void cae::GLFW::close()
00303 {
00304     if (m_window != nullptr)
00305     {
00306         glfwDestroyWindow(m_window);
00307         m_window = nullptr;
00308     }
00309     glfwTerminate();
00310 }
00311
00312 cae::WindowSize cae::GLFW::getWindowSize() const
00313 {
00314     int width = 0;
00315     int height = 0;
00316     glfwGetWindowSize(m_window, &width, &height);
00317     return {.width = static_cast<uint16_t>(width), .height = static_cast<uint16_t>(height)};
00318 }
00319
00320 cae::NativeWindowHandle cae::GLFW::getNativeHandle() const
00321 {
00322     NativeWindowHandle handle{};
00323 #ifdef _WIN32
00324     handle.window = glfwGetWin32Window(m_window);
00325     handle.display = GetModuleHandle(nullptr);
00326 #elifdef __linux__
00327     handle.window = reinterpret_cast<void *>(glfwGetX11Window(m_window));
00328     handle.display = glfwGetX11Display();
00329 #elifdef __APPLE__
00330     handle.window = glfwGetCocoaWindow(m_window);
00331     handle.display = nullptr;
00332 #endif
00333     return handle;
00334 }
00335
00336 void cae::GLFW::setIcon(const std::string &path) const
00337 {
00338     static const utl::Image image(path);
00339     if (image.pixels == nullptr)
00340     {
00341         utl::Logger::log("Failed to create icon.", utl::LogLevel::WARNING);
00342         return;
00343     }
00344     static const GLFWimage appIcon{.width = image.width, .height = image.height, .pixels = image.pixels};
00345     glfwSetWindowIcon(m_window, 1, &appIcon);
00346 }
00347
00348 bool cae::GLFW::pollEvent(WindowEvent &event)
00349 {
00350     if (m_eventQueue.empty())
00351     {
00352         return false;
00353     }
00354
00355     event = m_eventQueue.front();
00356     m_eventQueue.pop();
00357     return true;
00358 }
```

43.247 plugins/Input/Win32/include/Win32/Win32.hpp File Reference

43.248 Win32.hpp

Go to the documentation of this file.
00001

43.249 plugins/Window/Win32/include/Win32/Win32.hpp File Reference

This file contains the Win32 class declaration.

Namespaces

- namespace cae

43.249.1 Detailed Description

This file contains the Win32 class declaration.
Definition in file Win32.hpp.

43.250 Win32.hpp

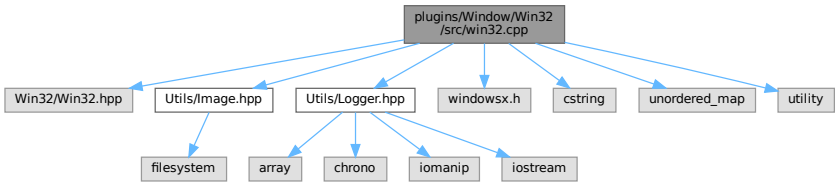
Go to the documentation of this file.

```
00001 ///
00002 /// @file Win32.hpp
00003 /// @brief This file contains the Win32 class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #ifdef _WIN32
00010
00011 #include "Interfaces/Window/AWindow.hpp"
00012
00013 #include <windows.h>
00014
00015 #include <queue>
00016
00017 namespace cae
00018 {
00019
00020     ///
00021     /// @class Win32
00022     /// @brief Class for the Win32 plugin
00023     /// @namespace cae
00024     ///
00025     class Win32 final : public AWindow
00026     {
00027
00028         public:
00029             Win32() = default;
00030             ~Win32() override = default;
00031
00032             Win32(const Win32 &) = delete;
00033             Win32 &operator=(const Win32 &) = delete;
00034             Win32(Win32 &&) = delete;
00035             Win32 &operator=(Win32 &&) = delete;
00036
00037             [[nodiscard]] std::string getName() const override { return "Win32"; }
00038             [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::WINDOW; }
00039             [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::WINDOWS; }
00040
00041             bool create(const std::string &name, WindowSize size) override;
00042             void close() override;
00043
00044             [[nodiscard]] NativeWindowHandle getNativeHandle() const override;
00045             [[nodiscard]] WindowSize getWindowSize() const override;
00046
00047             void setIcon(const std::string &path) const override;
00048
00049             [[nodiscard]] bool shouldClose() const override { return m_shouldClose; }
00050             void pollEvents() override;
```

```
00051             bool pollEvent(WindowEvent &event) override;
00052
00053             bool wasResized() const override { return m_frameBufferResized; }
00054             void resetResizedFlag() override { m_frameBufferResized = false; }
00055
00056         private:
00057             static LRESULT CALLBACK WindowProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);
00058             static KeyCode mapWinKey(WPARAM key);
00059             std::queue<WindowEvent> m_eventQueue;
00060
00061             std::wstring m_title;
00062             HWND m_hwnd = nullptr;
00063             HINSTANCE m_hInstance = nullptr;
00064             WindowSize m_frameBufferSize;
00065             bool m_frameBufferResized = false;
00066             bool m_shouldClose = false;
00067
00068     }; // class Win32
00069
00070 } // namespace cae
00071
00072 #endif
```

43.251 plugins/Window/Win32/src/win32.cpp File Reference

#include ”Win32/Win32.hpp”
#include ”Utils/Image.hpp”
#include ”Utils/Logger.hpp”
#include <windowsx.h>
#include <cstring>
#include <unordered_map>
#include <utility>
Include dependency graph for win32.cpp:



Variables

- constexpr wchar_t WINDOW_CLASS_NAME [] = L”CAE_WindowsWindowClass”

43.251.1 Variable Documentation

43.251.1.1 WINDOW_CLASS_NAME

wchar_t WINDOW_CLASS_NAME[] = L”CAE_WindowsWindowClass” [constexpr]
Definition at line 12 of file win32.cpp.

43.252 win32.cpp

Go to the documentation of this file.

```
00001 #include ”Win32/Win32.hpp”
00002
00003 #include ”Utils/Image.hpp”
00004 #include ”Utils/Logger.hpp”
00005
00006 #include <windowsx.h>
00007
00008 #include <cstring>
00009 #include <unordered_map>
00010 #include <utility>
```

```
00011
00012 constexpr wchar_t WINDOW_CLASS_NAME[] = L"CAE_WindowsWindowClass";
00013
00014 cae::KeyCode cae::Win32::mapWinKey(const WPARAM key)
00015 {
00016     static const std::unordered_map<WPARAM, KeyCode> keyMap = {
00017         {'A', KeyCode::A},
00018         {'B', KeyCode::B},
00019         {'C', KeyCode::C},
00020         {'D', KeyCode::D},
00021         {'E', KeyCode::E},
00022         {'F', KeyCode::F},
00023         {'G', KeyCode::G},
00024         {'H', KeyCode::H},
00025         {'I', KeyCode::I},
00026         {'J', KeyCode::J},
00027         {'K', KeyCode::K},
00028         {'L', KeyCode::L},
00029         {'M', KeyCode::M},
00030         {'N', KeyCode::N},
00031         {'O', KeyCode::O},
00032         {'P', KeyCode::P},
00033         {'Q', KeyCode::Q},
00034         {'R', KeyCode::R},
00035         {'S', KeyCode::S},
00036         {'T', KeyCode::T},
00037         {'U', KeyCode::U},
00038         {'V', KeyCode::V},
00039         {'W', KeyCode::W},
00040         {'X', KeyCode::X},
00041         {'Y', KeyCode::Y},
00042         {'Z', KeyCode::Z},
00043
00044         {'0', KeyCode::Num0},
00045         {'1', KeyCode::Num1},
00046         {'2', KeyCode::Num2},
00047         {'3', KeyCode::Num3},
00048         {'4', KeyCode::Num4},
00049         {'5', KeyCode::Num5},
00050         {'6', KeyCode::Num6},
00051         {'7', KeyCode::Num7},
00052         {'8', KeyCode::Num8},
00053         {'9', KeyCode::Num9},
00054
00055         {VK_ESCAPE, KeyCode::Escape},
00056         {VK_LEFT, KeyCode::Left},
00057         {VK_RIGHT, KeyCode::Right},
00058         {VK_UP, KeyCode::Up},
00059         {VK_DOWN, KeyCode::Down},
00060         {VK_SPACE, KeyCode::Space},
00061         {VK_RETURN, KeyCode::Enter},
00062         {VK_BACK, KeyCode::Backspace},
00063         {VK_TAB, KeyCode::Tab},
00064         {VK_LSHIFT, KeyCode::LShift},
00065         {VK_RSHIFT, KeyCode::RShift},
00066         {VK_LCONTROL, KeyCode::LCtrl},
00067         {VK_RCONTROL, KeyCode::RCtrl},
00068         {VK_LMENU, KeyCode::LAlt},
00069         {VK_RMENU, KeyCode::RAlt}
00070         // ...
00071     };
00072
00073     const auto it = keyMap.find(key);
00074     return it != keyMap.end() ? it->second : KeyCode::Count;
00075 }
00076
00077 LRESULT CALLBACK cae::Win32::WindowProc(const HWND hwnd, const UINT msg, const WPARAM wParam, const
        LPARAM lParam)
00078 {
00079     Win32 *self = nullptr;
00080
00081     if (msg == WM_NCCREATE)
00082     {
00083         const auto *cs = reinterpret_cast<CREATESTRUCTW *>(lParam);
00084         self = static_cast<Win32 *>(cs->lpCreateParams);
00085         SetWindowLongPtrW(hwnd, GWLP_USERDATA, reinterpret_cast<LONG_PTR>(self));
00086         return TRUE;
00087     }
00088
00089     self = reinterpret_cast<Win32 *>(GetWindowLongPtrW(hwnd, GWLP_USERDATA));
00090     if (self == nullptr)
00091     {
00092         return DefWindowProcW(hwnd, msg, wParam, lParam);
00093     }
00094
00095     WindowEvent e{};
00096     switch (msg)
```

```
00097     {
00098         case WM_SIZE:
00099             self->m_frameBufferResized = true;
00100             self->m_frameBufferSize = {.width = LOWORD(lParam), .height = HIWORD(lParam)};
00101             e.type = WindowEventType::Resize;
00102             e.resize = {.w = LOWORD(lParam), .h = HIWORD(lParam)};
00103             self->m_eventQueue.push(e);
00104             return 0;
00105 
00106         case WM_DESTROY:
00107             PostQuitMessage(0);
00108             e.type = WindowEventType::Close;
00109             self->m_eventQueue.push(e);
00110             return 0;
00111 
00112         case WM_KEYDOWN:
00113         case WM_SYSKEYDOWN:
00114             e.type = WindowEventType::KeyDown;
00115             e.key.key = mapWinKey(wParam);
00116             self->m_eventQueue.push(e);
00117             return 0;
00118 
00119         case WM_KEYUP:
00120         case WM_SYSKEYUP:
00121             e.type = WindowEventType::KeyUp;
00122             e.key.key = mapWinKey(wParam);
00123             self->m_eventQueue.push(e);
00124             return 0;
00125 
00126             // mouse, scroll, ...
00127     }
00128 
00129     return DefWindowProcW(hwnd, msg, wParam, lParam);
00130 }
00131 
00132 bool cae::Win32::create(const std::string &name, const WindowSize size)
00133 {
00134     m_hInstance = GetModuleHandleW(nullptr);
00135     m_frameBufferSize = size;
00136     m_shouldClose = false;
00137     m_frameBufferResized = false;
00138 
00139     const int len = MultiByteToWideChar(CP_UTF8, 0, name.c_str(), -1, nullptr, 0);
00140 
00141     m_title.resize(len);
00142 
00143     MultiByteToWideChar(CP_UTF8, 0, name.c_str(), -1, m_title.data(), len);
00144 
00145     if (!m_title.empty() && m_title.back() == L'\0')
00146     {
00147         m_title.pop_back();
00148     }
00149 
00150     static bool classRegistered = false;
00151     if (!classRegistered)
00152     {
00153         WNDCLASSW wc{};
00154         wc.lpfnWndProc = WindowProc;
00155         wc.hInstance = m_hInstance;
00156         wc.lpszClassName = WINDOW_CLASS_NAME;
00157         wc.hCursor = LoadCursorW(nullptr, IDC_ARROW);
00158         wc.hIcon = LoadIconW(nullptr, IDI_APPLICATION);
00159         wc.hbrBackground = reinterpret_cast<HBRUSH>(COLOR_WINDOW + 1);
00160 
00161         if (RegisterClassW(&wc) == 0U)
00162         {
00163             utl::Logger::log("Failed to register Win32 window class", utl::LogLevel::WARNING);
00164             return false;
00165         }
00166         classRegistered = true;
00167     }
00168     m_hwnd = CreateWindowExW(0, WINDOW_CLASS_NAME, L"", WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT, size.width,
00169                              size.height, nullptr, nullptr, m_hInstance, this);
00170 
00171     if (m_hwnd == nullptr)
00172     {
00173         return false;
00174     }
00175 
00176     SetWindowTextW(m_hwnd, m_title.c_str());
00177 
00178     ShowWindow(m_hwnd, SW_SHOW);
00179     UpdateWindow(m_hwnd);
00180     return true;
00181 }
00182 
```

```
00183 void cae::Win32::close()
00184 {
00185     if (m_hwnd != nullptr)
00186     {
00187         DestroyWindow(m_hwnd);
00188         m_hwnd = nullptr;
00189     }
00190     UnregisterClassW(WINDOW_CLASS_NAME, m_hInstance);
00191 }
00192
00193 cae::NativeWindowHandle cae::Win32::getNativeHandle() const { return {.window = m_hwnd, .display = m_hInstance}; }
00194
00195 cae::WindowSize cae::Win32::getWindowSize() const
00196 {
00197     RECT rect{};
00198     GetClientRect(m_hwnd, &rect);
00199     return {.width = static_cast<uint16_t>(rect.right - rect.left),
00200             .height = static_cast<uint16_t>(rect.bottom - rect.top)};
00201 }
00202
00203 void cae::Win32::setIcon(const std::string &path) const
00204 {
00205     try
00206     {
00207         const utl::Image image(path);
00208
00209         for (size_t i = 0; std::cmp_less(i, image.width * image.height); ++i)
00210         {
00211             std::swap(image.pixels[(i * 4) + 0], image.pixels[(i * 4) + 2]);
00212         }
00213
00214         ICONINFO iconInfo{};
00215         iconInfo.fIcon = TRUE;
00216
00217         BITMAPV5HEADER bi{};
00218         bi.bV5Size = sizeof(BITMAPV5HEADER);
00219         bi.bV5Width = image.width;
00220         bi.bV5Height = -static_cast<LONG>(image.height);
00221         bi.bV5Planes = 1;
00222         bi.bV5BitCount = 32;
00223         bi.bV5Compression = BI_BITFIELDS;
00224         bi.bV5RedMask = 0x00FF0000;
00225         bi.bV5GreenMask = 0x0000FF00;
00226         bi.bV5BlueMask = 0x000000FF;
00227         bi.bV5AlphaMask = 0xFF000000;
00228
00229         void *pBits = nullptr;
00230         const HDC hdc = GetDC(nullptr);
00231         const HBITMAP hBitmap =
00232             CreateDIBSection(hdc, reinterpret_cast<BITMAPINFO *>(&bi), DIB_RGB_COLORS, &pBits, nullptr, 0);
00233         ReleaseDC(nullptr, hdc);
00234
00235         if (hBitmap == nullptr)
00236         {
00237             utl::Logger::log("Failed to create window icon.", utl::LogLevel::WARNING);
00238             return;
00239         }
00240
00241         std::memcpy(pBits, image.pixels, static_cast<size_t>(image.width * image.height * 4));
00242
00243         iconInfo.hbmColor = hBitmap;
00244         iconInfo.hbmMask = CreateBitmap(image.width, image.height, 1, 1, nullptr);
00245
00246         HICON hIcon = CreateIconIndirect(&iconInfo);
00247
00248         DeleteObject(hBitmap);
00249         DeleteObject(iconInfo.hbmMask);
00250
00251         if (hIcon == nullptr)
00252         {
00253             utl::Logger::log("Failed to create window icon", utl::LogLevel::WARNING);
00254             return;
00255         }
00256
00257         SendMessageW(m_hwnd, WM_SETICON, ICON_BIG, reinterpret_cast<LPARAM>(hIcon));
00258         SendMessageW(m_hwnd, WM_SETICON, ICON_SMALL, reinterpret_cast<LPARAM>(hIcon));
00259     }
00260     catch (const std::exception &e)
00261     {
00262         utl::Logger::log("Failed to load icon: " + std::string(e.what()), utl::LogLevel::WARNING);
00263     }
00264 }
00265
00266 void cae::Win32::pollEvents()
00267 {
00268     MSG msg{};
00269     while (PeekMessage(&msg, nullptr, 0, 0, PM_REMOVE))
```

```
00270     {
00271         if (msg.message == WM_QUIT)
00272         {
00273             m_shouldClose = true;
00274         }
00275         TranslateMessage(&msg);
00276         DispatchMessage(&msg);
00277     }
00278 }
00279
00280 bool cae::Win32::pollEvent(WindowEvent &event)
00281 {
00282     MSG msg{};
00283     while (PeekMessage(&msg, nullptr, 0, 0, PM_REMOVE))
00284     {
00285         TranslateMessage(&msg);
00286         DispatchMessage(&msg);
00287     }
00288
00289     if (!m_eventQueue.empty())
00290     {
00291         event = m_eventQueue.front();
00292         m_eventQueue.pop();
00293         return true;
00294     }
00295
00296     return false;
00297 }
```

43.253 plugins/Input/X11/include/X11/X11.hpp File Reference

43.254 X11.hpp

Go to the documentation of this file.

```
00001
```

43.255 plugins/Window/X11/include/X11/X11.hpp File Reference

This file contains the X11 class declaration.

Namespaces

- namespace cae

43.255.1 Detailed Description

This file contains the X11 class declaration.
Definition in file X11.hpp.

43.256 X11.hpp

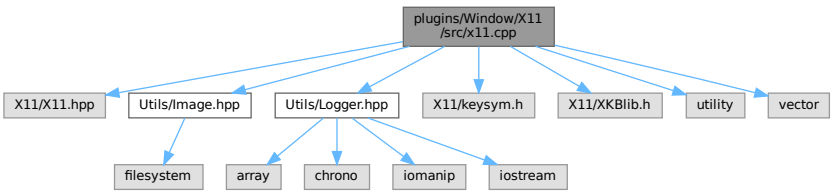
Go to the documentation of this file.

```
00001 ///
00002 /// @file X11.hpp
00003 /// @brief This file contains the X11 class declaration
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #ifdef __linux__
00010
00011 #include "Interfaces/Window/AWindow.hpp"
00012
00013 #include <X11/Xlib.h>
00014
00015 #include <queue>
00016
00017 namespace cae
00018 {
00019
00020     ///
00021     /// @class X11
```

```
00022     /// @brief Class for the X11 plugin
00023     /// @namespace cae
00024     ///
00025     class X11 final : public AWindow
00026     {
00027
00028         public:
00029             X11() = default;
00030             ~X11() override = default;
00031
00032             X11(const X11 &) = delete;
00033             X11 &operator=(const X11 &) = delete;
00034             X11(X11 &&) = delete;
00035             X11 &operator=(X11 &&) = delete;
00036
00037             [[nodiscard]] std::string getName() const override { return "X11"; }
00038             [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::WINDOW; }
00039             [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::LINUX; }
00040
00041             bool create(const std::string &name, WindowSize size) override;
00042             void close() override;
00043
00044             [[nodiscard]] NativeWindowHandle getNativeHandle() const override
00045             {
00046                 return {.window = reinterpret_cast<void *>(m_window), .display = reinterpret_cast<void *>(m_display)};
00047             }
00048             [[nodiscard]] WindowSize getWindowSize() const override;
00049
00050             void setIcon(const std::string &path) const override;
00051
00052             [[nodiscard]] bool shouldClose() const override;
00053             void pollEvents() override;
00054             bool pollEvent(WindowEvent &outEvent) override;
00055
00056             bool wasResized() const override { return m_frameBufferResized; }
00057             void resetResizedFlag() override { m_frameBufferResized = false; }
00058
00059         private:
00060             std::queue<WindowEvent> m_eventQueue;
00061             Display *m_display = nullptr;
00062             Window m_window = 0;
00063             WindowSize m_frameBufferSize;
00064             mutable bool m_frameBufferResized = false;
00065             Atom m_wmDeleteMessage = 0;
00066             bool m_shouldClose = false;
00067
00068     }; // class X11
00069
00070 } // namespace cae
00071
00072 #endif
```

43.257 plugins/Window/X11/src/x11.cpp File Reference

#include "X11/X11.hpp"
#include "Utils/Image.hpp"
#include "Utils/Logger.hpp"
#include <X11/keysym.h>
#include <X11/XKBlib.h>
#include <utility>
#include <vector>
Include dependency graph for x11.cpp:



Functions

- static cae::KeyCode translateKeysym (const KeySym sym)

43.257.1 Function Documentation

43.257.1.1 translateKeysym()

static cae::KeyCode translateKeysym (
const KeySym sym) [static]

Definition at line 12 of file x11.cpp.

References cae::A, cae::B, cae::Backspace, cae::C, cae::CapsLock, cae::Count, cae::D, cae::Delete, cae::Down, cae::E, cae::End, cae::Enter, cae::Escape, cae::F, cae::F1, cae::F10, cae::F11, cae::F12, cae::F2, cae::F3, cae::F4, cae::F5, cae::F6, cae::F7, cae::F8, cae::F9, cae::G, cae::H, cae::Home, cae::I, cae::Insert, cae::J, cae::K, cae::L, cae::LAlt, cae::LCtrl, cae::Left, cae::LShift, cae::LSuper, cae::M, cae::Menu, cae::N, cae::Num0, cae::Num1, cae::Num2, cae::Num3, cae::Num4, cae::Num5, cae::Num6, cae::Num7, cae::Num8, cae::Num9, cae::O, cae::P, cae::PageDown, cae::PageUp, cae::Pause, cae::PrintScreen, cae::Q, cae::R, cae::RAlt, cae::RCtrl, cae::Right, cae::RShift, cae::RSuper, cae::S, cae::Space, cae::T, cae::Tab, cae::U, cae::Up, cae::V, cae::W, cae::X, cae::Y, and cae::Z.

43.258 x11.cpp

Go to the documentation of this file.

```
00001 #include "X11/X11.hpp"
00002
00003 #include "Utils/Image.hpp"
00004 #include "Utils/Logger.hpp"
00005
00006 #include <X11/keysym.h>
00007
00008 #include <X11/XKBlib.h>
00009 #include <utility>
00010 #include <vector>
00011
00012 static cae::KeyCode translateKeysym(const KeySym sym)
00013 {
00014     switch (sym)
00015     {
00016         case XK_a:
00017             return cae::KeyCode::A;
00018         case XK_b:
00019             return cae::KeyCode::B;
00020         case XK_c:
00021             return cae::KeyCode::C;
00022         case XK_d:
00023             return cae::KeyCode::D;
00024         case XK_e:
00025             return cae::KeyCode::E;
00026         case XK_f:
00027             return cae::KeyCode::F;
00028         case XK_g:
00029             return cae::KeyCode::G;
00030         case XK_h:
00031             return cae::KeyCode::H;
00032         case XK_i:
00033             return cae::KeyCode::I;
00034         case XK_j:
00035             return cae::KeyCode::J;
00036         case XK_k:
00037             return cae::KeyCode::K;
00038         case XK_l:
00039             return cae::KeyCode::L;
00040         case XK_m:
00041             return cae::KeyCode::M;
00042         case XK_n:
00043             return cae::KeyCode::N;
00044         case XK_o:
00045             return cae::KeyCode::O;
00046         case XK_p:
00047             return cae::KeyCode::P;
00048         case XK_q:
00049             return cae::KeyCode::Q;
00050         case XK_r:
00051             return cae::KeyCode::R;
00052         case XK_s:
00053             return cae::KeyCode::S;
00054         case XK_t:
```

```
00055             return cae::KeyCode::T;
00056         case XK_u:
00057             return cae::KeyCode::U;
00058         case XK_v:
00059             return cae::KeyCode::V;
00060         case XK_w:
00061             return cae::KeyCode::W;
00062         case XK_x:
00063             return cae::KeyCode::X;
00064         case XK_y:
00065             return cae::KeyCode::Y;
00066         case XK_z:
00067             return cae::KeyCode::Z;
00068 
00069         // Numbers
00070         case XK_0:
00071             return cae::KeyCode::Num0;
00072         case XK_1:
00073             return cae::KeyCode::Num1;
00074         case XK_2:
00075             return cae::KeyCode::Num2;
00076         case XK_3:
00077             return cae::KeyCode::Num3;
00078         case XK_4:
00079             return cae::KeyCode::Num4;
00080         case XK_5:
00081             return cae::KeyCode::Num5;
00082         case XK_6:
00083             return cae::KeyCode::Num6;
00084         case XK_7:
00085             return cae::KeyCode::Num7;
00086         case XK_8:
00087             return cae::KeyCode::Num8;
00088         case XK_9:
00089             return cae::KeyCode::Num9;
00090 
00091         // Modifiers
00092         case XK_Shift_L:
00093             return cae::KeyCode::LShift;
00094         case XK_Shift_R:
00095             return cae::KeyCode::RShift;
00096         case XK_Control_L:
00097             return cae::KeyCode::LCtrl;
00098         case XK_Control_R:
00099             return cae::KeyCode::RCtrl;
00100         case XK_Alt_L:
00101             return cae::KeyCode::LAlt;
00102         case XK_Alt_R:
00103             return cae::KeyCode::RAlt;
00104         case XK_Super_L:
00105             return cae::KeyCode::LSuper;
00106         case XK_Super_R:
00107             return cae::KeyCode::RSuper;
00108         case XK_Caps_Lock:
00109             return cae::KeyCode::CapsLock;
00110 
00111         // Navigation
00112         case XK_Up:
00113             return cae::KeyCode::Up;
00114         case XK_Down:
00115             return cae::KeyCode::Down;
00116         case XK_Left:
00117             return cae::KeyCode::Left;
00118         case XK_Right:
00119             return cae::KeyCode::Right;
00120         case XK_Home:
00121             return cae::KeyCode::Home;
00122         case XK_End:
00123             return cae::KeyCode::End;
00124         case XK_Page_Up:
00125             return cae::KeyCode::PageUp;
00126         case XK_Page_Down:
00127             return cae::KeyCode::PageDown;
00128 
00129         // Editing
00130         case XK_Return:
00131             return cae::KeyCode::Enter;
00132         case XK_BackSpace:
00133             return cae::KeyCode::Backspace;
00134         case XK_Tab:
00135             return cae::KeyCode::Tab;
00136         case XK_space:
00137             return cae::KeyCode::Space;
00138         case XK_Delete:
00139             return cae::KeyCode::Delete;
00140         case XK_Insert:
00141             return cae::KeyCode::Insert;
```

```
00142
00143         // Function keys
00144         case XK_F1:
00145             return cae::KeyCode::F1;
00146         case XK_F2:
00147             return cae::KeyCode::F2;
00148         case XK_F3:
00149             return cae::KeyCode::F3;
00150         case XK_F4:
00151             return cae::KeyCode::F4;
00152         case XK_F5:
00153             return cae::KeyCode::F5;
00154         case XK_F6:
00155             return cae::KeyCode::F6;
00156         case XK_F7:
00157             return cae::KeyCode::F7;
00158         case XK_F8:
00159             return cae::KeyCode::F8;
00160         case XK_F9:
00161             return cae::KeyCode::F9;
00162         case XK_F10:
00163             return cae::KeyCode::F10;
00164         case XK_F11:
00165             return cae::KeyCode::F11;
00166         case XK_F12:
00167             return cae::KeyCode::F12;
00168
00169         // System
00170         case XK_Escape:
00171             return cae::KeyCode::Escape;
00172         case XK_Print:
00173             return cae::KeyCode::PrintScreen;
00174         case XK_Pause:
00175             return cae::KeyCode::Pause;
00176         case XK_Menu:
00177             return cae::KeyCode::Menu;
00178
00179         default:
00180             return cae::KeyCode::Count;
00181     }
00182 }
00183
00184 bool cae::X11::create(const std::string &name, const WindowSize size)
00185 {
00186     m_display = XOpenDisplay(nullptr);
00187     if (m_display == nullptr)
00188     {
00189         utl::Logger::log("Failed to open X display", utl::LogLevel::WARNING);
00190         return false;
00191     }
00192
00193     const int screen = DefaultScreen(m_display);
00194     const Window root = RootWindow(m_display, screen);
00195
00196     m_window = XCreateSimpleWindow(m_display, root, 0, 0, size.width, size.height, 1, BlackPixel(m_display, screen),
00197                                    WhitePixel(m_display, screen));
00198
00199     if (m_window == 0U)
00200     {
00201         utl::Logger::log("Failed to create X11 window", utl::LogLevel::WARNING);
00202         return false;
00203     }
00204
00205     XStoreName(m_display, m_window, name.c_str());
00206
00207     XSelectInput(m_display, m_window,
00208                  ExposureMask | KeyPressMask | KeyReleaseMask | StructureNotifyMask | PointerMotionMask |
00209                      ButtonPressMask | ButtonReleaseMask);
00210     m_wmDeleteMessage = XInternAtom(m_display, "WM_DELETE_WINDOW", False);
00211     XSetWMProtocols(m_display, m_window, &m_wmDeleteMessage, 1);
00212
00213     XMapWindow(m_display, m_window);
00214     XFlush(m_display);
00215
00216     m_frameBufferSize = size;
00217
00218     return true;
00219 }
00220
00221 void cae::X11::close()
00222 {
00223     if (m_display != nullptr && m_window != 0U)
00224     {
00225         XDestroyWindow(m_display, m_window);
00226         XCloseDisplay(m_display);
00227         m_display = nullptr;
00228         m_window = 0;
```

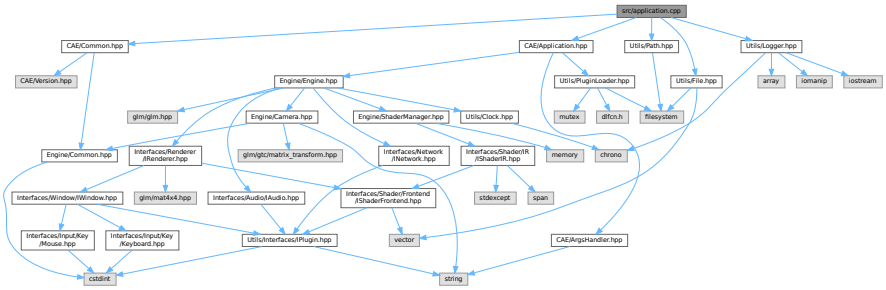
```
00229     }
00230 }
00231
00232 cae::WindowSize cae::X11::getWindowSize() const
00233 {
00234     if (m_display == nullptr || m_window == 0U)
00235     {
00236         return m_frameBufferSize;
00237     }
00238
00239     XWindowAttributes attrs;
00240     XGetWindowAttributes(m_display, m_window, &attrs);
00241     return {.width = static_cast<uint16_t>(attrs.width), .height = static_cast<uint16_t>(attrs.height)};
00242 }
00243
00244 void cae::X11::setIcon(const std::string &path) const
00245 {
00246     if ((m_display == nullptr) || m_window == 0)
00247     {
00248         utl::Logger::log("Failed to create window icon", utl::LogLevel::WARNING);
00249         return;
00250     }
00251
00252     try
00253     {
00254         const utl::Image image(path);
00255
00256         const auto pixelCount = image.width * image.height;
00257         std::vector<unsigned long> iconData;
00258         iconData.reserve(2 + pixelCount);
00259
00260         iconData.push_back(static_cast<unsigned long>(image.width));
00261         iconData.push_back(static_cast<unsigned long>(image.height));
00262
00263         const uint8_t *pixels = image.pixels;
00264         for (size_t i = 0; std::cmp_less(i, pixelCount); ++i)
00265         {
00266             const uint8_t r = pixels[(i * 4) + 0];
00267             const uint8_t g = pixels[(i * 4) + 1];
00268             const uint8_t b = pixels[(i * 4) + 2];
00269             const uint8_t a = pixels[(i * 4) + 3];
00270
00271             const unsigned long argb = (static_cast<unsigned long>(a) « 24) | (static_cast<unsigned long>(r) « 16) |
00272                                        (static_cast<unsigned long>(g) « 8) | (static_cast<unsigned long>(b));
00273
00274             iconData.push_back(argb);
00275         }
00276
00277         const Atom netWmIcon = XInternAtom(m_display, "_NET_WM_ICON", False);
00278         const Atom cardinal = XInternAtom(m_display, "CARDINAL", False);
00279
00280         XChangeProperty(m_display, m_window, netWmIcon, cardinal, 32, PropModeReplace,
00281                         reinterpret_cast<const unsigned char *>(iconData.data()), static_cast<int>(iconData.size()));
00282
00283         XFlush(m_display);
00284     }
00285     catch (const std::exception &e)
00286     {
00287         utl::Logger::log(std::string("Failed to set X11 window icon: ") + e.what(), utl::LogLevel::WARNING);
00288     }
00289 }
00290
00291 bool cae::X11::shouldClose() const { return m_shouldClose; }
00292
00293 void cae::X11::pollEvents() {}
00294
00295 bool cae::X11::pollEvent(WindowEvent &outEvent)
00296 {
00297     if (m_eventQueue.empty() && XPending(m_display) == 0)
00298     {
00299         return false;
00300     }
00301
00302     while (XPending(m_display) > 0)
00303     {
00304         XEvent event;
00305         XNextEvent(m_display, &event);
00306         WindowEvent e{};
00307
00308         switch (event.type)
00309         {
00310             case KeyPress:
00311             {
00312                 e.type = WindowEventType::KeyDown;
00313
00314                 const KeySym sym = XkbKeycodeToKeysym(m_display, event.xkey.keycode, 0, 0);
00315
```

```
00316                 e.key.key = translateKeysym(sym);
00317                 m_eventQueue.push(e);
00318                 break;
00319             }
00320
00321             case KeyRelease:
00322             {
00323                 e.type = WindowEventType::KeyUp;
00324
00325                 const KeySym sym = XkbKeycodeToKeysym(m_display, event.xkey.keycode, 0, 0);
00326
00327                 e.key.key = translateKeysym(sym);
00328                 m_eventQueue.push(e);
00329                 break;
00330             }
00331
00332             case ConfigureNotify:
00333                 m_frameBufferResized = true;
00334                 m_frameBufferSize.width = event.xconfigure.width;
00335                 m_frameBufferSize.height = event.xconfigure.height;
00336                 e.type = WindowEventType::Resize;
00337                 e.resize.w = event.xconfigure.width;
00338                 e.resize.h = event.xconfigure.height;
00339                 m_eventQueue.push(e);
00340                 break;
00341
00342             case ClientMessage:
00343                 if (std::cmp_equal(event.xclient.data.l[0], m_wmDeleteMessage))
00344                 {
00345                     m_shouldClose = true;
00346                     e.type = WindowEventType::Close;
00347                     m_eventQueue.push(e);
00348                 }
00349                 break;
00350
00351             case MotionNotify:
00352                 e.type = WindowEventType::MouseMove;
00353                 e.mouseMove.x = event.xmotion.x;
00354                 e.mouseMove.y = event.xmotion.y;
00355                 m_eventQueue.push(e);
00356                 break;
00357
00358             case ButtonPress:
00359             case ButtonRelease:
00360                 e.type =
00361                     (event.type == ButtonPress) ? WindowEventType::MouseButtonDown :
    WindowEventType::MouseButtonUp;
00362                 e.mouseButton.button = static_cast<MouseButton>(event.xbutton.button);
00363                 m_eventQueue.push(e);
00364                 break;
00365
00366             case Expose:
00367                 break;
00368
00369             default:
00370                 break;
00371         }
00372     }
00373
00374     if (!m_eventQueue.empty())
00375     {
00376         outEvent = m_eventQueue.front();
00377         m_eventQueue.pop();
00378         return true;
00379     }
00380
00381     return false;
00382 }
```

43.259 src/application.cpp File Reference

```
#include "CAE/Application.hpp"
#include "CAE/Common.hpp"
#include "Utils/File.hpp"
#include "Utils/Logger.hpp"
#include "Utils/Path.hpp"
```

Include dependency graph for application.cpp:



Functions

- static std::vector< std::shared_ptr< utl::IPlugin > > loadPlugins (const std::unique_ptr< utl::PluginLoader > &loader)

Variables

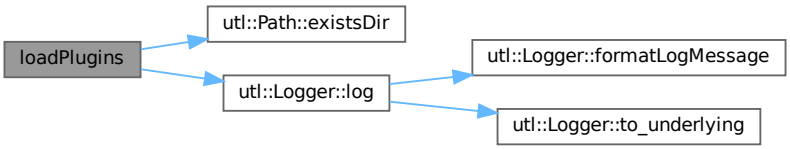
- static const std::vector< float > cubeVertices

43.259.1 Function Documentation

43.259.1.1 loadPlugins()

static std::vector< std::shared_ptr< utl::IPlugin > > loadPlugins (
const std::unique_ptr< utl::PluginLoader > & loader) [static]

Definition at line 8 of file application.cpp.
References utl::Path::existsDir(), utl::INFO, utl::Logger::log(), and utl::WARNING.
Referenced by cae::Application::setupEngine().
Here is the call graph for this function:



Here is the caller graph for this function:

43.259.2 Variable Documentation

43.259.2.1 cubeVertices

const std::vector<float> cubeVertices [static]

Initial value:
= {

```
    -0.5f, -0.5f, -0.5f, 1, 0, 0, 0.5f,  -0.5f, -0.5f, 0, 1, 0, 0.5f,  0.5f,  -0.5f, 0, 0, 1,
    0.5f,  0.5f,  -0.5f, 0, 0, 1, -0.5f, 0.5f,  -0.5f, 1, 1, 0, -0.5f, -0.5f, -0.5f, 1, 0, 0,

    -0.5f, -0.5f, 0.5f,  1, 0, 1, 0.5f,  -0.5f, 0.5f,  0, 1, 1, 0.5f,  0.5f,  0.5f,  1, 1, 1,
    0.5f,  0.5f,  0.5f,  1, 1, 1, -0.5f, 0.5f,  0.5f,  0, 0, 0, -0.5f, -0.5f, 0.5f,  1, 0, 1,

    -0.5f, 0.5f,  0.5f,  1, 0, 0, -0.5f, 0.5f,  -0.5f, 0, 1, 0, -0.5f, -0.5f, -0.5f, 0, 0, 1,
    -0.5f, -0.5f, -0.5f, 0, 0, 1, -0.5f, -0.5f, 0.5f,  1, 1, 0, -0.5f, 0.5f,  0.5f,  1, 0, 0,

    0.5f,  0.5f,  0.5f,  1, 0, 1, 0.5f,  0.5f,  -0.5f, 0, 1, 1, 0.5f,  -0.5f, -0.5f, 1, 1, 1,
    0.5f,  -0.5f, -0.5f, 1, 1, 1, 0.5f,  -0.5f, 0.5f,  0, 0, 0, 0.5f,  0.5f,  0.5f,  1, 0, 1,

    -0.5f, -0.5f, -0.5f, 1, 0, 0, 0.5f,  -0.5f, -0.5f, 0, 1, 0, 0.5f,  -0.5f, 0.5f,  0, 0, 1,
    0.5f,  -0.5f, 0.5f,  0, 0, 1, -0.5f, -0.5f, 0.5f,  1, 1, 0, -0.5f, -0.5f, -0.5f, 1, 0, 0,

    -0.5f, 0.5f,  -0.5f, 1, 0, 1, 0.5f,  0.5f,  -0.5f, 0, 1, 1, 0.5f,  0.5f,  0.5f,  1, 1, 1,
    0.5f,  0.5f,  0.5f,  1, 1, 1, -0.5f, 0.5f,  0.5f,  0, 0, 0, -0.5f, 0.5f,  -0.5f, 1, 0, 1}
```

Definition at line 122 of file application.cpp.
Referenced by cae::Application::start().

43.260 application.cpp

Go to the documentation of this file.

```
00001 #include "CAE/Application.hpp"
00002 #include "CAE/Common.hpp"
00003
00004 #include "Utils/File.hpp"
00005 #include "Utils/Logger.hpp"
00006 #include "Utils/Path.hpp"
00007
00008 static std::vector<std::shared_ptr<utl::IPlugin» loadPlugins(const std::unique_ptr<utl::PluginLoader> &loader)
00009 {
00010     std::vector<std::shared_ptr<utl::IPlugin» loadedPlugins;
00011     const std::filesystem::path pluginDir{PLUGINS_DIR};
00012     if (!utl::Path::existsDir(pluginDir))
00013     {
00014         utl::Logger::log("Plugins directory does not exist: " + pluginDir.string(), utl::LogLevel::WARNING);
00015         return loadedPlugins;
00016     }
00017
00018     for (const auto &entry : std::filesystem::directory_iterator(pluginDir))
00019     {
00020         if (!entry.is_regular_file() || entry.path().extension() != PLUGINS_EXTENSION)
00021         {
00022             continue;
00023         }
00024         const std::string pluginPath = entry.path().string();
00025         if (auto plugin = loader->loadPlugin<utl::IPlugin>(pluginPath, "cae-"); plugin != nullptr)
00026         {
00027             utl::Logger::log("Loaded plugin:\t " + plugin->getName() + " from " + pluginPath, utl::LogLevel::INFO);
00028             loadedPlugins.push_back(plugin);
00029         }
00030         else
00031         {
00032             utl::Logger::log("Failed to load plugin: " + pluginPath, utl::LogLevel::WARNING);
00033         }
00034     }
00035     if (loadedPlugins.empty())
00036     {
00037         utl::Logger::log("No plugins loaded from directory: " + pluginDir.string(), utl::LogLevel::WARNING);
00038     }
00039
00040     return loadedPlugins;
00041 }
00042
00043 cae::Application::Application(const ArgsConfig &argsConfig, const EnvConfig &envConfig)
00044     : m_pluginLoader(std::make_unique<utl::PluginLoader>())
00045 {
00046     utl::Logger::log("PROJECT INFO:\n" + std::string(MESSAGE::VERSION_MSG) + '\n', utl::LogLevel::INFO);
00047
00048     try
00049     {
00050         m_appConfig.envConfig = envConfig;
00051
00052         if (!argsConfig.config_path.empty())
00053         {
00054             m_appConfig.engineConfig = parseEngineConf(argsConfig.config_path);
00055         }
00056         setupEngine(PLUGINS::NAME::RENDERER::OPENGL, PLUGINS::NAME::WINDOW::GLFW,
        PLUGINS::NAME::SHADER::FRONTEND::GLSL,
```

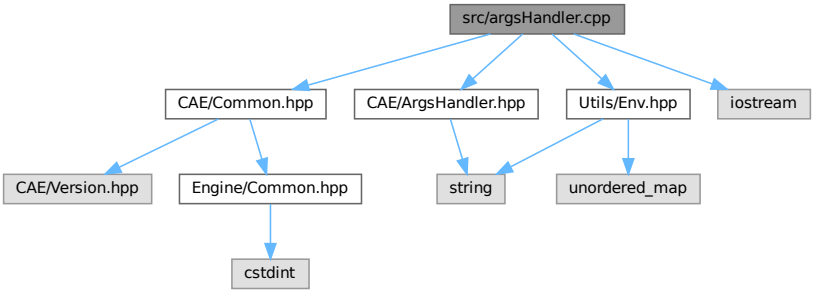
```
00057                   PLUGINS::NAME::SHADER::IR::SPIRV);
00058     }
00059     catch (const std::exception &e)
00060     {
00061         utl::Logger::log(std::string("Application initialization failed: ") + e.what(), utl::LogLevel::WARNING);
00062     }
00063 }
00064
00065 void cae::Application::setupEngine(const std::string &rendererName, const std::string &windowName,
00066                                    const std::string &shaderFrontendName, const std::string &shaderIRName)
00067 {
00068     std::shared_ptr<IWindow> windowPlugin = nullptr;
00069     std::shared_ptr<IRenderer> rendererPlugin = nullptr;
00070     std::shared_ptr<IShaderIR> shaderIRPlugin = nullptr;
00071     std::vector<std::function<std::shared_ptr<IShaderFrontend>()» shaderFactories;
00072
00073     for (auto &plugin : loadPlugins(m_pluginLoader))
00074     {
00075         if (const auto renderer = std::dynamic_pointer_cast<IRenderer>(plugin))
00076         {
00077             if (renderer->getName() == rendererName)
00078             {
00079                 rendererPlugin = renderer;
00080             }
00081         }
00082         if (const auto window = std::dynamic_pointer_cast<IWindow>(plugin))
00083         {
00084             if (window->getName() == windowName)
00085             {
00086                 windowPlugin = window;
00087             }
00088         }
00089         if (const auto shader = std::dynamic_pointer_cast<IShaderFrontend>(plugin))
00090         {
00091             if (shader->getName() == shaderFrontendName)
00092             {
00093                 shaderFactories.emplace_back([shader]() { return shader; });
00094             }
00095         }
00096         if (const auto shaderIR = std::dynamic_pointer_cast<IShaderIR>(plugin))
00097         {
00098             if (shaderIR->getName() == shaderIRName)
00099             {
00100                 shaderIRPlugin = shaderIR;
00101             }
00102         }
00103     }
00104     if (windowPlugin == nullptr)
00105     {
00106         utl::Logger::log("No window plugin found with name: " + windowName, utl::LogLevel::WARNING);
00107     }
00108     if (rendererPlugin == nullptr)
00109     {
00110         utl::Logger::log("No renderer plugin found with name: " + rendererName, utl::LogLevel::WARNING);
00111     }
00112     if (shaderFactories.empty())
00113     {
00114         utl::Logger::log("No shader plugin found with name: " + shaderFrontendName, utl::LogLevel::WARNING);
00115     }
00116     m_engine = std::make_unique<Engine>(
00117         m_appConfig.engineConfig, []() { return nullptr; }, []() { return nullptr; },
00118         [rendererPlugin]() { return rendererPlugin; }, [shaderIRPlugin]() { return shaderIRPlugin; }, shaderFactories,
00119         [windowPlugin]() { return windowPlugin; });
00120 }
00121
00122 static const std::vector<float> cubeVertices = {
00123     // positions          // colors
00124     -0.5f, -0.5f, -0.5f, 1, 0, 0, 0.5f,  -0.5f, -0.5f, 0, 1, 0, 0.5f,  0.5f,  -0.5f, 0, 0, 1,
00125     0.5f,  0.5f,  -0.5f, 0, 0, 1, -0.5f, 0.5f,  -0.5f, 1, 1, 0, -0.5f, -0.5f, -0.5f, 1, 0, 0,
00126
00127     -0.5f, -0.5f, 0.5f,  1, 0, 1, 0.5f,  -0.5f, 0.5f,  0, 1, 1, 0.5f,  0.5f,  0.5f,  1, 1, 1,
00128     0.5f,  0.5f,  0.5f,  1, 1, 1, -0.5f, 0.5f,  0.5f,  0, 0, 0, -0.5f, -0.5f, 0.5f,  1, 0, 1,
00129
00130     -0.5f, 0.5f,  0.5f,  1, 0, 0, -0.5f, 0.5f,  -0.5f, 0, 1, 0, -0.5f, -0.5f, -0.5f, 0, 0, 1,
00131     -0.5f, -0.5f, -0.5f, 0, 0, 1, -0.5f, -0.5f, 0.5f,  1, 1, 0, -0.5f, 0.5f,  0.5f,  1, 0, 0,
00132
00133     0.5f,  0.5f,  0.5f,  1, 0, 1, 0.5f,  0.5f,  -0.5f, 0, 1, 1, 0.5f,  -0.5f, -0.5f, 1, 1, 1,
00134     0.5f,  -0.5f, -0.5f, 1, 1, 1, 0.5f,  -0.5f, 0.5f,  0, 0, 0, 0.5f,  0.5f,  0.5f,  1, 0, 1,
00135
00136     -0.5f, -0.5f, -0.5f, 1, 0, 0, 0.5f,  -0.5f, -0.5f, 0, 1, 0, 0.5f,  -0.5f, 0.5f,  0, 0, 1,
00137     0.5f,  -0.5f, 0.5f,  0, 0, 1, -0.5f, -0.5f, 0.5f,  1, 1, 0, -0.5f, -0.5f, -0.5f, 1, 0, 0,
00138
00139     -0.5f, 0.5f,  -0.5f, 1, 0, 1, 0.5f,  0.5f,  -0.5f, 0, 1, 1, 0.5f,  0.5f,  0.5f,  1, 1, 1,
00140     0.5f,  0.5f,  0.5f,  1, 1, 1, -0.5f, 0.5f,  0.5f,  0, 0, 0, -0.5f, 0.5f,  -0.5f, 1, 0, 1};
00141
00142 void cae::Application::start()
00143 {
```

```
00144     static const std::vector<ShaderSourceDesc> shaderSources = {
00145         {.id = "basic_vertex",
00146          .type = ShaderSourceType::GLSL,
00147          .source = utl::fileToString(utl::Path::resolveRelativeToExe("assets/shaders/glsl/texture.vert")),
00148          .stage = ShaderStage::VERTEX},
00149
00150         {.id = "basic_fragment",
00151          .type = ShaderSourceType::GLSL,
00152          .source = utl::fileToString(utl::Path::resolveRelativeToExe("assets/shaders/glsl/texture.frag")),
00153          .stage = ShaderStage::FRAGMENT},
00154     };
00155     m_engine->initializeRenderResources(shaderSources, cubeVertices);
00156     mainLoop();
00157 }
00158
00159 void cae::Application::stop()
00160 {
00161     m_engine->stop();
00162
00163     m_pluginLoader = nullptr;
00164     m_engine = nullptr;
00165 }
00166
00167 void cae::Application::mainLoop()
00168 {
00169     std::array<float, 10> fpsBuffer{};
00170     int fpsIndex = 0;
00171     WindowEvent e{};
00172
00173     while (!m_engine->getWindow()->shouldClose())
00174     {
00175         m_engine->render();
00176         glm::vec3 moveDir(0.0F);
00177         glm::vec2 lookDir(0.0F);
00178         m_engine->getWindow()->pollEvents();
00179         while (m_engine->getWindow()->pollEvent(e))
00180         {
00181             if (e.type == WindowEventType::KeyDown)
00182             {
00183                 m_keyState[e.key.key] = true;
00184             }
00185             else if (e.type == WindowEventType::KeyUp)
00186             {
00187                 m_keyState[e.key.key] = false;
00188             }
00189         }
00190
00191         if (m_keyState[KeyCode::Up])
00192         {
00193             lookDir.y += 1.0F;
00194         }
00195         if (m_keyState[KeyCode::Down])
00196         {
00197             lookDir.y -= 1.0F;
00198         }
00199         if (m_keyState[KeyCode::Left])
00200         {
00201             lookDir.x -= 1.0F;
00202         }
00203         if (m_keyState[KeyCode::Right])
00204         {
00205             lookDir.x += 1.0F;
00206         }
00207
00208         if (glm::length(lookDir) > 0.0F)
00209         {
00210             lookDir *= m_engine->getCamera()->getLookSpeed() * m_engine->getClock()->getDeltaSeconds();
00211             m_engine->getCamera()->rotate(lookDir.x, lookDir.y, 1.0F);
00212         }
00213
00214         glm::vec3 forward = glm::normalize(
00215             glm::vec3(m_engine->getCamera()->getDirection().x, 0.0F, m_engine->getCamera()->getDirection().z));
00216         glm::vec3 right = glm::normalize(glm::cross(forward, glm::vec3(0.0F, 1.0F, 0.0F)));
00217
00218         if (m_keyState[KeyCode::W])
00219         {
00220             moveDir += forward;
00221         }
00222         if (m_keyState[KeyCode::S])
00223         {
00224             moveDir -= forward;
00225         }
00226         if (m_keyState[KeyCode::A])
00227         {
00228             moveDir -= right;
00229         }
00230         if (m_keyState[KeyCode::D])
```

```
00231         {
00232             moveDir += right;
00233         }
00234
00235         if (glm::length(moveDir) > 0.0F)
00236         {
00237             moveDir = glm::normalize(moveDir);
00238             m_engine->getCamera()->move(moveDir, m_engine->getClock()->getDeltaSeconds());
00239         }
00240
00241         if (m_keyState[KeyCode::LCtrl])
00242         {
00243             m_engine->getCamera()->move(glm::vec3(0.0F, -1.0F, 0.0F), m_engine->getClock()->getDeltaSeconds());
00244         }
00245         if (m_keyState[KeyCode::Space])
00246         {
00247             m_engine->getCamera()->move(glm::vec3(0.0F, 1.0F, 0.0F), m_engine->getClock()->getDeltaSeconds());
00248         }
00249
00250         m_engine->update(fpsBuffer, fpsIndex);
00251     }
00252 }
```

43.261 src/argsHandler.cpp File Reference

#include ”CAE/ArgsHandler.hpp”
#include ”CAE/Common.hpp”
#include ”Utils/Env.hpp”
#include <iostream>
Include dependency graph for argsHandler.cpp:



43.262 argsHandler.cpp

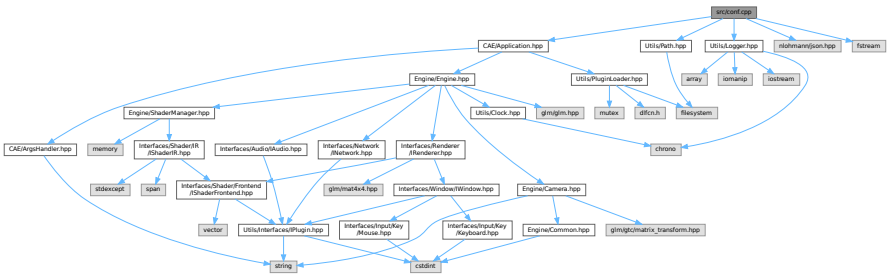
Go to the documentation of this file.

```
00001 #include ”CAE/ArgsHandler.hpp”
00002 #include ”CAE/Common.hpp”
00003
00004 #include ”Utils/Env.hpp”
00005
00006 #include <iostream>
00007
00008 cae::ArgsConfig cae::ArgsHandler::ParseArgs(const int argc, const char *const *argv)
00009 {
00010     ArgsConfig config;
00011     config.run = true;
00012
00013     if (argc <= 1)
00014     {
00015         return config;
00016     }
00017
00018     for (int i = 1; i < argc; ++i)
00019     {
00020         std::string arg = argv[i];
00021
00022         if (arg == ”-h” || arg == ”--help”)
00023         {
```

```
00024             std::cout « MESSAGE::HELP_MSG;
00025             config.run = false;
00026             return config;
00027         }
00028         if (arg == "-v" || arg == "--version")
00029         {
00030             std::cout « MESSAGE::VERSION_MSG;
00031             config.run = false;
00032             return config;
00033         }
00034         if (arg == "-c" || arg == "--config")
00035         {
00036             if (i + 1 >= argc)
00037             {
00038                 throw std::runtime_error("Missing value for argument " + arg);
00039             }
00040
00041             config.config_path = argv[++i];
00042         }
00043         else
00044         {
00045             throw std::runtime_error("Unknown argument: " + arg + ". Use -h or --help to see available options.");
00046         }
00047     }
00048
00049     return config;
00050 }
00051
00052 cae::EnvConfig cae::ArgsHandler::ParseEnv(const char *const *envp)
00053 {
00054     EnvConfig config;
00055     const auto envMap = utl::getEnvMap(envp);
00056
00057     if (envMap.contains("USER"))
00058     {
00059         config.user_name = envMap.at("USER");
00060     }
00061     if (envMap.contains("PWD"))
00062     {
00063         config.pwd = envMap.at("PWD");
00064     }
00065
00066     return config;
00067 }
```

43.263 src/conf.cpp File Reference

#include "CAE/Application.hpp"
#include "Utils/Logger.hpp"
#include "Utils/Path.hpp"
#include <nlohmann/json.hpp>
#include <fstream>
Include dependency graph for conf.cpp:



Typedefs

- using json = nlohmann::json

43.263.1 Typedef Documentation

43.263.1.1 json

using json = nlohmann::json
Definition at line 10 of file conf.cpp.

43.264 conf.cpp

Go to the documentation of this file.

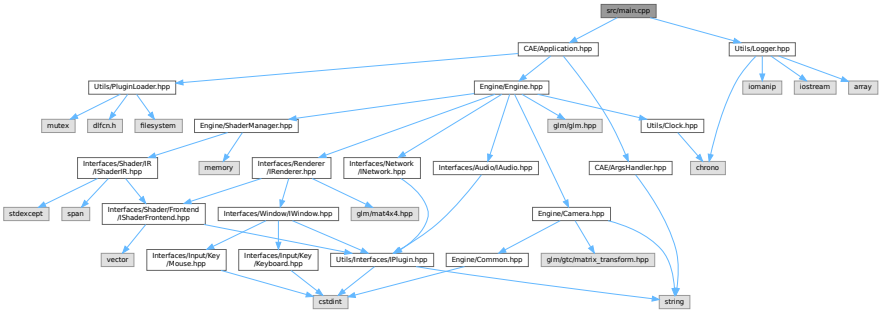
```
00001 #include "CAE/Application.hpp"
00002
00003 #include "Utils/Logger.hpp"
00004 #include "Utils/Path.hpp"
00005
00006 #include <nlohmann/json.hpp>
00007
00008 #include <fstream>
00009
00010 using json = nlohmann::json;
00011
00012 cae::EngineConfig cae::Application::parseEngineConf(const std::string &path)
00013 {
00014     auto confPath = utl::Path::resolveRelativeToCwd(path);
00015     if (!utl::Path::existsFile(confPath))
00016     {
00017         utl::Logger::log("Config file does not exist: " + confPath.string(), utl::LogLevel::WARNING);
00018         return {};
00019     }
00020
00021     std::ifstream file(confPath);
00022     if (!file.is_open())
00023     {
00024         utl::Logger::log("Failed to open config file: " + confPath.string(), utl::LogLevel::WARNING);
00025         return {};
00026     }
00027
00028     json j;
00029     try
00030     {
00031         file » j;
00032     }
00033     catch (const json::parse_error &e)
00034     {
00035         utl::Logger::log("Failed to parse JSON config (" + confPath.string() + "): " + std::string(e.what()),
00036                          utl::LogLevel::WARNING);
00037         return {};
00038     }
00039     EngineConfig config;
00040     utl::Logger::log("Loading config: " + confPath.string(), utl::LogLevel::INFO);
00041     if (j.contains("audio"))
00042     {
00043         const auto &audio = j["audio"];
00044         if (audio.contains("masterVolume") && audio["masterVolume"].is_number_float())
00045         {
00046             config.audio_master_volume = audio["masterVolume"];
00047         }
00048         if (audio.contains("muted") && audio["muted"].is_boolean())
00049         {
00050             config.audio_muted = audio["muted"];
00051         }
00052     }
00053     if (j.contains("camera"))
00054     {
00055         const auto &camera = j["camera"];
00056         if (camera.contains("position") && camera["position"].is_array() && camera["position"].size() == 3)
00057         {
00058             if (const auto &position = camera["position"];
00059                 position[0].is_number_float() && position[1].is_number_float() && position[2].is_number_float())
00060             {
00061                 config.camera_position.x = position[0];
00062                 config.camera_position.y = position[1];
00063                 config.camera_position.z = position[2];
00064             }
00065         }
00066         if (camera.contains("rotation") && camera["rotation"].is_array() && camera["rotation"].size() == 3)
00067         {
00068             if (const auto &rotation = camera["rotation"];
00069                 rotation[0].is_number_float() && rotation[1].is_number_float() && rotation[2].is_number_float())
00070             {
00071                 config.camera_rotation.x = rotation[0];
00072                 config.camera_rotation.y = rotation[1];
00073                 config.camera_rotation.z = rotation[2];
```

```
00074             }
00075         }
00076         if (camera.contains("direction") && camera["direction"].is_array() && camera["direction"].size() == 3)
00077         {
00078             if (const auto &direction = camera["direction"];
00079                 direction[0].is_number_float() && direction[1].is_number_float() && direction[2].is_number_float())
00080             {
00081                 config.camera_direction.x = direction[0];
00082                 config.camera_direction.y = direction[1];
00083                 config.camera_direction.z = direction[2];
00084             }
00085         }
00086         if (camera.contains("movementSpeed") && camera["movementSpeed"].is_number_float())
00087         {
00088             config.camera_move_speed = camera["movementSpeed"];
00089         }
00090         if (camera.contains("rotationSpeed") && camera["rotationSpeed"].is_number_float())
00091         {
00092             config.camera_look_speed = camera["rotationSpeed"];
00093         }
00094         if (camera.contains("fov") && camera["fov"].is_number_unsigned())
00095         {
00096             config.camera_fov = camera["fov"];
00097         }
00098         if (camera.contains("nearPlane") && camera["nearPlane"].is_number_float())
00099         {
00100             config.camera_near_plane = camera["nearPlane"];
00101         }
00102         if (camera.contains("farPlane") && camera["farPlane"].is_number_float())
00103         {
00104             config.camera_far_plane = camera["farPlane"];
00105         }
00106     }
00107     if (j.contains("log"))
00108     {
00109         if (const auto &log = j["log"]; log.contains("fps") && log["fps"].is_boolean())
00110         {
00111             config.log_fps = log["fps"];
00112         }
00113     }
00114     if (j.contains("network"))
00115     {
00116         const auto &network = j["network"];
00117         if (network.contains("host") && network["host"].is_string())
00118         {
00119             config.network_host = network["host"];
00120         }
00121         if (network.contains("port") && network["port"].is_number_unsigned())
00122         {
00123             config.network_port = network["port"];
00124         }
00125     }
00126     if (j.contains("renderer"))
00127     {
00128         const auto &renderer = j["renderer"];
00129         if (renderer.contains("vsync") && renderer["vsync"].is_boolean())
00130         {
00131             config.renderer_vsync = renderer["vsync"];
00132         }
00133         if (renderer.contains("frameRateLimit") && renderer["frameRateLimit"].is_number_unsigned())
00134         {
00135             config.renderer_frame_rate_limit = renderer["frameRateLimit"];
00136         }
00137         if (renderer.contains("clearColor") && renderer["clearColor"].is_array() && renderer["clearColor"].size() == 4)
00138         {
00139             if (const auto &clearColor = renderer["clearColor"];
00140                 clearColor[0].is_number_float() && clearColor[1].is_number_float() && clearColor[2].is_number_float() &&
00141                 clearColor[3].is_number_float())
00142             {
00143                 config.renderer_clear_color.r = clearColor[0];
00144                 config.renderer_clear_color.g = clearColor[1];
00145                 config.renderer_clear_color.b = clearColor[2];
00146                 config.renderer_clear_color.a = clearColor[3];
00147             }
00148         }
00149     }
00150     if (j.contains("window"))
00151     {
00152         const auto &window = j["window"];
00153         if (window.contains("width") && window["width"].is_number_unsigned())
00154         {
00155             config.window_width = window["width"];
00156         }
00157         if (window.contains("height") && window["height"].is_number_unsigned())
00158         {
00159             config.window_height = window["height"];
00160         }
```

```
00161         if (window.contains("fullscreen") && window["fullscreen"].is_boolean())
00162         {
00163             config.window_fullscreen = window["fullscreen"];
00164         }
00165         if (window.contains("name") && window["name"].is_string())
00166         {
00167             config.window_name = window["name"];
00168         }
00169         if (window.contains("iconPath") && window["iconPath"].is_string())
00170         {
00171             config.window_icon_path = utl::Path::resolveRelativeToExe(window["iconPath"]).string();
00172         }
00173     }
00174
00175     return config;
00176 }
```

43.265 src/main.cpp File Reference

#include "CAE/Application.hpp"
#include "Utils/Logger.hpp"
Include dependency graph for main.cpp:



Functions

- int main (const int argc, const char *const *argv, const char *const *envp)

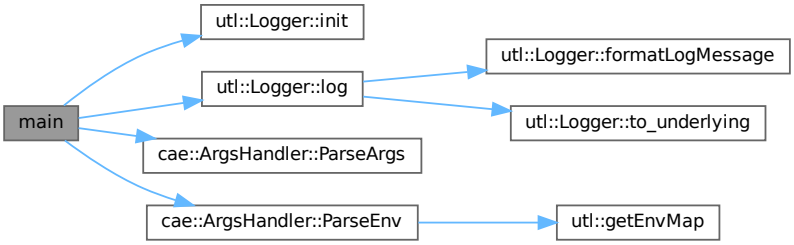
43.265.1 Function Documentation

43.265.1.1 main()

int main (
const int argc,
const char *const * argv,
const char *const * envp)

Definition at line 5 of file main.cpp.
References utl::Logger::init(), utl::Logger::log(), cae::ArgsHandler::ParseArgs(), cae::ArgsHandler::ParseEnv(), cae::ArgsConfig::run, and utl::WARNING.

Here is the call graph for this function:



43.266 main.cpp

Go to the documentation of this file.

```
00001 #include "CAE/Application.hpp"
00002
00003 #include "Utils/Logger.hpp"
00004
00005 int main(const int argc, const char *const *argv, const char *const *envp)
00006 {
00007     std::unique_ptr<cae::Application> app = nullptr;
00008
00009     utl::Logger::init();
00010     try
00011     {
00012         cae::ArgsConfig argsConfig = cae::ArgsHandler::ParseArgs(argc, argv);
00013         cae::EnvConfig envConfig = cae::ArgsHandler::ParseEnv(envp);
00014         if (!argsConfig.run)
00015         {
00016             return EXIT_SUCCESS;
00017         }
00018         app = std::make_unique<cae::Application>(argsConfig, envConfig);
00019         app->start();
00020         app->stop();
00021     }
00022     catch (const std::exception &e)
00023     {
00024         utl::Logger::log(std::string("Exception: ") + e.what() + '\n', utl::LogLevel::WARNING);
00025         return EXIT_FAILURE;
00026     }
00027     catch (...)
00028     {
00029         utl::Logger::log(std::string("Unknown exception"), utl::LogLevel::WARNING);
00030         return EXIT_FAILURE;
00031     }
00032     return EXIT_SUCCESS;
00033 }
```

Index