cae

0.0.0

Generated by Doxygen 1.11.0

Chapter 1

cae

1.1 Cross-API-Engine | Rendering Engine with multiple dynamic backends

Cross-API-Engine is a rendering engine designed to support multiple graphics APIs dynamically. It allows developers to switch between different rendering backends such as OpenGL, Vulkan, DirectX at runtime. It is useful to do benchmarks during development or to support multiple platforms with different graphics APIs.

```
flowchart LR
  subgraph CORE["  Engine Core"]
    Engine["Engine"]
  end

  subgraph Interfaces["  Interfaces"]
    Engine --> IAudio["IAudio"]
    Engine --> INetwork["INetwork"]
    Engine --> IRenderer["IRenderer"]
    Engine --> IWindow["IWindow"]
    Engine --> IInput["IInput"]
  end

  subgraph AUDIO["  Audio Plugins"]
    WASAPI["WASAPI / XAudio2 (Windows)"]
    Pulse["Pulse / ALSA (Linux)"]
    CoreAudio["CoreAudio (macOS)"]
  end
  IAudio --> WASAPI & Pulse & CoreAudio

  subgraph NETWORK["  Network Plugins"]
    WinSock["WinSock (Windows)"]
    POSIX["POSIX Sockets (Linux/macOS)"]
  end
  INetwork --> WinSock & POSIX

  subgraph RENDERER["  Renderer Plugins"]
    DirectX["DirectX (Windows)"]
    Vulkan["Vulkan (Cross-platform)"]
    OpenGL["OpenGL (Cross-platform)"]
    Metal["Metal (macOS)"]
  end
  IRenderer --> DirectX & Vulkan & OpenGL & Metal

  subgraph WINDOW["  Window System Plugins"]
    Win32["Win32 (Windows)"]
    X11["X11 / Wayland (Linux)"]
    Cocoa["Cocoa (macOS)"]
  end
  IWindow --> Win32 & X11 & Cocoa

  subgraph INPUT["  Input Plugins"]
    Keyboard["Keyboard"]
    Mouse["Mouse"]
    Controller["Gamepad / Controller"]
  end
  IInput --> Keyboard & Mouse & Controller
```

1.1.1 Prerequisites

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

- CMake 4.0.0
- C++23
- Vulkan SDK

1.1.2 External Libraries

- GLFW: For creating windows, receiving input, and managing OpenGL and Vulkan contexts.
- Google Test: A testing framework for C++.
- ImGui: Immediate Mode Graphical User Interface for real-time debugging and tool development.
- stb: A set of single-file public domain libraries for graphics, image loading, and more.

1.1.3 Contributing

Want to contribute? See CONTRIBUTING.md.

1.1.4 License

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

Chapter 2

README

Chapter 3

README

Chapter 4

README

Chapter 5

README

Chapter 6

README

Chapter 7

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 8

LICENSE

Chapter 9

Namespace Index

9.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 10

Hierarchical Index

10.1 Class Hierarchy

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

Chapter 11

Class Index

11.1 Class List

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

Chapter 12

File Index

12.1 File List

Here is a list of all files with brief descriptions:

Chapter 13

Namespace Documentation

13.1 cae Namespace Reference

Namespaces

- namespace Audio
- namespace Message
- namespace Network
- namespace Plugins
- namespace Renderer
- namespace User
- namespace Window

Classes

- struct AppConfig
- class Application

  Main class.
- struct ArgsConfig
- class ArgsHandler

  Class to handle command line arguments.
- class Engine

  Engine class.
- struct EngineConfig
- struct EnvConfig
- class GLFW

  Class for the GLFW plugin.
- interface IAudio

  Interface for audio.
- interface IGamepad

  Interface for gamepad.
- interface IInput

  Interface for audio.
- interface IKeyboard

  Interface for keyboard.
- interface IMouse

Interface for mouse.

- interface INetwork

  Interface for network.
- interface IRenderer

  Interface for renderer.
- interface IWindow

  Interface for window.
- struct NativeWindowHandle
- class OPGL

  Class for the OpenGL plugin.
- class VULKN

  Class for the Vulkan plugin.
- struct WindowSize
- class X11

  Class for the X11 plugin.

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 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 ,
  Count }
- enum class MouseButton : uint8_t {
  Left = 0 , Right , Middle , XButton1 ,
  XButton2 , WheelUp , WheelDown }

13.1.1 Enumeration Type Documentation

13.1.1.1 GamepadAxis

enum class cae::GamepadAxis : uint8_t [strong]

Enumerator

LeftX	
LeftY	
RightX	
RightY	
TriggerLeft	
TriggerRight	

Definition at line 32 of file Gamepad.hpp.

13.1.1.2 GamepadButton

enum class cae::GamepadButton : uint8_t [strong]

Enumerator

A	
B	
X	
Y	
Back	
Guide	
Start	
LThumb	
RThumb	
LShoulder	
RShoulder	
DPadUp	
DPadDown	
DPadLeft	
DPadRight	

Definition at line 13 of file Gamepad.hpp.

13.1.1.3 KeyCode

enum class cae::KeyCode : uint8_t [strong]

Enumerator

A	

Enumerator

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	

Enumerator

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	
Count	

Definition at line 22 of file Keyboard.hpp.

13.1.1.4 KeyState

enum cae::KeyState : std::uint8_t

Enumerator

Pressed	
Released	
Held	
Toggled	

Definition at line 14 of file Keyboard.hpp.

13.1.1.5 MouseButton

enum class cae::MouseButton : uint8_t [strong]

Enumerator

Left	
Right	
Middle	
XButton1	
XButton2	
WheelUp	
WheelDown	

Definition at line 13 of file Mouse.hpp.

13.2 cae::Audio Namespace Reference

Variables

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

13.2.1 Variable Documentation

13.2.1.1 MUTED

auto cae::Audio::MUTED = false [inline], [constexpr]

Definition at line 25 of file Common.hpp.

13.2.1.2 VOLUME

auto cae::Audio::VOLUME = 1.F [inline], [constexpr]

Definition at line 24 of file Common.hpp.

13.3 cae::Message Namespace Reference

Variables

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

13.3.1 Variable Documentation

13.3.1.1 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 30 of file Common.hpp.

Referenced by cae::ArgsHandler::ParseArgs().

13.3.1.2 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 35 of file Common.hpp.

Referenced by cae::Application::Application(), and cae::ArgsHandler::ParseArgs().

13.4 cae::Network Namespace Reference

Variables

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

13.4.1 Variable Documentation

13.4.1.1 HOST

auto cae::Network::HOST = "127.0.0.1" [inline], [constexpr]

Definition at line 41 of file Common.hpp.

13.4.1.2 PORT

auto cae::Network::PORT = 4242 [inline], [constexpr]

Definition at line 42 of file Common.hpp.

13.5 cae::Plugins Namespace Reference

Namespaces

- namespace Name

13.6 cae::Plugins::Name Namespace Reference

Variables

- constexpr auto RENDERER_OPENGL = "OpenGL"
- constexpr auto RENDERER_VULKAN = "Vulkan"
- constexpr auto WINDOW_GLFW = "GLFW"

13.6.1 Variable Documentation

13.6.1.1 RENDERER_OPENGL

auto cae::Plugins::Name::RENDERER_OPENGL = "OpenGL" [inline], [constexpr]

Definition at line 47 of file Common.hpp.

13.6.1.2 RENDERER_VULKAN

auto cae::Plugins::Name::RENDERER_VULKAN = "Vulkan" [inline], [constexpr]

Definition at line 48 of file Common.hpp.

13.6.1.3 WINDOW_GLFW

auto cae::Plugins::Name::WINDOW_GLFW = "GLFW" [inline], [constexpr]

Definition at line 49 of file Common.hpp.

13.7 cae::Renderer Namespace Reference

Variables

- constexpr auto VSYNC = false
- constexpr auto FRAME_RATE_LIMIT = 90

13.7.1 Variable Documentation

13.7.1.1 FRAME_RATE_LIMIT

auto cae::Renderer::FRAME_RATE_LIMIT = 90 [inline], [constexpr]

Definition at line 55 of file Common.hpp.

13.7.1.2 VSYNC

auto cae::Renderer::VSYNC = false [inline], [constexpr]

Definition at line 54 of file Common.hpp.

13.8 cae::User Namespace Reference

Variables

- constexpr auto NAME = "User"

13.8.1 Variable Documentation

13.8.1.1 NAME

auto cae::User::NAME = "User" [inline], [constexpr]

Definition at line 60 of file Common.hpp.

13.9 cae::Window Namespace Reference

Variables

- constexpr uint16_t WIDTH = 1920
- constexpr uint16_t HEIGHT = 1080
- constexpr auto NAME = "CAE - Cross API Engine"
- constexpr auto FULLSCREEN = false

13.9.1 Variable Documentation

13.9.1.1 FULLSCREEN

auto cae::Window::FULLSCREEN = false [inline], [constexpr]

Definition at line 68 of file Common.hpp.

13.9.1.2 HEIGHT

uint16_t cae::Window::HEIGHT = 1080 [inline], [constexpr]

Definition at line 66 of file Common.hpp.

13.9.1.3 NAME

auto cae::Window::NAME = ”CAE - Cross API Engine” [inline], [constexpr]

Definition at line 67 of file Common.hpp.

13.9.1.4 WIDTH

uint16_t cae::Window::WIDTH = 1920 [inline], [constexpr]

Definition at line 65 of file Common.hpp.

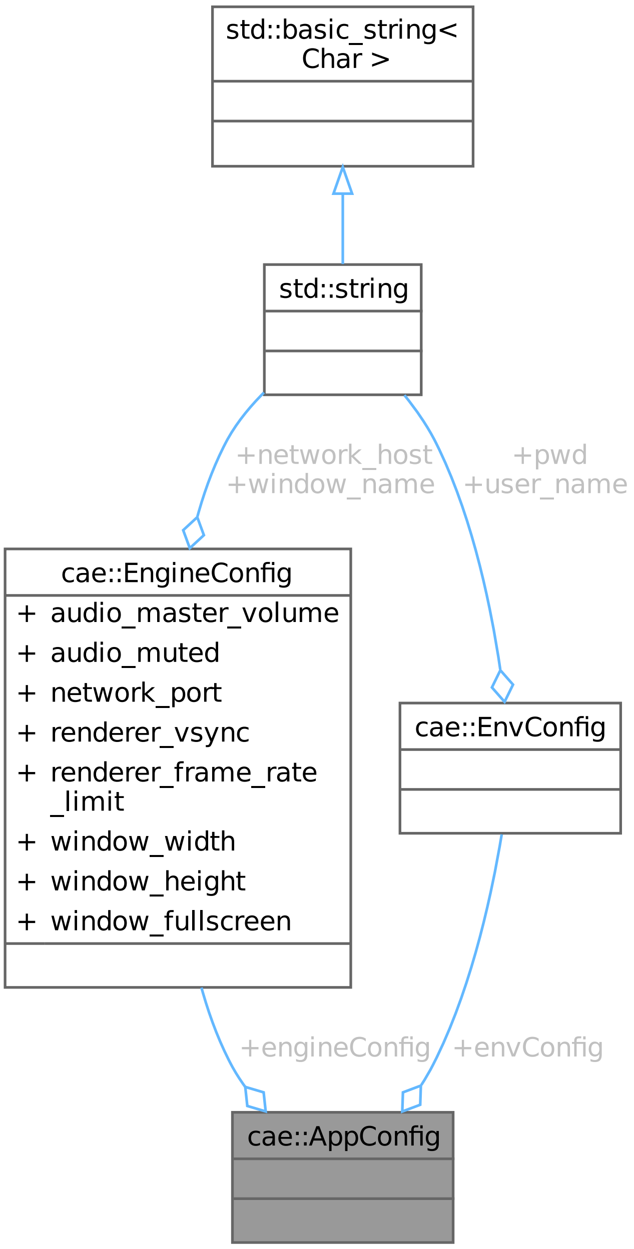
Chapter 14

Class Documentation

14.1 cae::AppConfig Struct Reference

#include <Application.hpp>

Collaboration diagram for cae::AppConfig:



Public Attributes

- EngineConfig engineConfig
- EnvConfig envConfig

14.1.1 Detailed Description

Definition at line 17 of file Application.hpp.

14.1.2 Member Data Documentation

14.1.2.1 engineConfig

EngineConfig cae::AppConfig::engineConfig

Definition at line 19 of file Application.hpp.

Referenced by cae::Application::Application().

14.1.2.2 envConfig

EnvConfig cae::AppConfig::envConfig

Definition at line 20 of file Application.hpp.

Referenced by cae::Application::Application().

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

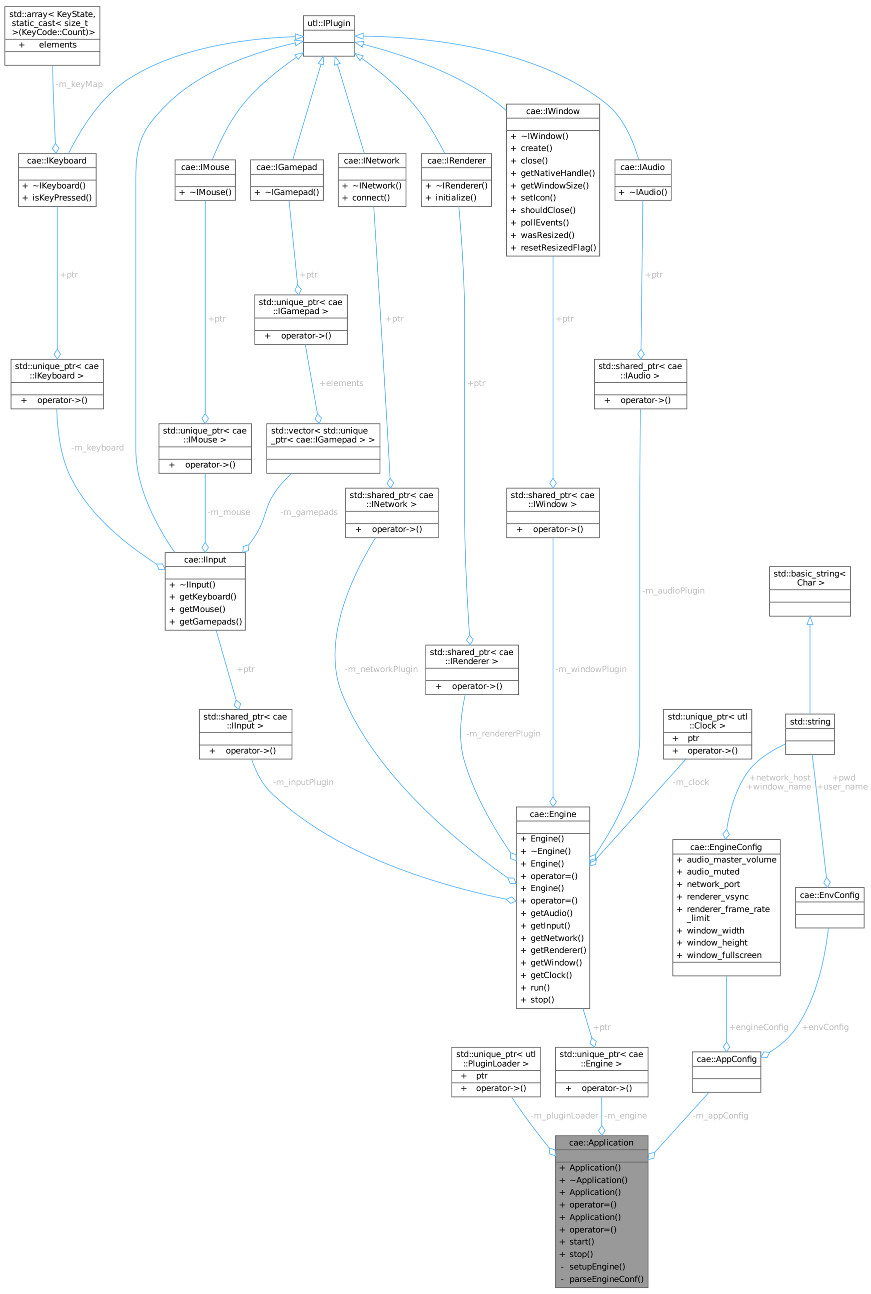
- include/CAE/Application.hpp

14.2 cae::Application Class Reference

Main class.

#include <Application.hpp>

Collaboration diagram for cae::Application:



Public Member Functions

- Application (const ArgsConfig &argsConfig, const EnvConfig &envConfig)
- ∼Application ()=default
- Application (const Application &)=delete
- Application & operator= (const Application &)=delete
- Application (Application &&)=delete
- Application & operator= (Application &&)=delete
- void start () const
- void stop ()

Private Member Functions

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

Static Private Member Functions

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

Private Attributes

- std::unique_ptr< utl::PluginLoader > m_pluginLoader = nullptr
- std::unique_ptr< Engine > m_engine = nullptr
- AppConfig m_appConfig

14.2.1 Detailed Description

Main class.

Definition at line 28 of file Application.hpp.

14.2.2 Constructor & Destructor Documentation

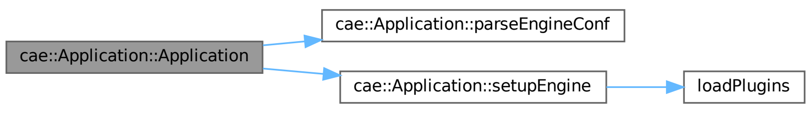
14.2.2.1 Application() [1/3]

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

Definition at line 34 of file application.cpp.

References cae::ArgsConfig::config_path, cae::AppConfig::engineConfig, cae::AppConfig::envConfig, m_appConfig, parseEngineConf(), setupEngine(), and cae::Message::VERSION_MSG.

Here is the call graph for this function:


cae::Application::Application
cae::Application::parseEngineConf
cae::Application::setupEngine
loadPlugins

14.2.2.2 ~Application()

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

14.2.2.3 Application() [2/3]

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

14.2.2.4 Application() [3/3]

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

14.2.3 Member Function Documentation

14.2.3.1 operator=() [1/2]

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

14.2.3.2 operator=() [2/2]

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

14.2.3.3 parseEngineConf()

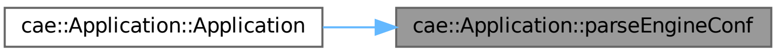
cae::EngineConfig cae::Application::parseEngineConf (
    const std::string & path) [static], [private]

Definition at line 10 of file conf.cpp.

References cae::EngineConfig::audio_master_volume, cae::EngineConfig::audio_muted, cae::EngineConfig::network_host
cae::EngineConfig::network_port, cae::EngineConfig::renderer_frame_rate_limit, cae::EngineConfig::renderer_vsync, cae::EngineConfig::window_fullscreen, cae::EngineConfig::window_height, cae::EngineConfig::window_name, and cae::EngineConfig::window_width.

Referenced by Application().

Here is the caller graph for this function:


cae::Application::Application
cae::Application::parseEngineConf

14.2.3.4 setupEngine()

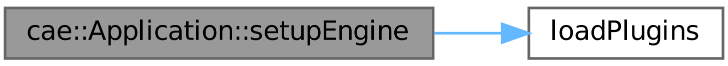
```
void cae::Application::setupEngine (
        const std::string & rendererName,
        const std::string & windowName) [private]
```

Definition at line 55 of file application.cpp.

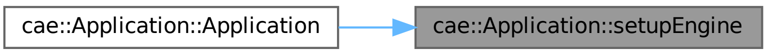
References loadPlugins().

Referenced by Application().

Here is the call graph for this function:



Here is the caller graph for this function:



14.2.3.5 start()

```
void cae::Application::start () const
```

Definition at line 82 of file application.cpp.

14.2.3.6 stop()

```
void cae::Application::stop ()
```

Definition at line 84 of file application.cpp.

14.2.4 Member Data Documentation

14.2.4.1 m_appConfig

```
AppConfig cae::Application::m_appConfig [private]
```

Definition at line 51 of file Application.hpp.

Referenced by Application().

14.2.4.2 m_engine

std::unique_ptr<Engine> cae::Application::m_engine = nullptr [private]

Definition at line 49 of file Application.hpp.

14.2.4.3 m_pluginLoader

std::unique_ptr<utl::PluginLoader> cae::Application::m_pluginLoader = nullptr [private]

Definition at line 48 of file Application.hpp.

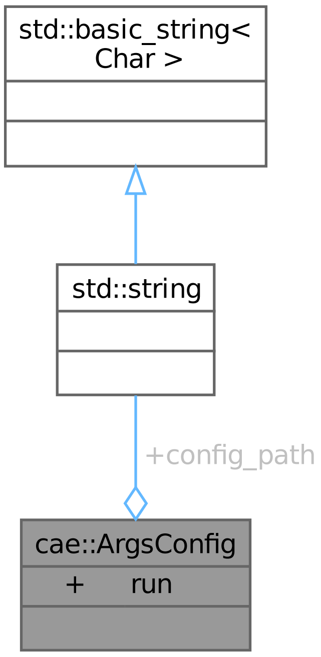
The documentation for this class was generated from the following files:

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

14.3 cae::ArgsConfig Struct Reference

#include <ArgsHandler.hpp>

Collaboration diagram for cae::ArgsConfig:



Public Attributes

- bool run = false
- std::string config_path

14.3.1 Detailed Description

Definition at line 14 of file ArgsHandler.hpp.

14.3.2 Member Data Documentation

14.3.2.1 config_path

std::string cae::ArgsConfig::config_path

Definition at line 17 of file ArgsHandler.hpp.

Referenced by cae::Application::Application(), and cae::ArgsHandler::ParseArgs().

14.3.2.2 run

bool cae::ArgsConfig::run = false

Definition at line 16 of file ArgsHandler.hpp.

Referenced by main(), and cae::ArgsHandler::ParseArgs().

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

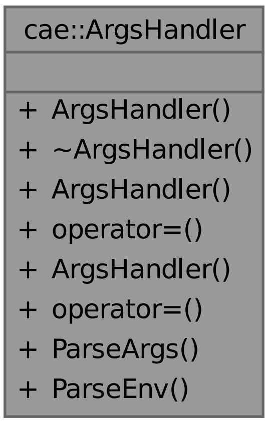
- include/CAE/ArgsHandler.hpp

14.4 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)
- static EnvConfig ParseEnv (const char ∗const ∗envp)

14.4.1 Detailed Description

Class to handle command line arguments.

Definition at line 30 of file ArgsHandler.hpp.

14.4.2 Constructor & Destructor Documentation

14.4.2.1 ArgsHandler() [1/3]

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

14.4.2.2 ∼ArgsHandler()

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

14.4.2.3 ArgsHandler() [2/3]

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

14.4.2.4 ArgsHandler() [3/3]

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

14.4.3 Member Function Documentation

14.4.3.1 operator=() [1/2]

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

14.4.3.2 operator=() [2/2]

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

14.4.3.3 ParseArgs()

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

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:

main
cae::ArgsHandler::ParseArgs

14.4.3.4 ParseEnv()

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

Definition at line 52 of file argsHandler.cpp.

References cae::EnvConfig::pwd, and cae::EnvConfig::user_name.

Referenced by main().

Here is the caller graph for this function:

main
cae::ArgsHandler::ParseEnv

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

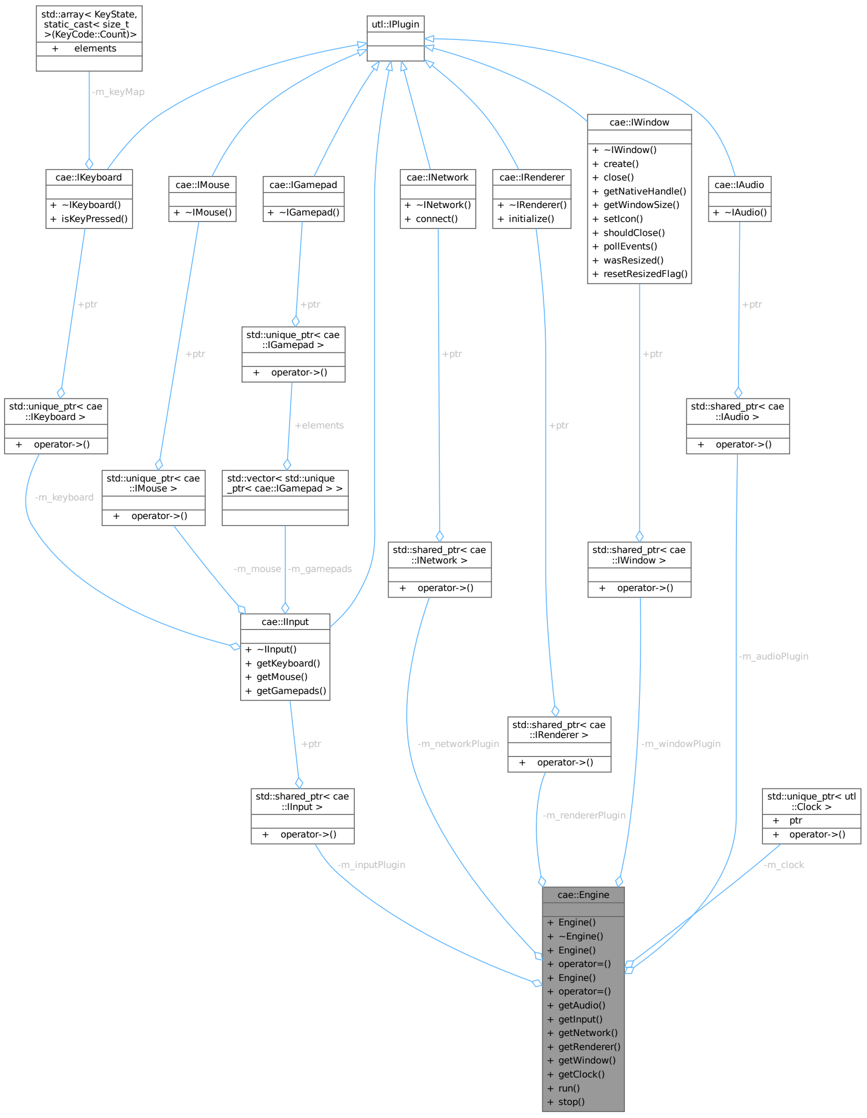
- include/CAE/ArgsHandler.hpp
- src/argsHandler.cpp

14.5 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 >()> &audioFactory, const std::function< std::shared_ptr< IInput >()> &inputFactory, const std↩ ::function< std::shared_ptr< INetwork >()> &networkFactory, const std::function< std::shared↩ _ptr< IRenderer >()> &rendererFactory, 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< IInput > & getInput () 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 ()
- void run () const
- void stop ()

Private Attributes

- std::shared_ptr< IAudio > m_audioPlugin = nullptr
- std::shared_ptr< IInput > m_inputPlugin = 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

14.5.1 Detailed Description

Engine class.

Definition at line 45 of file Engine.hpp.

14.5.2 Constructor & Destructor Documentation

14.5.2.1 Engine() [1/3]

```
cae::Engine::Engine (
        const EngineConfig & config,
        const std::function< std::shared_ptr< IAudio >()> & audioFactory,
        const std::function< std::shared_ptr< IInput >()> & inputFactory,
        const std::function< std::shared_ptr< INetwork >()> & networkFactory,
        const std::function< std::shared_ptr< IRenderer >()> & rendererFactory,
        const std::function< std::shared_ptr< IWindow >()> & windowFactory)
```

Definition at line 5 of file engine.cpp.

References cae::EngineConfig::audio_master_volume, cae::EngineConfig::audio_muted, m_windowPlugin, cae::EngineConfig::network_host, cae::EngineConfig::network_port, cae::EngineConfig::renderer_frame_rate_limit, cae::EngineConfig::renderer_vsync, cae::EngineConfig::window_fullscreen, cae::EngineConfig::window_height, cae::EngineConfig::window_name, and cae::EngineConfig::window_width.

14.5.2.2 ~Engine()

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

14.5.2.3 Engine() [2/3]

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

14.5.2.4 Engine() [3/3]

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

14.5.3 Member Function Documentation

14.5.3.1 getAudio()

const std::shared_ptr< IAudio > & cae::Engine::getAudio () const   [inline], [nodiscard]

Definition at line 61 of file Engine.hpp.

References m_audioPlugin.

14.5.3.2 getClock()

const std::unique_ptr< utl::Clock > & cae::Engine::getClock ()   [inline], [nodiscard]

Definition at line 67 of file Engine.hpp.

References m_clock.

14.5.3.3 getInput()

const std::shared_ptr< IInput > & cae::Engine::getInput () const   [inline], [nodiscard]

Definition at line 62 of file Engine.hpp.

References m_inputPlugin.

14.5.3.4 getNetwork()

const std::shared_ptr< INetwork > & cae::Engine::getNetwork () const   [inline], [nodiscard]

Definition at line 63 of file Engine.hpp.

References m_networkPlugin.

14.5.3.5 getRenderer()

const std::shared_ptr< IRenderer > & cae::Engine::getRenderer () const [inline], [nodiscard]

Definition at line 64 of file Engine.hpp.

References m_rendererPlugin.

14.5.3.6 getWindow()

const std::shared_ptr< IWindow > & cae::Engine::getWindow () const [inline], [nodiscard]

Definition at line 65 of file Engine.hpp.

References m_windowPlugin.

14.5.3.7 operator=() [1/2]

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

14.5.3.8 operator=() [2/2]

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

14.5.3.9 run()

void cae::Engine::run () const

Definition at line 29 of file engine.cpp.

14.5.3.10 stop()

void cae::Engine::stop ()

Definition at line 37 of file engine.cpp.

14.5.4 Member Data Documentation

14.5.4.1 m_audioPlugin

std::shared_ptr<IAudio> cae::Engine::m_audioPlugin = nullptr [private]

Definition at line 73 of file Engine.hpp.

Referenced by getAudio().

14.5.4.2 m_clock

std::unique_ptr<utl::Clock> cae::Engine::m_clock = nullptr [private]

Definition at line 79 of file Engine.hpp.

Referenced by getClock().

14.5.4.3 m_inputPlugin

std::shared_ptr<IInput> cae::Engine::m_inputPlugin = nullptr [private]

Definition at line 74 of file Engine.hpp.

Referenced by getInput().

14.5.4.4 m_networkPlugin

std::shared_ptr<INetwork> cae::Engine::m_networkPlugin = nullptr [private]

Definition at line 75 of file Engine.hpp.

Referenced by getNetwork().

14.5.4.5 m_rendererPlugin

std::shared_ptr<IRenderer> cae::Engine::m_rendererPlugin = nullptr [private]

Definition at line 76 of file Engine.hpp.

Referenced by getRenderer().

14.5.4.6 m_windowPlugin

std::shared_ptr<IWindow> cae::Engine::m_windowPlugin = nullptr [private]

Definition at line 77 of file Engine.hpp.

Referenced by Engine(), and getWindow().

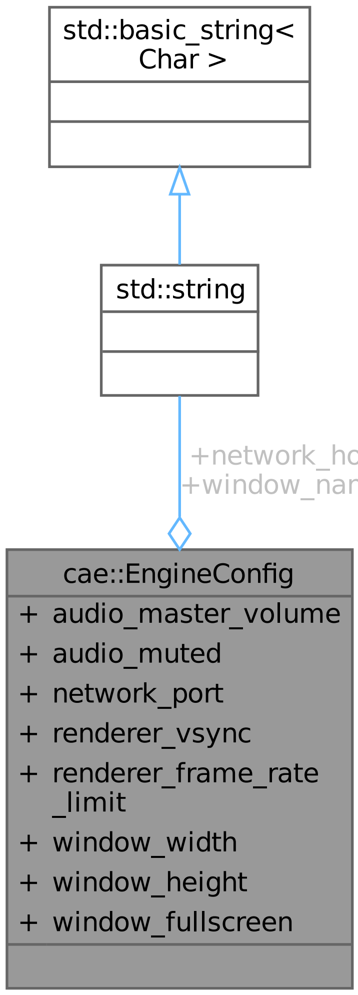
The documentation for this class was generated from the following files:

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

14.6 cae::EngineConfig Struct Reference

#include <Engine.hpp>

Collaboration diagram for cae::EngineConfig:



Public Attributes

- float audio_master_volume = Audio::VOLUME
- bool audio_muted = Audio::MUTED
- 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
- uint16_t window_width = Window::WIDTH
- uint16_t window_height = Window::HEIGHT
- bool window_fullscreen = Window::FULLSCREEN
- std::string window_name = Window::NAME

14.6.1 Detailed Description

Definition at line 23 of file Engine.hpp.

14.6.2 Member Data Documentation

14.6.2.1 audio_master_volume

float cae::EngineConfig::audio_master_volume = Audio::VOLUME

Definition at line 25 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.2 audio_muted

bool cae::EngineConfig::audio_muted = Audio::MUTED

Definition at line 26 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.3 network_host

std::string cae::EngineConfig::network_host = Network::HOST

Definition at line 28 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.4 network_port

uint16_t cae::EngineConfig::network_port = Network::PORT

Definition at line 29 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.5 renderer_frame_rate_limit

uint16_t cae::EngineConfig::renderer_frame_rate_limit = Renderer::FRAME_RATE_LIMIT

Definition at line 32 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.6 renderer_vsync

bool cae::EngineConfig::renderer_vsync = Renderer::VSYNC

Definition at line 31 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.7 window_fullscreen

bool cae::EngineConfig::window_fullscreen = Window::FULLSCREEN

Definition at line 36 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.8 window_height

uint16_t cae::EngineConfig::window_height = Window::HEIGHT

Definition at line 35 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.9 window_name

std::string cae::EngineConfig::window_name = Window::NAME

Definition at line 37 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

14.6.2.10 window_width

uint16_t cae::EngineConfig::window_width = Window::WIDTH

Definition at line 34 of file Engine.hpp.

Referenced by cae::Engine::Engine(), and cae::Application::parseEngineConf().

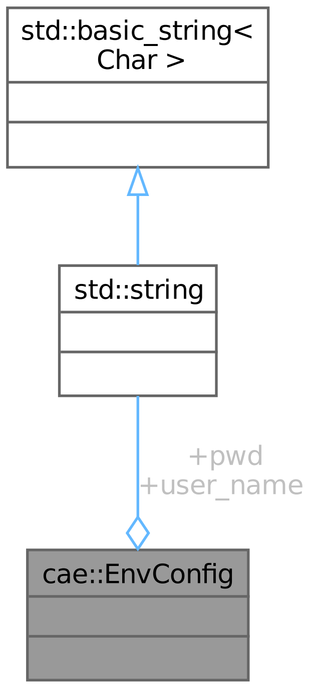
The documentation for this struct was generated from the following file:

- include/CAE/Engine/Engine.hpp

14.7 cae::EnvConfig Struct Reference

#include <ArgsHandler.hpp>

Collaboration diagram for cae::EnvConfig:



Public Attributes

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

14.7.1 Detailed Description

Definition at line 19 of file ArgsHandler.hpp.

14.7.2 Member Data Documentation

14.7.2.1 pwd

std::string cae::EnvConfig::pwd

Definition at line 22 of file ArgsHandler.hpp.

Referenced by cae::ArgsHandler::ParseEnv().

14.7.2.2 user_name

std::string cae::EnvConfig::user_name

Definition at line 21 of file ArgsHandler.hpp.

Referenced by cae::ArgsHandler::ParseEnv().

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

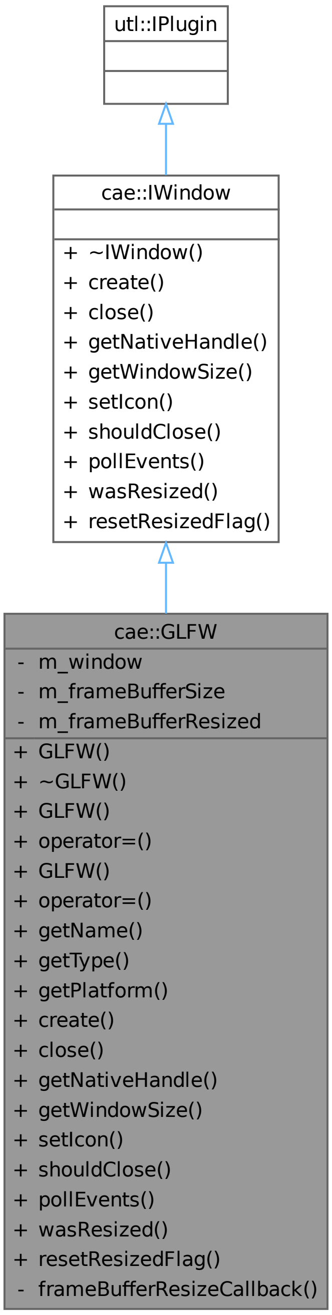
- include/CAE/ArgsHandler.hpp

14.8 cae::GLFW Class Reference

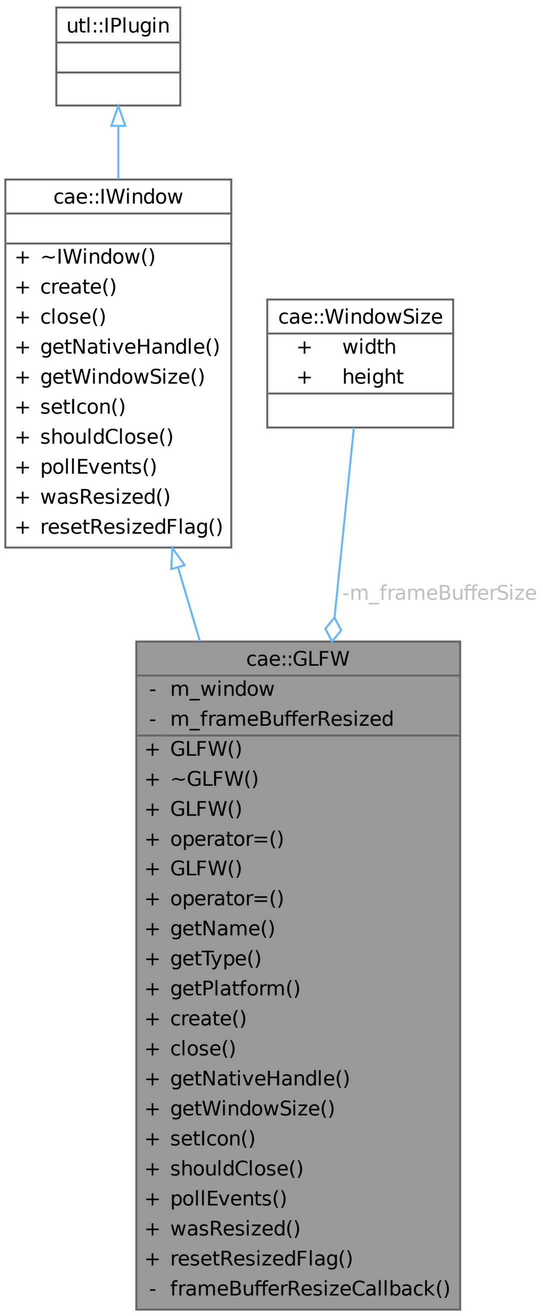
Class for the GLFW plugin.

#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
- utl::PluginType getType () const override
- utl::PluginPlatform getPlatform () const override
- bool create (const std::string &name, WindowSize size) override
- void close () override
- NativeWindowHandle getNativeHandle () const override
- WindowSize getWindowSize () const override
- bool setIcon (const std::string &path) const override
- bool shouldClose () const override
- void pollEvents () override
- bool wasResized () const override
- void resetResizedFlag () override

Public Member Functions inherited from cae::IWindow

- ∼IWindow () override=default

Static Private Member Functions

- static void frameBufferResizeCallback (GLFWwindow ∗window, int width, int height)

Private Attributes

- GLFWwindow ∗ m_window = nullptr
- WindowSize m_frameBufferSize
- bool m_frameBufferResized = false

14.8.1 Detailed Description

Class for the GLFW plugin.

Definition at line 30 of file GLFW.hpp.

14.8.2 Constructor & Destructor Documentation

14.8.2.1 GLFW() [1/3]

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

14.8.2.2 ∼GLFW()

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

14.8.2.3 GLFW() [2/3]

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

14.8.2.4 GLFW() [3/3]

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

14.8.3 Member Function Documentation

14.8.3.1 close()

void cae::GLFW::close () [override], [virtual]

Implements cae::IWindow.

Definition at line 37 of file glfw.cpp.

14.8.3.2 create()

bool cae::GLFW::create (
    const std::string & name,
    WindowSize size) [override], [virtual]

Implements cae::IWindow.

Definition at line 13 of file glfw.cpp.

References cae::WindowSize::height, and cae::WindowSize::width.

14.8.3.3 frameBufferResizeCallback()

void cae::GLFW::frameBufferResizeCallback (
    GLFWwindow * window,
    int width,
    int height) [static], [private]

Definition at line 6 of file glfw.cpp.

References m_frameBufferResized.

14.8.3.4 getName()

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

Definition at line 42 of file GLFW.hpp.

14.8.3.5 getNativeHandle()

cae::NativeWindowHandle cae::GLFW::getNativeHandle () const [nodiscard], [override], [virtual]

Implements cae::IWindow.

Definition at line 55 of file glfw.cpp.

References cae::NativeWindowHandle::window.

14.8.3.6 getPlatform()

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

Definition at line 44 of file GLFW.hpp.

14.8.3.7 getType()

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

Definition at line 43 of file GLFW.hpp.

14.8.3.8 getWindowSize()

cae::WindowSize cae::GLFW::getWindowSize () const [nodiscard], [override], [virtual]

Implements cae::IWindow.

Definition at line 47 of file glfw.cpp.

14.8.3.9 operator=() [1/2]

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

14.8.3.10 operator=() [2/2]

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

14.8.3.11 pollEvents()

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

Implements cae::IWindow.

Definition at line 55 of file GLFW.hpp.

14.8.3.12 resetResizedFlag()

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

Implements cae::IWindow.

Definition at line 58 of file GLFW.hpp.

References m_frameBufferResized.

14.8.3.13 setIcon()

bool cae::GLFW::setIcon (
    const std::string & path) const [nodiscard], [override], [virtual]

Implements cae::IWindow.

Definition at line 71 of file glfw.cpp.

14.8.3.14 shouldClose()

bool cae::GLFW::shouldClose () const [inline], [nodiscard], [override], [virtual]

Implements cae::IWindow.

Definition at line 54 of file GLFW.hpp.

References m_window.

14.8.3.15 wasResized()

bool cae::GLFW::wasResized () const [inline], [override], [virtual]

Implements cae::IWindow.

Definition at line 57 of file GLFW.hpp.

References m_frameBufferResized.

14.8.4 Member Data Documentation

14.8.4.1 m_frameBufferResized

bool cae::GLFW::m_frameBufferResized = false [private]

Definition at line 65 of file GLFW.hpp.

Referenced by frameBufferResizeCallback(), resetResizedFlag(), and wasResized().

14.8.4.2 m_frameBufferSize

WindowSize cae::GLFW::m_frameBufferSize [private]

Definition at line 64 of file GLFW.hpp.

14.8.4.3 m_window

GLFWwindow∗ cae::GLFW::m_window = nullptr [private]

Definition at line 63 of file GLFW.hpp.

Referenced by shouldClose().

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

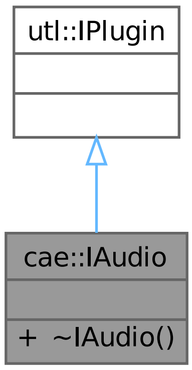
- plugins/Window/GLFW/include/GLFW/GLFW.hpp
- plugins/Window/GLFW/src/glfw.cpp

14.9 cae::IAudio Interface Reference

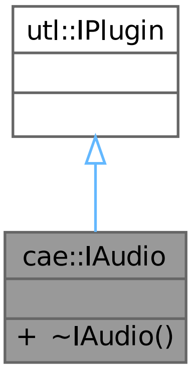
Interface for audio.

#include <IAudio.hpp>

Inheritance diagram for cae::IAudio:



Collaboration diagram for cae::IAudio:



Public Member Functions

- ∼IAudio () override=default

14.9.1 Detailed Description

Interface for audio.

Definition at line 19 of file IAudio.hpp.

14.9.2 Constructor & Destructor Documentation

14.9.2.1 ∼IAudio()

cae::IAudio::∼IAudio () [override], [default]

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

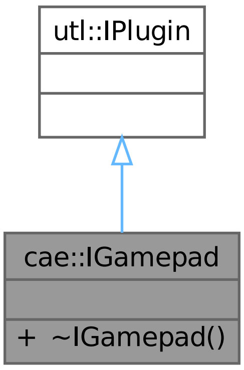
- modules/Interfaces/include/Interfaces/IAudio.hpp

14.10 cae::IGamepad Interface Reference

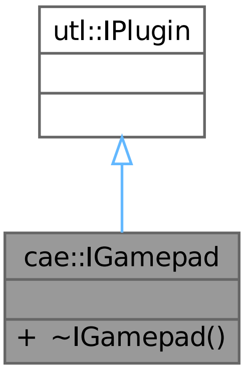
Interface for gamepad.

#include <IGamepad.hpp>

Inheritance diagram for cae::IGamepad:



Collaboration diagram for cae::IGamepad:



Public Member Functions

- ~IGamepad () override=default

14.10.1 Detailed Description

Interface for gamepad.

Definition at line 19 of file IGamepad.hpp.

14.10.2 Constructor & Destructor Documentation

14.10.2.1 ~IGamepad()

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

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

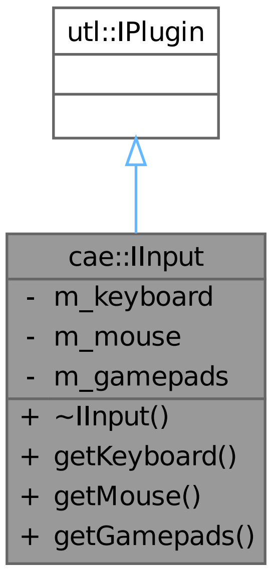
- modules/Interfaces/include/Interfaces/Input/IGamepad.hpp

14.11 cae::IInput Interface Reference

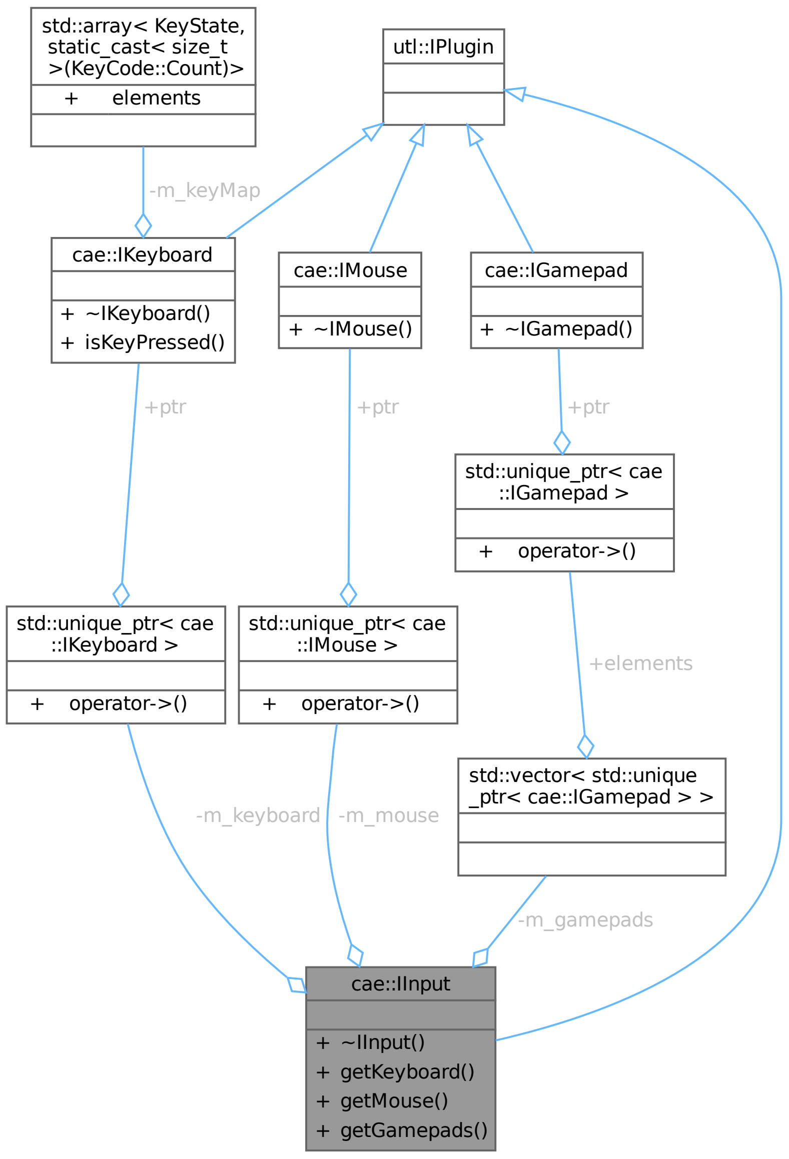
Interface for audio.

#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

Private Attributes

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

14.11.1 Detailed Description

Interface for audio.

Definition at line 25 of file IInput.hpp.

14.11.2 Constructor & Destructor Documentation

14.11.2.1 ∼IInput()

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

14.11.3 Member Function Documentation

14.11.3.1 getGamepads()

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

14.11.3.2 getKeyboard()

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

14.11.3.3 getMouse()

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

14.11.4 Member Data Documentation

14.11.4.1 m_gamepads

std::vector<std::unique_ptr<IGamepad> > cae::IInput::m_gamepads [private]

Definition at line 38 of file IInput.hpp.

14.11.4.2 m_keyboard

std::unique_ptr<IKeyboard> cae::IInput::m_keyboard [private]

Definition at line 36 of file IInput.hpp.

14.11.4.3 m_mouse

std::unique_ptr<IMouse> cae::IInput::m_mouse [private]

Definition at line 37 of file IInput.hpp.

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

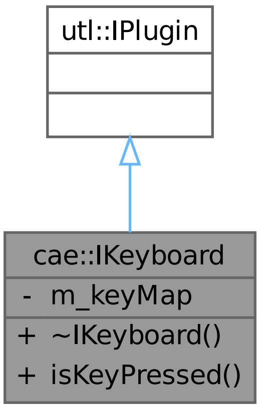
- modules/Interfaces/include/Interfaces/Input/IInput.hpp

14.12 cae::IKeyboard Interface Reference

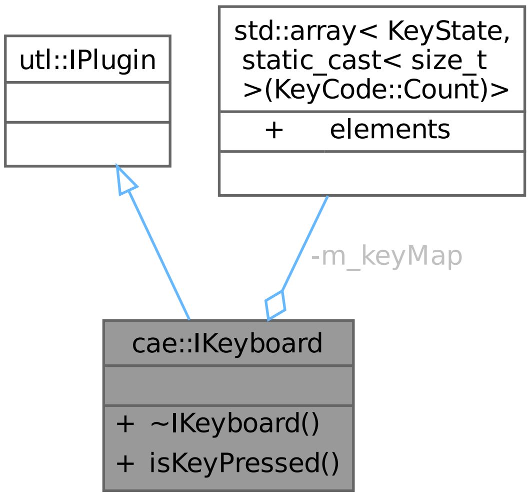
Interface for keyboard.

#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

Private Attributes

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

14.12.1 Detailed Description

Interface for keyboard.

Definition at line 23 of file IKeyboard.hpp.

14.12.2 Constructor & Destructor Documentation

14.12.2.1 ~IKeyboard()

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

14.12.3 Member Function Documentation

14.12.3.1 isKeyPressed()

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

14.12.4 Member Data Documentation

14.12.4.1 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:

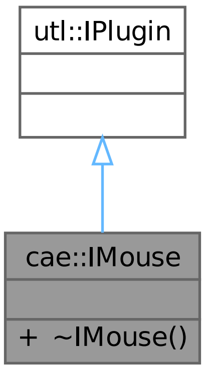
- modules/Interfaces/include/Interfaces/Input/IKeyboard.hpp

14.13 cae::IMouse Interface Reference

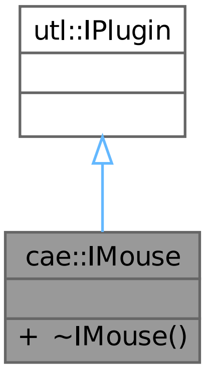
Interface for mouse.

#include <IMouse.hpp>

Inheritance diagram for cae::IMouse:



Collaboration diagram for cae::IMouse:



Public Member Functions

- ∼IMouse () override=default

14.13.1 Detailed Description

Interface for mouse.

Definition at line 19 of file IMouse.hpp.

14.13.2 Constructor & Destructor Documentation

14.13.2.1 ∼IMouse()

cae::IMouse::∼IMouse () [override], [default]

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

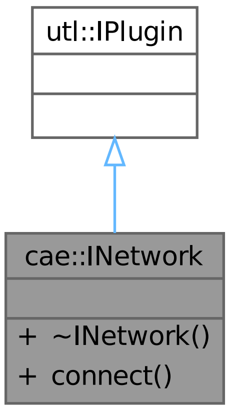
- modules/Interfaces/include/Interfaces/Input/IMouse.hpp

14.14 cae::INetwork Interface Reference

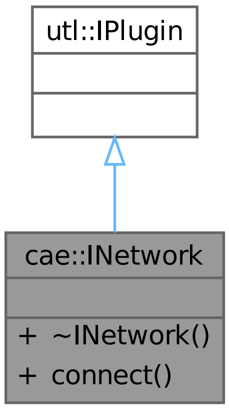
Interface for network.

#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

14.14.1 Detailed Description

Interface for network.

Definition at line 19 of file INetwork.hpp.

14.14.2 Constructor & Destructor Documentation

14.14.2.1 ∼INetwork()

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

14.14.3 Member Function Documentation

14.14.3.1 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:

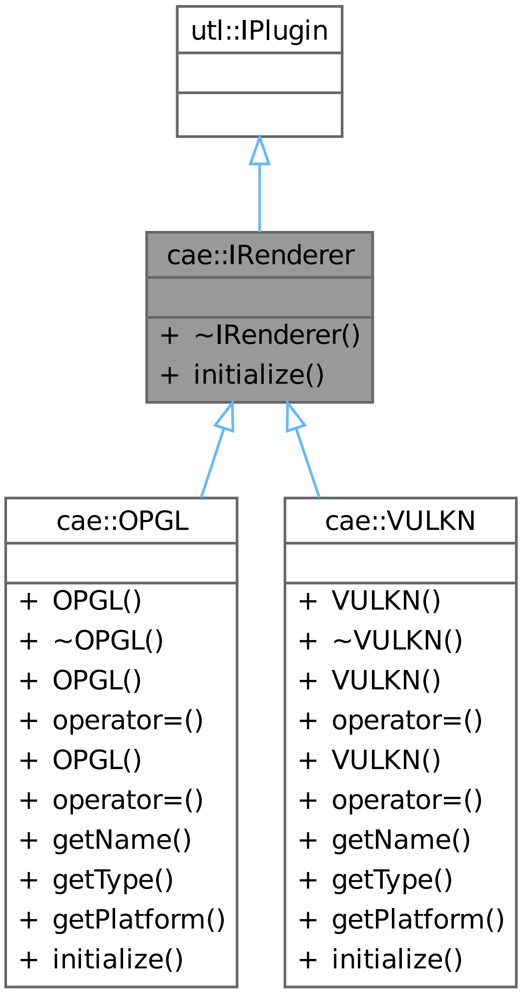
- modules/Interfaces/include/Interfaces/INetwork.hpp

14.15 cae::IRenderer Interface Reference

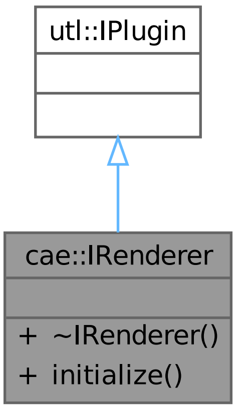
Interface for renderer.

#include <IRenderer.hpp>

Inheritance diagram for cae::IRenderer:



Collaboration diagram for cae::IRenderer:



Public Member Functions

- ∼IRenderer () override=default
- virtual void initialize (void ∗nativeWindowHandle)=0

14.15.1 Detailed Description

Interface for renderer.

Definition at line 19 of file IRenderer.hpp.

14.15.2 Constructor & Destructor Documentation

14.15.2.1 ∼IRenderer()

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

14.15.3 Member Function Documentation

14.15.3.1 initialize()

virtual void cae::IRenderer::initialize (
void ∗ nativeWindowHandle) [pure virtual]

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

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

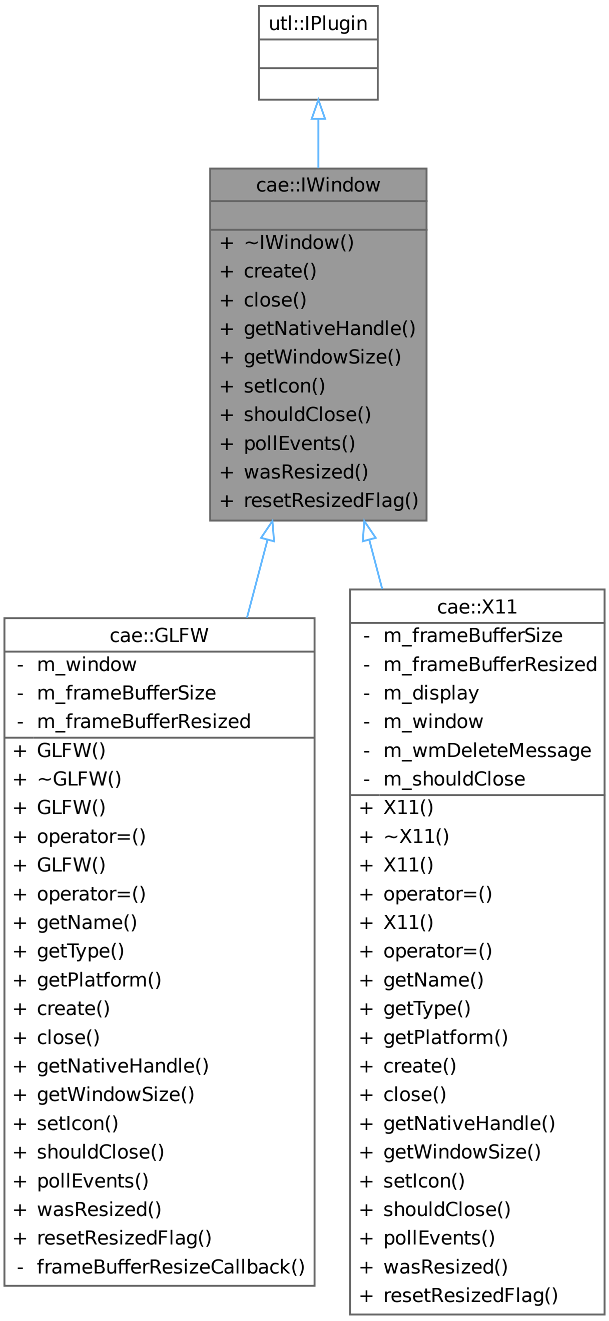
- modules/Interfaces/include/Interfaces/IRenderer.hpp

14.16 cae::IWindow Interface Reference

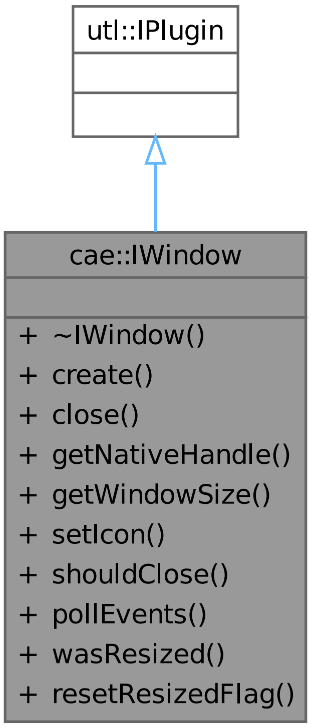
Interface for window.

#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
- virtual void close ()=0
- virtual NativeWindowHandle getNativeHandle () const =0
- virtual WindowSize getWindowSize () const =0
- virtual bool setIcon (const std::string &path) const =0
- virtual bool shouldClose () const =0
- virtual void pollEvents ()=0
- virtual bool wasResized () const =0
- virtual void resetResizedFlag ()=0

14.16.1 Detailed Description

Interface for window.

Definition at line 31 of file IWindow.hpp.

14.16.2 Constructor & Destructor Documentation

14.16.2.1 ~IWindow()

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

14.16.3 Member Function Documentation

14.16.3.1 close()

virtual void cae::IWindow::close () [pure virtual]

Implemented in cae::GLFW, and cae::X11.

14.16.3.2 create()

virtual bool cae::IWindow::create (
  const std::string & name,
  WindowSize size) [pure virtual]

Implemented in cae::GLFW, and cae::X11.

14.16.3.3 getNativeHandle()

virtual NativeWindowHandle cae::IWindow::getNativeHandle () const [pure virtual]

Implemented in cae::GLFW, and cae::X11.

14.16.3.4 getWindowSize()

virtual WindowSize cae::IWindow::getWindowSize () const [pure virtual]

Implemented in cae::GLFW, and cae::X11.

14.16.3.5 pollEvents()

virtual void cae::IWindow::pollEvents () [pure virtual]

Implemented in cae::GLFW, and cae::X11.

14.16.3.6 resetResizedFlag()

virtual void cae::IWindow::resetResizedFlag () [pure virtual]

Implemented in cae::GLFW, and cae::X11.

14.16.3.7 setIcon()

virtual bool cae::IWindow::setIcon (
    const std::string & path) const [pure virtual]

Implemented in cae::GLFW, and cae::X11.

14.16.3.8 shouldClose()

virtual bool cae::IWindow::shouldClose () const [pure virtual]

Implemented in cae::GLFW, and cae::X11.

14.16.3.9 wasResized()

virtual bool cae::IWindow::wasResized () const [pure virtual]

Implemented in cae::GLFW, and cae::X11.

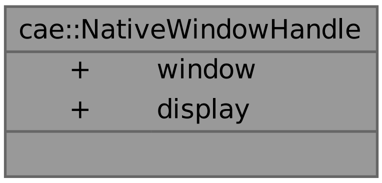
The documentation for this interface was generated from the following file:

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

14.17 cae::NativeWindowHandle Struct Reference

#include <IWindow.hpp>

Collaboration diagram for cae::NativeWindowHandle:



Public Attributes

- void ∗ window
- void ∗ display

14.17.1 Detailed Description

Definition at line 20 of file IWindow.hpp.

14.17.2 Member Data Documentation

14.17.2.1 display

void∗ cae::NativeWindowHandle::display

Definition at line 23 of file IWindow.hpp.

14.17.2.2 window

void∗ cae::NativeWindowHandle::window

Definition at line 22 of file IWindow.hpp.

Referenced by cae::GLFW::getNativeHandle(), and cae::X11::getNativeHandle().

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

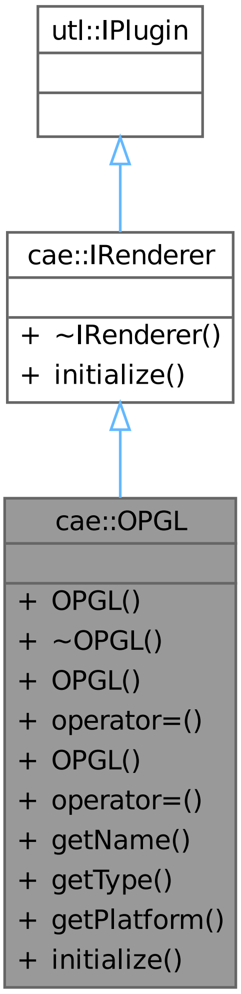
- modules/Interfaces/include/Interfaces/IWindow.hpp

14.18 cae::OPGL Class Reference

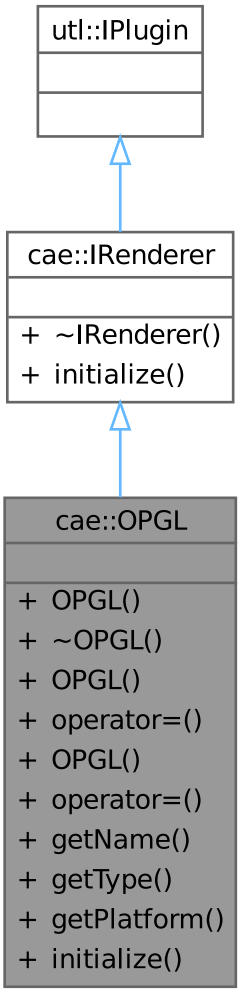
Class for the OpenGL plugin.

#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
- utl::PluginType getType () const override
- utl::PluginPlatform getPlatform () const override
- void initialize (void ∗nativeWindowHandle) override

Public Member Functions inherited from cae::IRenderer

- ∼IRenderer () override=default

14.18.1 Detailed Description

Class for the OpenGL plugin.

Definition at line 19 of file OPGL.hpp.

14.18.2 Constructor & Destructor Documentation

14.18.2.1 OPGL() [1/3]

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

14.18.2.2 ~OPGL()

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

14.18.2.3 OPGL() [2/3]

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

14.18.2.4 OPGL() [3/3]

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

14.18.3 Member Function Documentation

14.18.3.1 getName()

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

Definition at line 31 of file OPGL.hpp.

14.18.3.2 getPlatform()

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

Definition at line 33 of file OPGL.hpp.

14.18.3.3 getType()

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

Definition at line 32 of file OPGL.hpp.

14.18.3.4 initialize()

```
void cae::OPGL::initialize (
                    void * nativeWindowHandle)   [inline], [override], [virtual]
```

Implements cae::IRenderer.

Definition at line 35 of file OPGL.hpp.

14.18.3.5 operator=() [1/2]

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

14.18.3.6 operator=() [2/2]

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

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

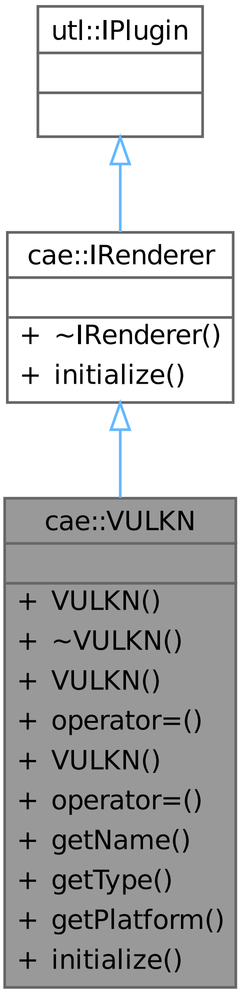
- plugins/Renderer/OpenGL/include/OPGL/OPGL.hpp

14.19 cae::VULKN Class Reference

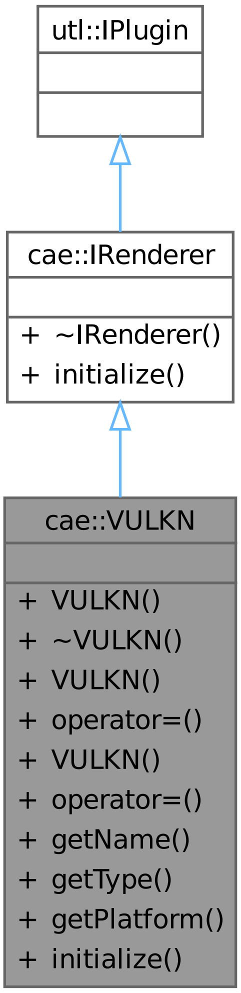
Class for the Vulkan plugin.

#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
- utl::PluginType getType () const override
- utl::PluginPlatform getPlatform () const override
- void initialize (void ∗nativeWindowHandle) override

Public Member Functions inherited from cae::IRenderer

- ∼IRenderer () override=default

14.19.1 Detailed Description

Class for the Vulkan plugin.

Definition at line 19 of file VULKN.hpp.

14.19.2 Constructor & Destructor Documentation

14.19.2.1 VULKN() [1/3]

cae::VULKN::VULKN () [default]

14.19.2.2 ~VULKN()

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

14.19.2.3 VULKN() [2/3]

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

14.19.2.4 VULKN() [3/3]

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

14.19.3 Member Function Documentation

14.19.3.1 getName()

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

Definition at line 31 of file VULKN.hpp.

14.19.3.2 getPlatform()

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

Definition at line 33 of file VULKN.hpp.

14.19.3.3 getType()

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

Definition at line 32 of file VULKN.hpp.

14.19.3.4 initialize()

```
void cae::VULKN::initialize (
                    void * nativeWindowHandle)   [inline], [override], [virtual]
```

Implements cae::IRenderer.

Definition at line 35 of file VULKN.hpp.

14.19.3.5 operator=() [1/2]

```
VULKN & cae::VULKN::operator= (
                    const VULKN & )   [delete]
```

14.19.3.6 operator=() [2/2]

```
VULKN & cae::VULKN::operator= (
                    VULKN && )   [delete]
```

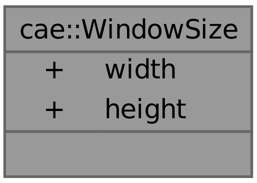
The documentation for this class was generated from the following file:

- plugins/Renderer/Vulkan/include/VULKN/VULKN.hpp

14.20 cae::WindowSize Struct Reference

#include <IWindow.hpp>

Collaboration diagram for cae::WindowSize:



Public Attributes

- uint16_t width
- uint16_t height

14.20.1 Detailed Description

Definition at line 14 of file IWindow.hpp.

14.20.2 Member Data Documentation

14.20.2.1 height

uint16_t cae::WindowSize::height

Definition at line 17 of file IWindow.hpp.

Referenced by cae::GLFW::create(), and cae::X11::create().

14.20.2.2 width

uint16_t cae::WindowSize::width

Definition at line 16 of file IWindow.hpp.

Referenced by cae::GLFW::create(), and cae::X11::create().

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

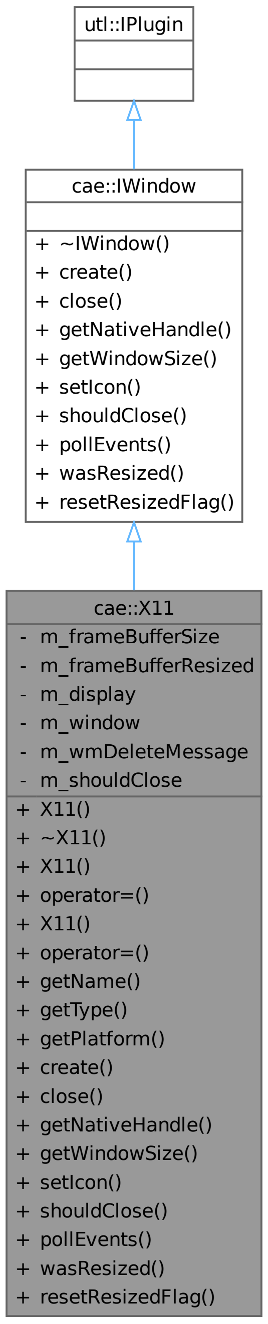
- modules/Interfaces/include/Interfaces/IWindow.hpp

14.21 cae::X11 Class Reference

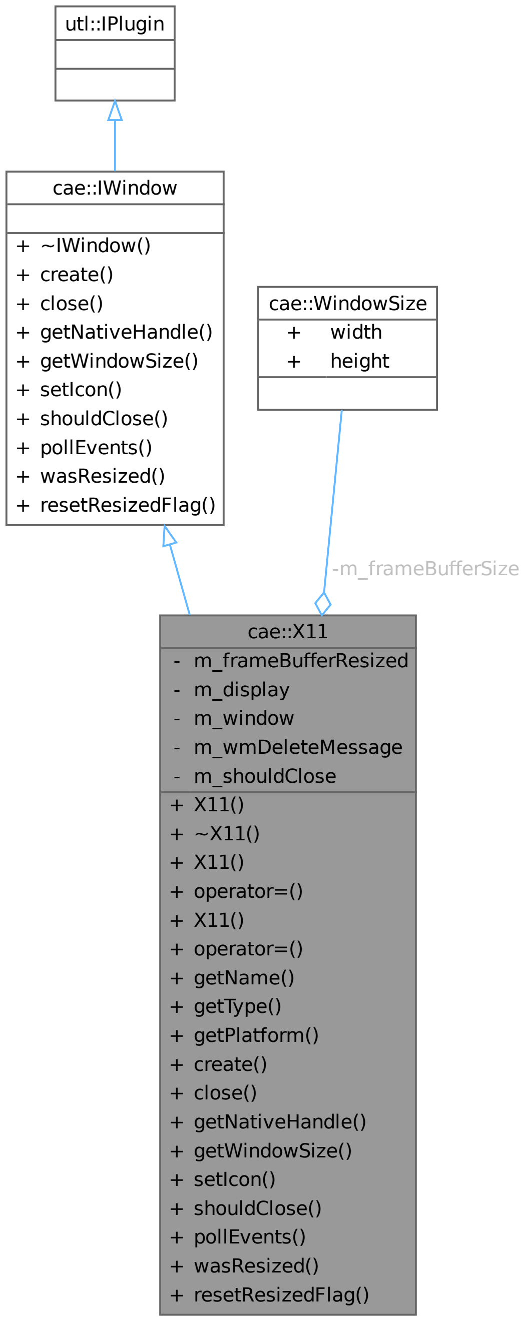
Class for the X11 plugin.

#include <X11.hpp>

Inheritance diagram for cae::X11:



Collaboration diagram for cae::X11:



Public Member Functions

- X11 ()=default
- ~X11 () override=default
- X11 (const X11 &)=delete
- X11 & operator= (const X11 &)=delete
- X11 (X11 &&)=delete

- X11 & operator= (X11 &&)=delete
- std::string getName () const override
- utl::PluginType getType () const override
- utl::PluginPlatform getPlatform () const override
- bool create (const std::string &name, WindowSize size) override
- void close () override
- NativeWindowHandle getNativeHandle () const override
- WindowSize getWindowSize () const override
- bool setIcon (const std::string &path) const override
- bool shouldClose () const override
- void pollEvents () override
- bool wasResized () const override
- void resetResizedFlag () override

Public Member Functions inherited from cae::IWindow

- ∼IWindow () override=default

Private Attributes

- WindowSize m_frameBufferSize
- bool m_frameBufferResized = false
- Display ∗ m_display = nullptr
- Window m_window = 0
- Atom m_wmDeleteMessage = 0
- bool m_shouldClose = false

14.21.1 Detailed Description

Class for the X11 plugin.

Definition at line 19 of file X11.hpp.

14.21.2 Constructor & Destructor Documentation

14.21.2.1 X11() [1/3]

cae::X11::X11 () [default]

14.21.2.2 ∼X11()

cae::X11::∼X11 () [override], [default]

14.21.2.3 X11() [2/3]

cae::X11::X11 (
const X11 &) [delete]

14.21.2.4 X11() [3/3]

cae::X11::X11 (
X11 &&) [delete]

14.21.3 Member Function Documentation

14.21.3.1 close()

void cae::X11::close () [override], [virtual]

Implements cae::IWindow.

Definition at line 44 of file x11.cpp.

References cae::U.

14.21.3.2 create()

bool cae::X11::create (
const std::string & name,
WindowSize size) [override], [virtual]

Implements cae::IWindow.

Definition at line 8 of file x11.cpp.

References cae::WindowSize::height, m_display, m_frameBufferSize, m_window, m_wmDeleteMessage, cae::U, and cae::WindowSize::width.

14.21.3.3 getName()

std::string cae::X11::getName () const [inline], [nodiscard], [override]

Definition at line 31 of file X11.hpp.

14.21.3.4 getNativeHandle()

NativeWindowHandle cae::X11::getNativeHandle () const [inline], [nodiscard], [override], [virtual]

Implements cae::IWindow.

Definition at line 38 of file X11.hpp.

References m_display, m_window, and cae::NativeWindowHandle::window.

14.21.3.5 getPlatform()

utl::PluginPlatform cae::X11::getPlatform () const [inline], [nodiscard], [override]

Definition at line 33 of file X11.hpp.

14.21.3.6 getType()

utl::PluginType cae::X11::getType () const [inline], [nodiscard], [override]

Definition at line 32 of file X11.hpp.

14.21.3.7 getWindowSize()

cae::WindowSize cae::X11::getWindowSize () const [nodiscard], [override], [virtual]

Implements cae::IWindow.

Definition at line 55 of file x11.cpp.

References cae::U.

14.21.3.8 operator=() [1/2]

X11 & cae::X11::operator= (
    const X11 &) [delete]

14.21.3.9 operator=() [2/2]

X11 & cae::X11::operator= (
    X11 &&) [delete]

14.21.3.10 pollEvents()

void cae::X11::pollEvents () [override], [virtual]

Implements cae::IWindow.

Definition at line 75 of file x11.cpp.

14.21.3.11 resetResizedFlag()

void cae::X11::resetResizedFlag () [inline], [override], [virtual]

Implements cae::IWindow.

Definition at line 50 of file X11.hpp.

References m_frameBufferResized.

14.21.3.12 setIcon()

bool cae::X11::setIcon (
const std::string & path) const [nodiscard], [override], [virtual]

Implements cae::IWindow.

Definition at line 67 of file x11.cpp.

14.21.3.13 shouldClose()

bool cae::X11::shouldClose () const [nodiscard], [override], [virtual]

Implements cae::IWindow.

Definition at line 73 of file x11.cpp.

14.21.3.14 wasResized()

bool cae::X11::wasResized () const [inline], [override], [virtual]

Implements cae::IWindow.

Definition at line 49 of file X11.hpp.

References m_frameBufferResized.

14.21.4 Member Data Documentation

14.21.4.1 m_display

Display∗ cae::X11::m_display = nullptr [private]

Definition at line 56 of file X11.hpp.

Referenced by create(), and getNativeHandle().

14.21.4.2 m_frameBufferResized

bool cae::X11::m_frameBufferResized = false [mutable], [private]

Definition at line 54 of file X11.hpp.

Referenced by resetResizedFlag(), and wasResized().

14.21.4.3 m_frameBufferSize

WindowSize cae::X11::m_frameBufferSize [private]

Definition at line 53 of file X11.hpp.

Referenced by create().

14.21.4.4 m_shouldClose

bool cae::X11::m_shouldClose = false [private]

Definition at line 59 of file X11.hpp.

14.21.4.5 m_window

Window cae::X11::m_window = 0 [private]

Definition at line 57 of file X11.hpp.

Referenced by create(), and getNativeHandle().

14.21.4.6 m_wmDeleteMessage

Atom cae::X11::m_wmDeleteMessage = 0 [private]

Definition at line 58 of file X11.hpp.

Referenced by create().

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

- plugins/Window/X11/include/X11/X11.hpp
- plugins/Window/X11/src/x11.cpp

Chapter 15

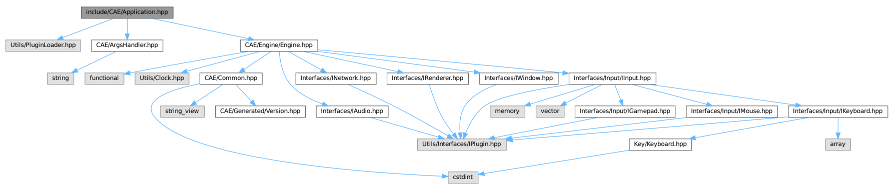
File Documentation

15.1 CONTRIBUTING.md File Reference

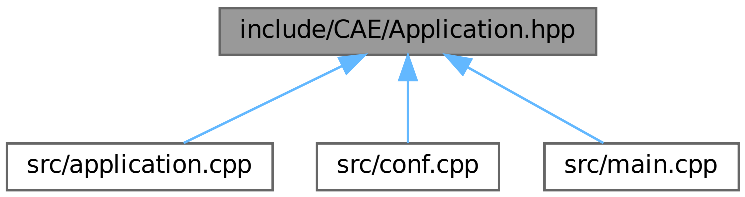
15.2 include/CAE/Application.hpp File Reference

This file contains the Application class declaration.

#include <Utils/PluginLoader.hpp>
#include "CAE/ArgsHandler.hpp"
#include "CAE/Engine/Engine.hpp"
Include dependency graph for Application.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::AppConfig
- class cae::Application

  Main class.

Namespaces

- namespace cae

15.2.1 Detailed Description

This file contains the Application class declaration.

Definition in file Application.hpp.

15.3 Application.hpp

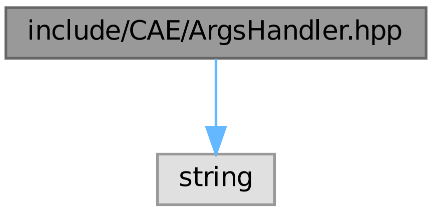
Go to the documentation of this file.

```
///
/// @file Application.hpp
/// @brief This file contains the Application class declaration
/// @namespace cae
///

#pragma once

#include <Utils/PluginLoader.hpp>

#include "CAE/ArgsHandler.hpp"
#include "CAE/Engine/Engine.hpp"

namespace cae
{

    struct AppConfig
    {
            EngineConfig engineConfig;
            EnvConfig envConfig;
    };

    ///
    /// @class Application
    /// @brief Main class
    /// @namespace cae
    ///
    class Application
    {

        public:
            Application(const ArgsConfig &argsConfig, const EnvConfig &envConfig);
            ~Application() = default;

            Application(const Application &) = delete;
            Application &operator=(const Application &) = delete;
            Application(Application &&) = delete;
            Application &operator=(Application &&) = delete;

            void start() const;
            void stop();

        private:
            void setupEngine(const std::string &rendererName, const std::string &windowName);

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

            std::unique_ptr<utl::PluginLoader> m_pluginLoader = nullptr;
            std::unique_ptr<Engine> m_engine = nullptr;

            AppConfig m_appConfig;
    }; // class Application

} // namespace cae
```

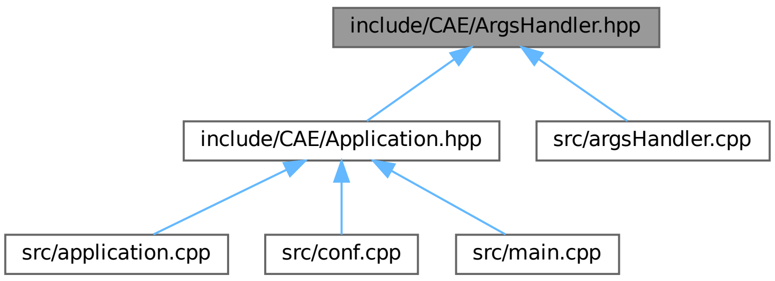
15.4 include/CAE/ArgsHandler.hpp File Reference

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 cae::EnvConfig
- class cae::ArgsHandler

  Class to handle command line arguments.

Namespaces

- namespace cae

15.4.1 Detailed Description

This file contains the ArgsHandler class declaration.

Definition in file ArgsHandler.hpp.

15.5 ArgsHandler.hpp

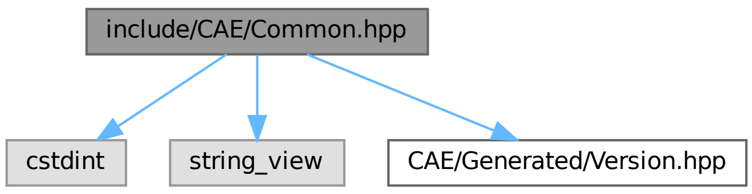
Go to the documentation of this file.

```
///
/// @file ArgsHandler.hpp
/// @brief This file contains the ArgsHandler class declaration
/// @namespace cae
///

#pragma once

#include <string>

namespace cae
{

    struct ArgsConfig
    {
            bool run = false;
            std::string config_path;
    };
    struct EnvConfig
    {
            std::string user_name;
            std::string pwd;
    };

    ///
    /// @class ArgsHandler
    /// @brief Class to handle command line arguments
    /// @namespace cae
    ///
    class ArgsHandler
    {

        public:
            ArgsHandler() = default;
            ~ArgsHandler() = default;

            ArgsHandler(const ArgsHandler &) = delete;
            ArgsHandler &operator=(const ArgsHandler &) = delete;
            ArgsHandler(ArgsHandler &&) = delete;
            ArgsHandler &operator=(ArgsHandler &&) = delete;

            static ArgsConfig ParseArgs(int argc, const char *const *argv);
            static EnvConfig ParseEnv(const char *const *envp);

        private:
    }; // class ArgsHandler

} // namespace cae
```

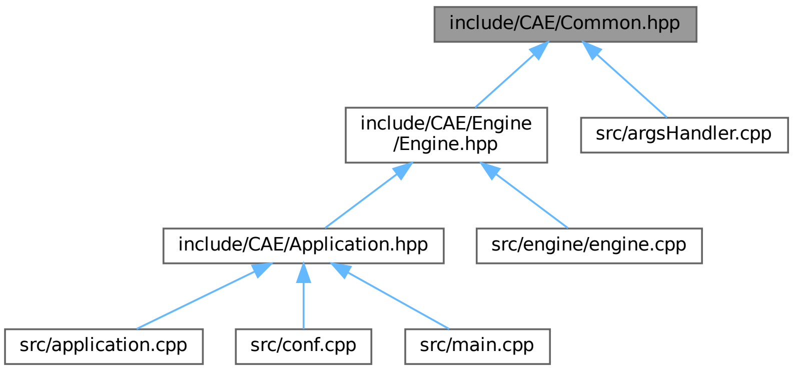
15.6 include/CAE/Common.hpp File Reference

This file contains.

#include <cstdint>
#include <string_view>
#include "CAE/Generated/Version.hpp"
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::Message
- namespace cae::Network
- namespace cae::Plugins
- namespace cae::Plugins::Name
- namespace cae::Renderer
- namespace cae::User
- namespace cae::Window

Macros

- #define APP_EXTENSION ""

Variables

- constexpr auto cae::Audio::VOLUME = 1.F
- constexpr auto cae::Audio::MUTED = false
- static constexpr std::string_view cae::Message::HELP_MSG
- static constexpr std::string_view cae::Message::VERSION_MSG
- constexpr auto cae::Network::HOST = "127.0.0.1"
- constexpr auto cae::Network::PORT = 4242
- constexpr auto cae::Plugins::Name::RENDERER_OPENGL = "OpenGL"
- constexpr auto cae::Plugins::Name::RENDERER_VULKAN = "Vulkan"
- constexpr auto cae::Plugins::Name::WINDOW_GLFW = "GLFW"
- constexpr auto cae::Renderer::VSYNC = false
- constexpr auto cae::Renderer::FRAME_RATE_LIMIT = 90
- constexpr auto cae::User::NAME = "User"
- constexpr uint16_t cae::Window::WIDTH = 1920
- constexpr uint16_t cae::Window::HEIGHT = 1080
- constexpr auto cae::Window::NAME = "CAE - Cross API Engine"
- constexpr auto cae::Window::FULLSCREEN = false

15.6.1 Detailed Description

This file contains.

Definition in file Common.hpp.

15.6.2 Macro Definition Documentation

15.6.2.1 APP_EXTENSION

#define APP_EXTENSION ""

Definition at line 15 of file Common.hpp.

15.7 Common.hpp

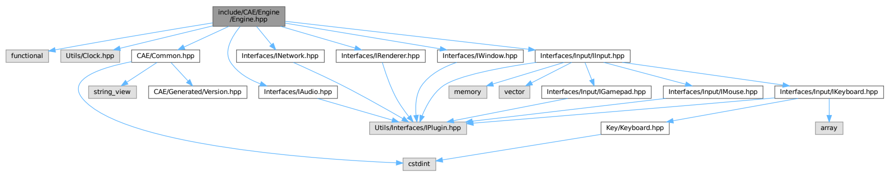
Go to the documentation of this file.

```
00001 ///
00002 /// @file Common.hpp
00003 /// @brief This file contains
00004 /// @namespace cae
00005 ///
00006
00007 #pragma once
00008
00009 #include <cstdint>
00010 #include <string_view>
00011
00012 #ifdef _WIN32
00013 #define APP_EXTENSION ".exe"
00014 #else
00015 #define APP_EXTENSION ""
00016 #endif
00017
00018 #include "CAE/Generated/Version.hpp"
00019
00020 namespace cae
00021 {
00022     namespace Audio
00023     {
00024         inline constexpr auto VOLUME = 1.F;
00025         inline constexpr auto MUTED = false;
00026     } // namespace Audio
00027
00028     namespace Message
00029     {
00030         static constexpr std::string_view HELP_MSG = "Usage: " PROJECT_NAME APP_EXTENSION " [options]\n\n"
00031                                                      "Options:\n"
00032                                                      "  -h, --help              Show this help message\n"
00033                                                      "  -v, --version           Show version information\n"
00034                                                      "  -c, --config <path>     Specify JSON configuration file";
00035         static constexpr std::string_view VERSION_MSG = PROJECT_NAME
00036             " v" PROJECT_VERSION " " BUILD_TYPE " (" GIT_TAG ", commit " GIT_COMMIT_HASH ") "
      __DATE__ " " __TIME__;
00037     } // namespace Message
00038
00039     namespace Network
00040     {
00041         inline constexpr auto HOST = "127.0.0.1";
00042         inline constexpr auto PORT = 4242;
00043     } // namespace Network
00044
00045     namespace Plugins::Name
00046     {
00047         inline constexpr auto RENDERER_OPENGL = "OpenGL";
00048         inline constexpr auto RENDERER_VULKAN = "Vulkan";
00049         inline constexpr auto WINDOW_GLFW = "GLFW";
00050     } // namespace Plugins::Name
00051
00052     namespace Renderer
00053     {
        inline constexpr auto VSYNC = false;
        inline constexpr auto FRAME_RATE_LIMIT = 90;
    } // namespace Renderer

    namespace User
    {
        inline constexpr auto NAME = "User";
    } // namespace User

    namespace Window
    {
        inline constexpr uint16_t WIDTH = 1920;
        inline constexpr uint16_t HEIGHT = 1080;
        inline constexpr auto NAME = "CAE - Cross API Engine";
        inline constexpr auto FULLSCREEN = false;
    } // namespace Window
} // namespace cae
```

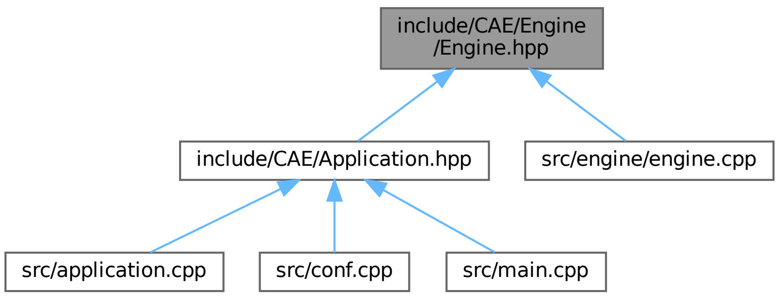
15.8 include/CAE/Engine/Engine.hpp File Reference

This file contains the engine class declaration.

#include <functional>
#include <Utils/Clock.hpp>
#include "CAE/Common.hpp"
#include "Interfaces/IAudio.hpp"
#include "Interfaces/INetwork.hpp"
#include "Interfaces/IRenderer.hpp"
#include "Interfaces/IWindow.hpp"
#include "Interfaces/Input/IInput.hpp"
Include dependency graph for Engine.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- struct cae::EngineConfig
- class cae::Engine

  Engine class.

Namespaces

- namespace cae

15.8.1 Detailed Description

This file contains the engine class declaration.

Definition in file Engine.hpp.

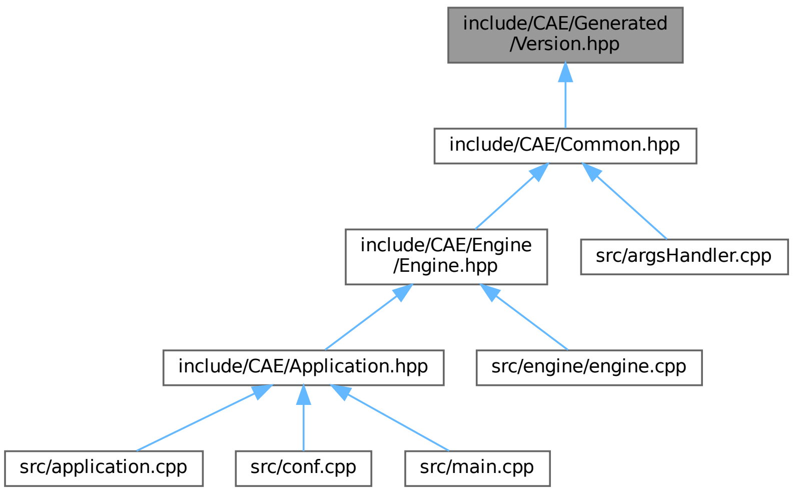
15.9 Engine.hpp

Go to the documentation of this file.

```
///
/// @file Engine.hpp
/// @brief This file contains the engine class declaration
/// @namespace cae
///

#pragma once

#include <functional>

#include <Utils/Clock.hpp>

#include "CAE/Common.hpp"
#include "Interfaces/IAudio.hpp"
#include "Interfaces/INetwork.hpp"
#include "Interfaces/IRenderer.hpp"
#include "Interfaces/IWindow.hpp"
#include "Interfaces/Input/IInput.hpp"

namespace cae
{

    struct EngineConfig
    {
            float audio_master_volume = Audio::VOLUME;
            bool audio_muted = Audio::MUTED;

            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;

            uint16_t window_width = Window::WIDTH;
            uint16_t window_height = Window::HEIGHT;
            bool window_fullscreen = Window::FULLSCREEN;
            std::string window_name = Window::NAME;
    };

    ///
    /// @class Engine
    /// @brief Engine class
    /// @namespace cae
    ///
    class Engine
    {

        public:
            Engine(const EngineConfig &config, const std::function<std::shared_ptr<IAudio>()> &audioFactory,
                   const std::function<std::shared_ptr<IInput>()> &inputFactory,
                   const std::function<std::shared_ptr<INetwork>()> &networkFactory,
                   const std::function<std::shared_ptr<IRenderer>()> &rendererFactory,
                   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;

            [[nodiscard]] const std::shared_ptr<IAudio> &getAudio() const { return m_audioPlugin; }
            [[nodiscard]] const std::shared_ptr<IInput> &getInput() const { return m_inputPlugin; }
            [[nodiscard]] const std::shared_ptr<INetwork> &getNetwork() const { return m_networkPlugin; }
            [[nodiscard]] const std::shared_ptr<IRenderer> &getRenderer() const { return m_rendererPlugin; }
            [[nodiscard]] const std::shared_ptr<IWindow> &getWindow() const { return m_windowPlugin; }

            [[nodiscard]] const std::unique_ptr<utl::Clock> &getClock() { return m_clock; }

            void run() const;
            void stop();

        private:
            std::shared_ptr<IAudio> m_audioPlugin = nullptr;
            std::shared_ptr<IInput> m_inputPlugin = 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;

    }; // class Engine
} // namespace cae
```

15.10 include/CAE/Generated/Version.hpp File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define PROJECT_NAME "cae"
- #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 "9438779"
- #define GIT_TAG "9438779"
- #define BUILD_TYPE "Release"

15.10.1 Macro Definition Documentation

15.10.1.1 BUILD_TYPE

#define BUILD_TYPE "Release"

Definition at line 15 of file Version.hpp.

15.10.1.2 GIT_COMMIT_HASH

#define GIT_COMMIT_HASH "9438779"

Definition at line 13 of file Version.hpp.

15.10.1.3 GIT_TAG

#define GIT_TAG "9438779"

Definition at line 14 of file Version.hpp.

15.10.1.4 PROJECT_NAME

#define PROJECT_NAME "cae"

Definition at line 7 of file Version.hpp.

15.10.1.5 PROJECT_VERSION

#define PROJECT_VERSION "0.0.0"

Definition at line 8 of file Version.hpp.

15.10.1.6 PROJECT_VERSION_MAJOR

#define PROJECT_VERSION_MAJOR "0"

Definition at line 9 of file Version.hpp.

15.10.1.7 PROJECT_VERSION_MINOR

#define PROJECT_VERSION_MINOR "0"

Definition at line 10 of file Version.hpp.

15.10.1.8 PROJECT_VERSION_PATCH

#define PROJECT_VERSION_PATCH "0"

Definition at line 11 of file Version.hpp.

15.11 Version.hpp

Go to the documentation of this file.

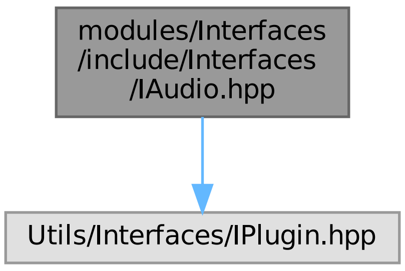
```
#pragma once

// ==========================================================================
// DO NOT EDIT THIS FILE MANUALLY. IT IS GENERATED BY CMAKE DURING THE BUILD PROCESS.
// ==========================================================================

#define PROJECT_NAME "cae"
#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 "9438779"
#define GIT_TAG "9438779"
#define BUILD_TYPE "Release"
```

15.12 LICENSE.md File Reference

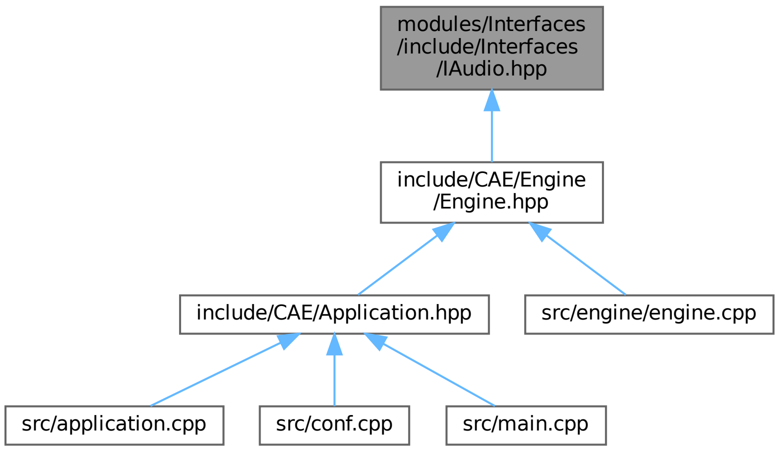
15.13 modules/Interfaces/include/Interfaces/IAudio.hpp File Reference

This file contains the audio interface.

#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IAudio.hpp:


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

This graph shows which files directly or indirectly include this file:


modules/Interfaces /include/Interfaces /IAudio.hpp
include/CAE/Engine /Engine.hpp
include/CAE/Application.hpp
src/engine/engine.cpp
src/application.cpp
src/conf.cpp
src/main.cpp

Classes

- interface cae::IAudio

    Interface for audio.

Namespaces

- namespace cae

15.13.1 Detailed Description

This file contains the audio interface.

Definition in file IAudio.hpp.

15.14 IAudio.hpp

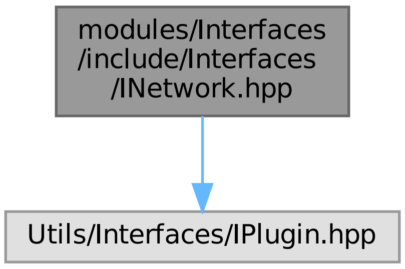
Go to the documentation of this file.

```
///
/// @file IAudio.hpp
/// @brief This file contains the audio interface
/// @namespace cae
///

#pragma once

#include "Utils/Interfaces/IPlugin.hpp"

namespace cae
{

    ///
    /// @interface IAudio
    /// @brief Interface for audio
    /// @namespace cae
    ///
    class IAudio : public utl::IPlugin
    {

        public:
            ~IAudio() override = default;

    }; // interface IAudio

} // namespace cae
```

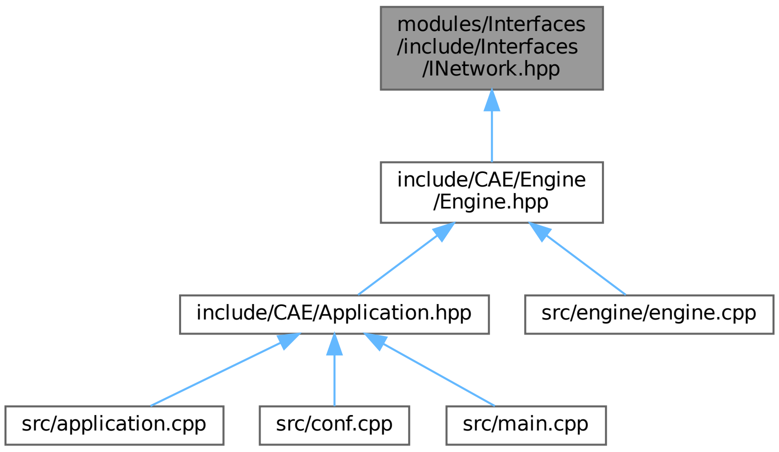
15.15 modules/Interfaces/include/Interfaces/INetwork.hpp File Reference

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

15.15.1 Detailed Description

This file contains the network interface.

Definition in file INetwork.hpp.

15.16 INetwork.hpp

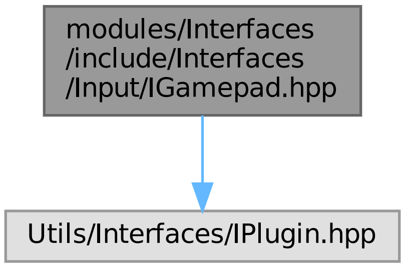
Go to the documentation of this file.

```
///
/// @file INetwork.hpp
/// @brief This file contains the network interface
/// @namespace cae
///

#pragma once

#include "Utils/Interfaces/IPlugin.hpp"

namespace cae
{

    ///
    /// @interface INetwork
    /// @brief Interface for network
    /// @namespace cae
    ///
    class INetwork : public utl::IPlugin
    {

        public:
            ~INetwork() override = default;

            virtual bool connect(const std::string &host, uint16_t port) = 0;

    }; // interface INetwork

} // namespace cae
```

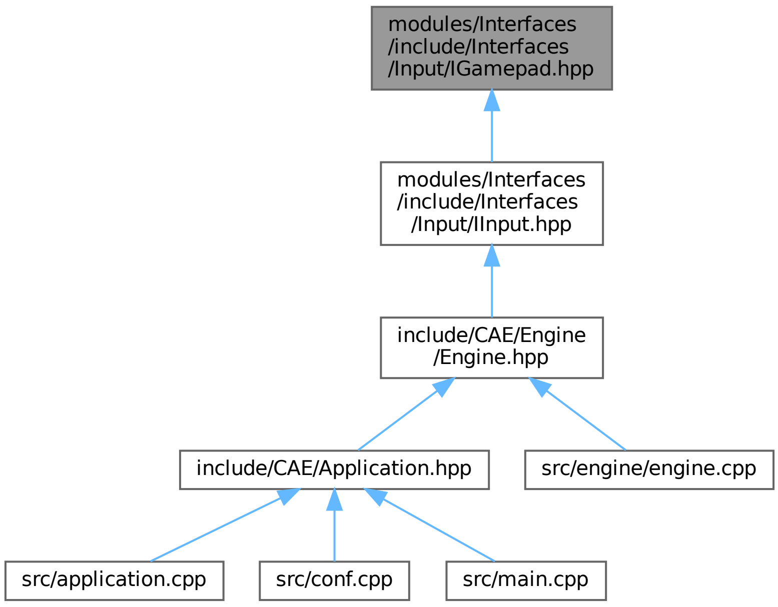
15.17 modules/Interfaces/include/Interfaces/Input/IGamepad.hpp File Reference

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

15.17.1 Detailed Description

This file contains the input gamepad interface.

Definition in file IGamepad.hpp.

15.18 IGamepad.hpp

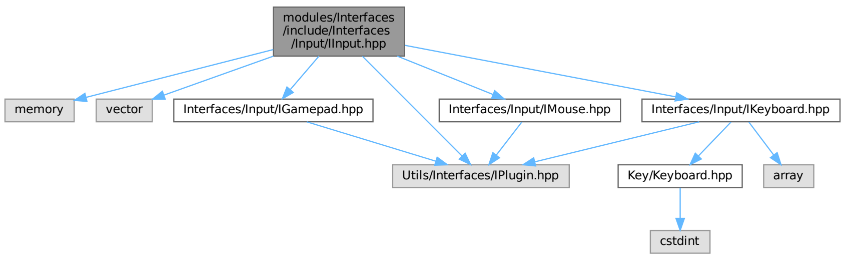
Go to the documentation of this file.

```
///
/// @file IGamepad.hpp
/// @brief This file contains the input gamepad interface
/// @namespace cae
///

#pragma once

#include "Utils/Interfaces/IPlugin.hpp"

namespace cae
{

    ///
    /// @interface IGamepad
    /// @brief Interface for gamepad
    /// @namespace cae
    ///
    class IGamepad : public utl::IPlugin
    {

        public:
            ~IGamepad() override = default;

    }; // interface IGamepad

} // namespace cae
```

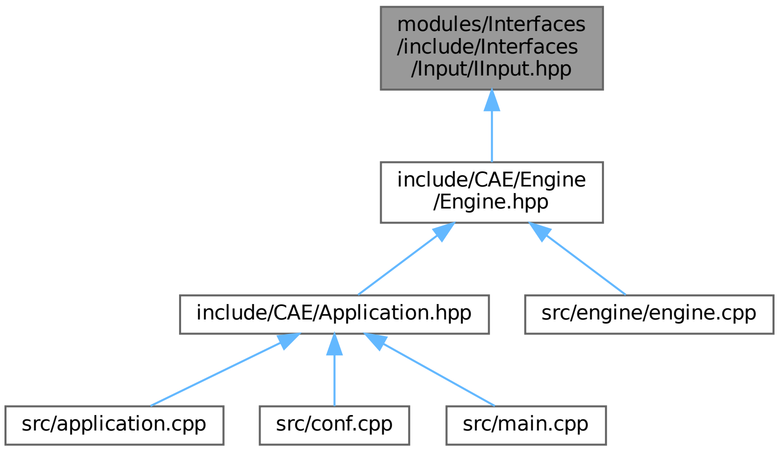
15.19 modules/Interfaces/include/Interfaces/Input/IInput.hpp File Reference

This file contains the input interface.

#include <memory>
#include <vector>
#include "Interfaces/Input/IGamepad.hpp"
#include "Interfaces/Input/IKeyboard.hpp"
#include "Interfaces/Input/IMouse.hpp"
#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IInput.hpp:


modules/Interfaces/include/Interfaces/Input/IInput.hpp
memory
vector
Interfaces/Input/IGamepad.hpp
Interfaces/Input/IMouse.hpp
Interfaces/Input/IKeyboard.hpp
Utils/Interfaces/IPlugin.hpp
Key/Keyboard.hpp
array
cstdint

This graph shows which files directly or indirectly include this file:


modules/Interfaces/include/Interfaces/Input/IInput.hpp
include/CAE/Engine/Engine.hpp
include/CAE/Application.hpp
src/engine/engine.cpp
src/application.cpp
src/conf.cpp
src/main.cpp

Classes

- interface cae::IInput

  Interface for audio.

Namespaces

- namespace cae

15.19.1 Detailed Description

This file contains the input interface.

Definition in file IInput.hpp.

15.20 IInput.hpp

Go to the documentation of this file.

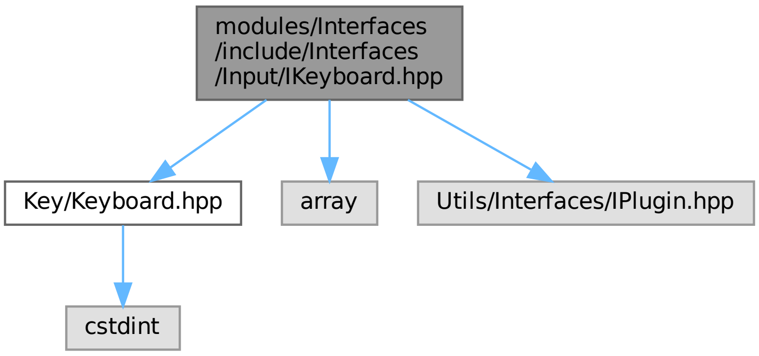
```
///
/// @file IInput.hpp
/// @brief This file contains the input interface
/// @namespace cae
///

#pragma once

#include <memory>
#include <vector>

#include "Interfaces/Input/IGamepad.hpp"
#include "Interfaces/Input/IKeyboard.hpp"
#include "Interfaces/Input/IMouse.hpp"
#include "Utils/Interfaces/IPlugin.hpp"

```

```
namespace cae
{

    ///
    /// @interface IInput
    /// @brief Interface for audio
    /// @namespace cae
    ///
    class IInput : public utl::IPlugin
    {

        public:
            ~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;

        private:
            std::unique_ptr<IKeyboard> m_keyboard;
            std::unique_ptr<IMouse> m_mouse;
            std::vector<std::unique_ptr<IGamepad>> m_gamepads;
    }; // interface IInput

} // namespace cae
```

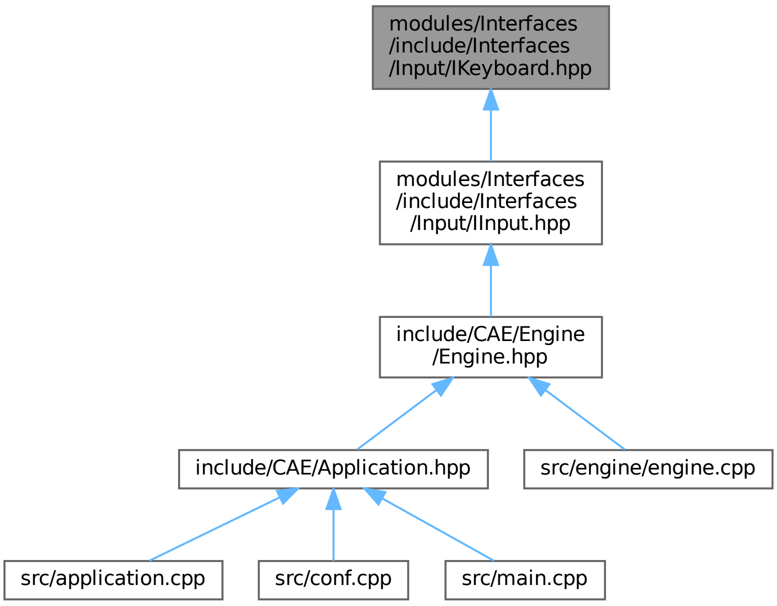
15.21 modules/Interfaces/include/Interfaces/Input/IKeyboard.hpp File Reference

This file contains the input keyboard interface.

#include ”Key/Keyboard.hpp”
#include <array>
#include ”Utils/Interfaces/IPlugin.hpp”
Include dependency graph for IKeyboard.hpp:



This graph shows which files directly or indirectly include this file:


modules/Interfaces /include/Interfaces /Input/IKeyboard.hpp
modules/Interfaces /include/Interfaces /Input/IInput.hpp
include/CAE/Engine /Engine.hpp
include/CAE/Application.hpp
src/engine/engine.cpp
src/application.cpp
src/conf.cpp
src/main.cpp

Classes

- interface cae::IKeyboard

  Interface for keyboard.

Namespaces

- namespace cae

15.21.1 Detailed Description

This file contains the input keyboard interface.

Definition in file IKeyboard.hpp.

15.22 IKeyboard.hpp

Go to the documentation of this file.

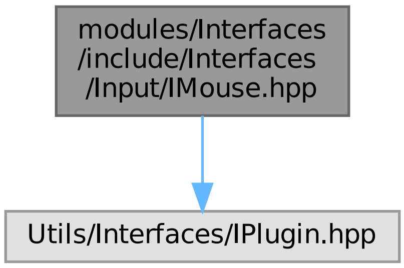
```
///
/// @file IKeyboard.hpp
/// @brief This file contains the input keyboard interface
/// @namespace cae
///

#pragma once

#include "Key/Keyboard.hpp"
```

```

#include <array>

#include "Utils/Interfaces/IPlugin.hpp"

namespace cae
{

    ///
    /// @interface IKeyboard
    /// @brief Interface for keyboard
    /// @namespace cae
    ///
    class IKeyboard : public utl::IPlugin
    {

        public:
            ~IKeyboard() override = default;

            virtual bool isKeyPressed(KeyCode keyCode) const = 0;

        private:
            std::array<KeyState, static_cast<size_t>(KeyCode::Count)> m_keyMap{};
    }; // interface IKeyboard

} // namespace cae
```

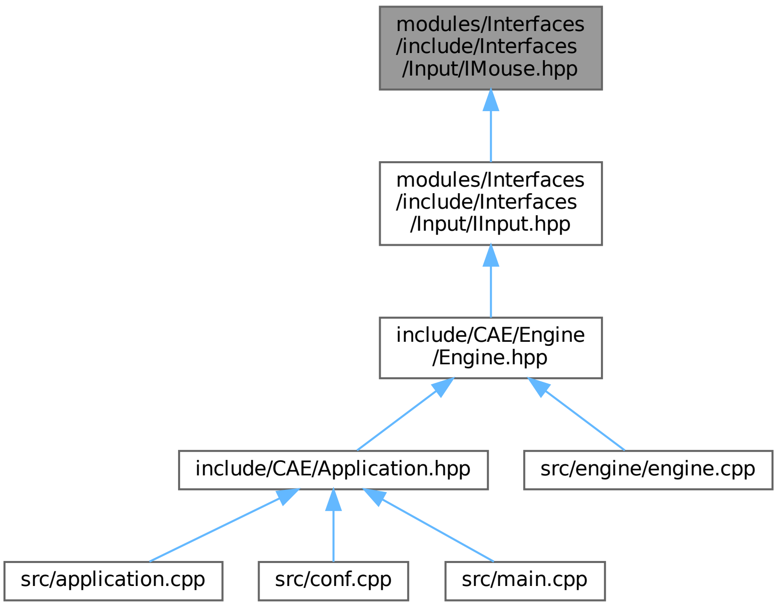
15.23 modules/Interfaces/include/Interfaces/Input/IMouse.hpp File Reference

This file contains the input mouse interface.

#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IMouse.hpp:


modules/Interfaces/include/Interfaces/Input/IMouse.hpp
Utils/Interfaces/IPlugin.hpp

This graph shows which files directly or indirectly include this file:


modules/Interfaces /include/Interfaces /Input/IMouse.hpp
modules/Interfaces /include/Interfaces /Input/IInput.hpp
include/CAE/Engine /Engine.hpp
include/CAE/Application.hpp
src/engine/engine.cpp
src/application.cpp
src/conf.cpp
src/main.cpp

Classes

- interface cae::IMouse

    Interface for mouse.

Namespaces

- namespace cae

15.23.1 Detailed Description

This file contains the input mouse interface.

Definition in file IMouse.hpp.

15.24 IMouse.hpp

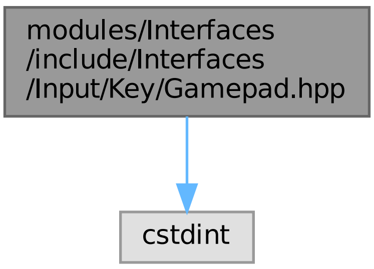
Go to the documentation of this file.

```
///
/// @file IMouse.hpp
/// @brief This file contains the input mouse interface
/// @namespace cae
///

#pragma once

#include "Utils/Interfaces/IPlugin.hpp"
```

```

namespace cae
{

    ///
    /// @interface IMouse
    /// @brief Interface for mouse
    /// @namespace cae
    ///
    class IMouse : public utl::IPlugin
    {

        public:
            ~IMouse() override = default;

    }; // interface IMouse

} // namespace cae
```

15.25 modules/Interfaces/include/Interfaces/Input/Key/Gamepad.hpp File Reference

#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 }

15.26 Gamepad.hpp

Go to the documentation of this file.

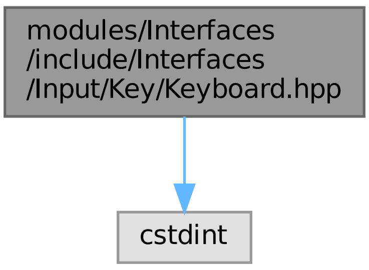
```
///
/// @file Mouse.hpp
/// @brief This file contains the gamepad keys
/// @namespace cae
///

#pragma once

#include <cstdint>

namespace cae
{
    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
    };
} // namespace cae
```

15.27 modules/Interfaces/include/Interfaces/Input/Key/Keyboard.hpp File Reference

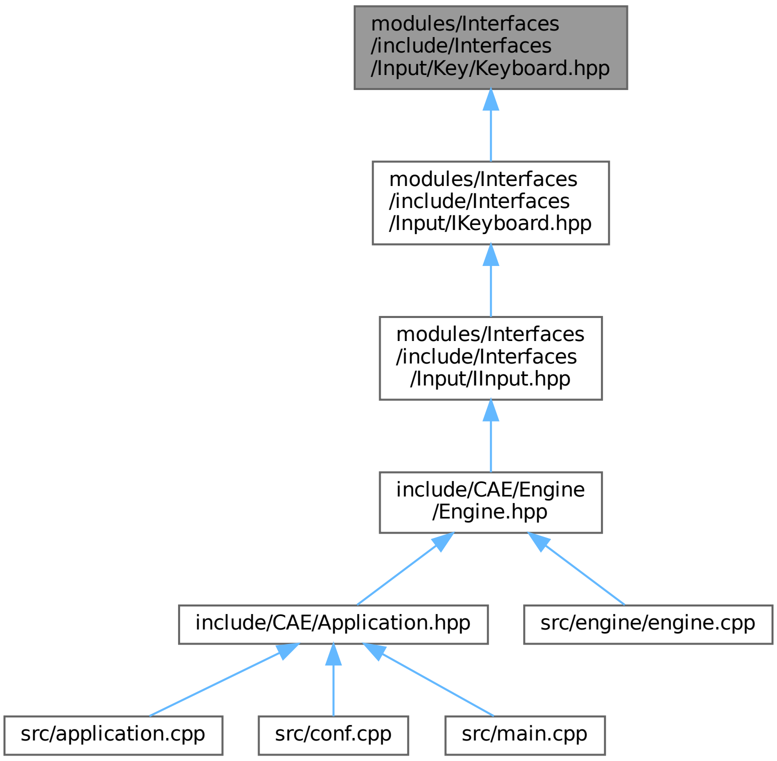
This file contains the keyboard keys.

#include <cstdint>

Include dependency graph for Keyboard.hpp:



This graph shows which files directly or indirectly include this file:


modules/Interfaces /include/Interfaces /Input/Key/Keyboard.hpp
modules/Interfaces /include/Interfaces /Input/IKeyboard.hpp
modules/Interfaces /include/Interfaces /Input/IInput.hpp
include/CAE/Engine /Engine.hpp
include/CAE/Application.hpp
src/engine/engine.cpp
src/application.cpp
src/conf.cpp
src/main.cpp

Namespaces

- namespace cae

Enumerations

- enum 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::Count }

15.27.1 Detailed Description

This file contains the keyboard keys.

Definition in file Keyboard.hpp.

15.28 Keyboard.hpp

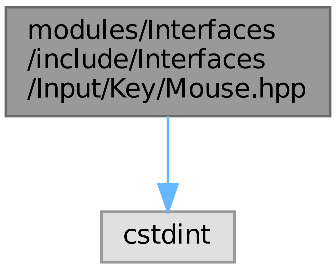
Go to the documentation of this file.

```
///
/// @file Keyboard.hpp
/// @brief This file contains the keyboard keys
/// @namespace cae
///

#pragma once

#include <cstdint>

namespace cae
{

    enum 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,

        Count
    };

} // namespace cae
```

15.29 modules/Interfaces/include/Interfaces/Input/Key/Mouse.hpp File Reference

This file contains the gamepad keys.

#include <cstdint>
Include dependency graph for Mouse.hpp:



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 }

15.29.1 Detailed Description

This file contains the gamepad keys.

This file contains the mouse keys.

Definition in file Mouse.hpp.

15.30 Mouse.hpp

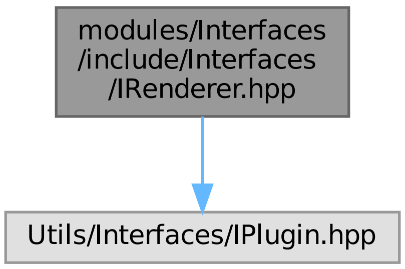
Go to the documentation of this file.

```
///
/// @file Mouse.hpp
/// @brief This file contains the mouse keys
/// @namespace cae
///

#pragma once

#include <cstdint>

namespace cae
{
    enum class MouseButton : uint8_t
    {
        Left = 0,
        Right,
        Middle,
        XButton1,
        XButton2,
        WheelUp,
        WheelDown
    };
} // namespace cae
```

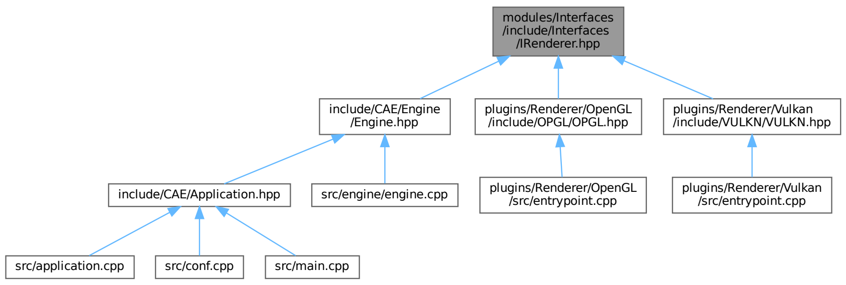
15.31 modules/Interfaces/include/Interfaces/IRenderer.hpp File Reference

This file contains the Renderer interface.

#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IRenderer.hpp:


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

This graph shows which files directly or indirectly include this file:


modules/Interfaces/include/Interfaces/IRenderer.hpp
include/CAE/Engine/Engine.hpp
plugins/Renderer/OpenGL/include/OPGL/OPGL.hpp
plugins/Renderer/Vulkan/include/VULKN/VULKN.hpp
include/CAE/Application.hpp
src/engine/engine.cpp
plugins/Renderer/OpenGL/src/entrypoint.cpp
plugins/Renderer/Vulkan/src/entrypoint.cpp
src/application.cpp
src/conf.cpp
src/main.cpp

Classes

- interface cae::IRenderer
  Interface for renderer.

Namespaces

- namespace cae

15.31.1 Detailed Description

This file contains the Renderer interface.

Definition in file IRenderer.hpp.

15.32 IRenderer.hpp

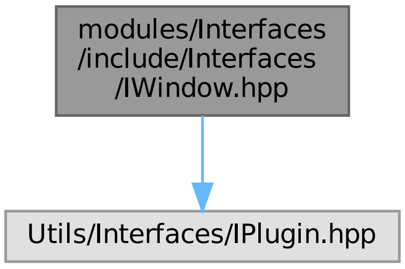
Go to the documentation of this file.

```
///
/// @file IRenderer.hpp
/// @brief This file contains the Renderer interface
/// @namespace cae
///

#pragma once

#include "Utils/Interfaces/IPlugin.hpp"

namespace cae
{

    ///
    /// @interface IRenderer
    /// @brief Interface for renderer
    /// @namespace cae
    ///
    class IRenderer : public utl::IPlugin
    {

        public:
            ~IRenderer() override = default;

            virtual void initialize(void *nativeWindowHandle) = 0;
    }; // interface IRenderer

} // namespace cae
```

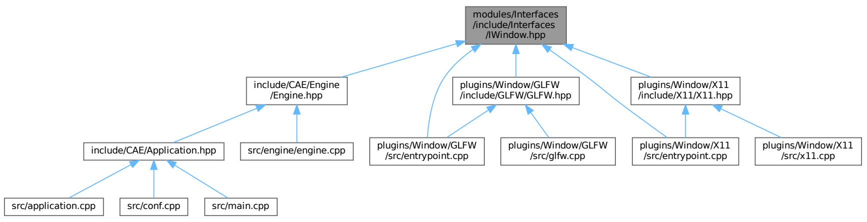
15.33 modules/Interfaces/include/Interfaces/IWindow.hpp File Reference

This file contains the Window interface.

#include "Utils/Interfaces/IPlugin.hpp"
Include dependency graph for IWindow.hpp:


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

This graph shows which files directly or indirectly include this file:


modules/Interfaces /include/Interfaces /IWindow.hpp
include/CAE/Engine /Engine.hpp
plugins/Window/GLFW /include/GLFW/GLFW.hpp
plugins/Window/X11 /include/X11/X11.hpp
include/CAE/Application.hpp
src/engine/engine.cpp
plugins/Window/GLFW /src/entrypoint.cpp
plugins/Window/GLFW /src/glfw.cpp
plugins/Window/X11 /src/entrypoint.cpp
plugins/Window/X11 /src/x11.cpp
src/application.cpp
src/conf.cpp
src/main.cpp

Classes

- struct cae::WindowSize
- struct cae::NativeWindowHandle
- interface cae::IWindow

  Interface for window.

Namespaces

- namespace cae

15.33.1 Detailed Description

This file contains the Window interface.

Definition in file IWindow.hpp.

15.34 IWindow.hpp

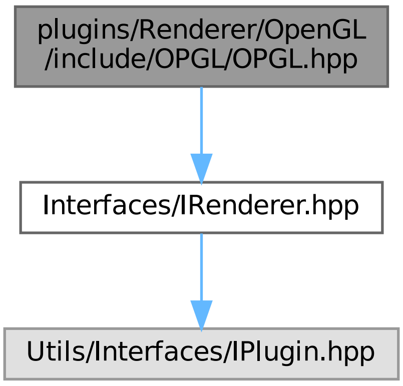
Go to the documentation of this file.

```
///
/// @file IWindow.hpp
/// @brief This file contains the Window interface
/// @namespace cae
///

#pragma once

#include "Utils/Interfaces/IPlugin.hpp"

namespace cae
{

    struct WindowSize
    {
        uint16_t width;
        uint16_t height;
    };

    struct NativeWindowHandle
    {
        void *window;
        void *display;
    };

    ///
    /// @interface IWindow
    /// @brief Interface for window
    /// @namespace cae
    ///
    class IWindow : public utl::IPlugin
    {

        public:
        ~IWindow() override = default;

        virtual bool create(const std::string &name, WindowSize size) = 0;
        virtual void close() = 0;

        virtual NativeWindowHandle getNativeHandle() const = 0;
        virtual WindowSize getWindowSize() const = 0;

        virtual bool setIcon(const std::string &path) const = 0;

        virtual bool shouldClose() const = 0;
        virtual void pollEvents() = 0;

        virtual bool wasResized() const = 0;
        virtual void resetResizedFlag() = 0;

        private:
    }; // interface IWindow

} // namespace cae
```

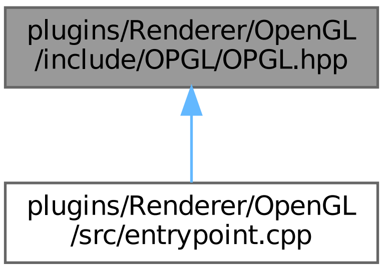
15.35 plugins/Renderer/OpenGL/include/OPGL/OPGL.hpp File Reference

This file contains the OPGL class declaration.

#include "Interfaces/IRenderer.hpp"
Include dependency graph for OPGL.hpp:


plugins/Renderer/OpenGL /include/OPGL/OPGL.hpp
Interfaces/IRenderer.hpp
Utils/Interfaces/IPlugin.hpp

This graph shows which files directly or indirectly include this file:


plugins/Renderer/OpenGL /include/OPGL/OPGL.hpp
plugins/Renderer/OpenGL /src/entrypoint.cpp

Classes

- class cae::OPGL

  Class for the OpenGL plugin.

Namespaces

- namespace cae

15.35.1 Detailed Description

This file contains the OPGL class declaration.

Definition in file OPGL.hpp.

15.36 OPGL.hpp

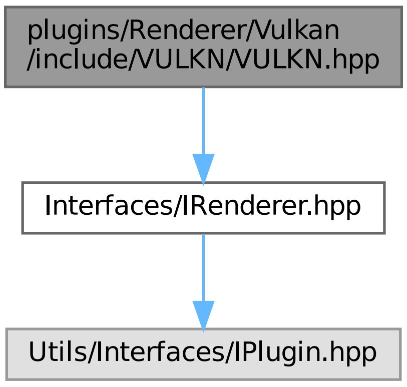
Go to the documentation of this file.

```
///
/// @file OPGL.hpp
/// @brief This file contains the OPGL class declaration
/// @namespace cae
///

#pragma once

#include "Interfaces/IRenderer.hpp"

namespace cae
{

    ///
    /// @class OPGL
    /// @brief Class for the OpenGL plugin
    /// @namespace cae
    ///
    class OPGL final : public IRenderer
    {

        public:
            OPGL() = default;
            ~OPGL() override = default;

            OPGL(const OPGL &) = delete;
            OPGL &operator=(const OPGL &) = delete;
            OPGL(OPGL &&) = delete;
            OPGL &operator=(OPGL &&) = delete;

            [[nodiscard]] std::string getName() const override { return "OpenGL"; }
            [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::RENDERER; }
            [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::ALL; }

            void initialize(void *nativeWindowHandle) override {}
    }; // class OPGL
} // namespace cae
```

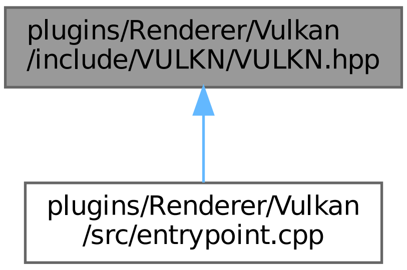
15.37 plugins/Renderer/Vulkan/include/VULKN/VULKN.hpp File Reference

This file contains the VULKN class declaration.

#include "Interfaces/IRenderer.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

15.37.1 Detailed Description

This file contains the VULKN class declaration.

Definition in file VULKN.hpp.

15.38 VULKN.hpp

Go to the documentation of this file.

```
///
/// @file VULKN.hpp
/// @brief This file contains the VULKN class declaration
/// @namespace cae
///

#pragma once

#include "Interfaces/IRenderer.hpp"

namespace cae
{

    ///
    /// @class VULKN
    /// @brief Class for the Vulkan plugin
    /// @namespace cae
    ///
    class VULKN final : public IRenderer
    {

        public:
            VULKN() = default;
            ~VULKN() override = default;

            VULKN(const VULKN &) = delete;
            VULKN &operator=(const VULKN &) = delete;
            VULKN(VULKN &&) = delete;
            VULKN &operator=(VULKN &&) = delete;

            [[nodiscard]] std::string getName() const override { return "Vulkan"; }
            [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::RENDERER; }
            [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::ALL; }

            void initialize(void *nativeWindowHandle) override {}
    }; // class VULKN
} // namespace cae
```

15.39 plugins/Renderer/Vulkan/src/VULKN.cpp File Reference

15.40 VULKN.cpp

Go to the documentation of this file.

```

```

15.41 plugins/Audio/ALSA/src/entrypoint.cpp File Reference

15.42 entrypoint.cpp

Go to the documentation of this file.
00001

15.43 plugins/Audio/Core/src/entrypoint.cpp File Reference

15.44 entrypoint.cpp

Go to the documentation of this file.
00001

15.45 plugins/Audio/OpenAL/src/entrypoint.cpp File Reference

15.46 entrypoint.cpp

Go to the documentation of this file.
00001

15.47 plugins/Audio/Pulse/src/entrypoint.cpp File Reference

15.48 entrypoint.cpp

Go to the documentation of this file.
00001

15.49 plugins/Audio/XAudio2/src/entrypoint.cpp File Reference

15.50 entrypoint.cpp

Go to the documentation of this file.
00001

15.51 plugins/Input/Cocoa/src/entrypoint.cpp File Reference

15.52 entrypoint.cpp

Go to the documentation of this file.
00001

15.53 plugins/Input/Win32/src/entrypoint.cpp File Reference

15.54 entrypoint.cpp

Go to the documentation of this file.

```
00001
```

15.55 plugins/Input/X11/src/entrypoint.cpp File Reference

15.56 entrypoint.cpp

Go to the documentation of this file.

```
00001
```

15.57 plugins/Network/Asio/src/entrypoint.cpp File Reference

15.58 entrypoint.cpp

Go to the documentation of this file.

```
00001
```

15.59 plugins/Network/Posix/src/entrypoint.cpp File Reference

15.60 entrypoint.cpp

Go to the documentation of this file.

```
00001
```

15.61 plugins/Network/WinSock/src/entrypoint.cpp File Reference

15.62 entrypoint.cpp

Go to the documentation of this file.

```
00001
```

15.63 plugins/Renderer/DirectX12/src/entrypoint.cpp File Reference

15.64 entrypoint.cpp

Go to the documentation of this file.

```
00001
```

15.65 plugins/Renderer/Metal/src/entrypoint.cpp File Reference

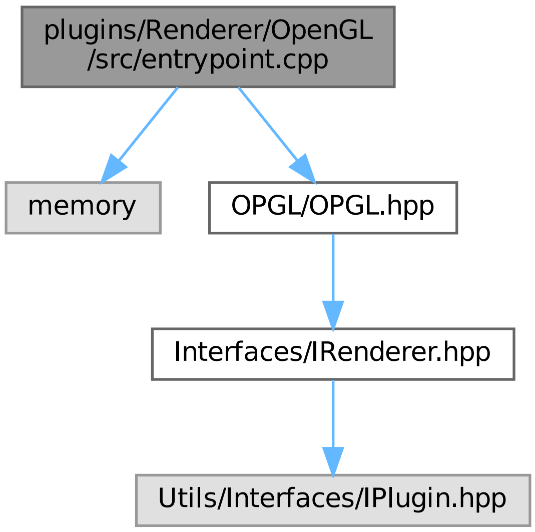
15.66 entrypoint.cpp

Go to the documentation of this file.

```
00001
```

15.67 plugins/Renderer/OpenGL/src/entrypoint.cpp File Reference

#include <memory>
#include "OPGL/OPGL.hpp"
Include dependency graph for entrypoint.cpp:


plugins/Renderer/OpenGL /src/entrypoint.cpp
memory
OPGL/OPGL.hpp
Interfaces/IRenderer.hpp
Utils/Interfaces/IPlugin.hpp

Functions

- cae::IRenderer * entryPoint ()

15.67.1 Function Documentation

15.67.1.1 entryPoint()

cae::IRenderer * entryPoint ()

Definition at line 7 of file entrypoint.cpp.

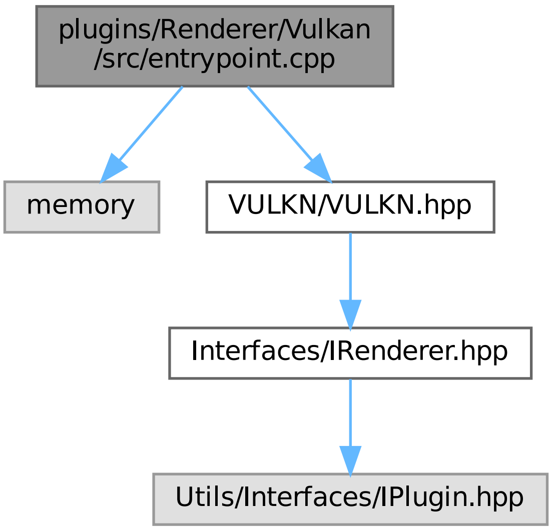
15.68 entrypoint.cpp

Go to the documentation of this file.

```
#include <memory>

#include "OPGL/OPGL.hpp"

extern "C"
{
    cae::IRenderer *entryPoint() { return std::make_unique<cae::OPGL>().release(); }
}
```

15.69 plugins/Renderer/Vulkan/src/entrypoint.cpp File Reference

#include <memory>
#include "VULKN/VULKN.hpp"
Include dependency graph for entrypoint.cpp:



Functions

- cae::IRenderer * entryPoint ()

15.69.1 Function Documentation

15.69.1.1 entryPoint()

cae::IRenderer * entryPoint ()

Definition at line 7 of file entrypoint.cpp.

15.70 entrypoint.cpp

Go to the documentation of this file.

```
#include <memory>

#include "VULKN/VULKN.hpp"

extern "C"
{
    cae::IRenderer *entryPoint() { return std::make_unique<cae::VULKN>().release(); }
}
```

15.71 plugins/Window/Cocoa/src/entrypoint.cpp File Reference

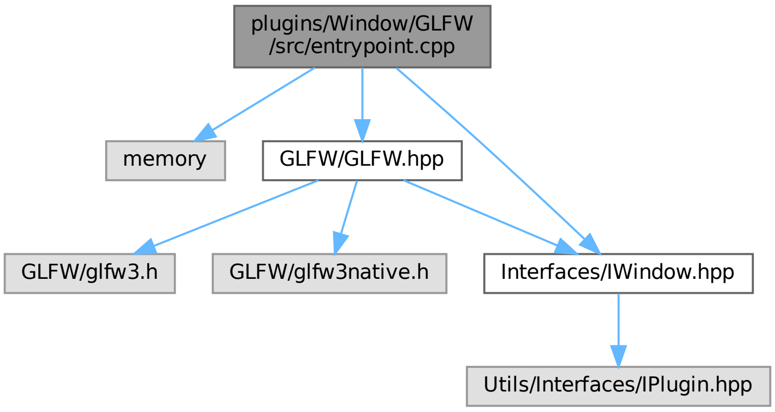
15.72 entrypoint.cpp

Go to the documentation of this file.

```
```

15.73 plugins/Window/GLFW/src/entrypoint.cpp File Reference

#include <memory>
#include "GLFW/GLFW.hpp"
#include "Interfaces/IWindow.hpp"
Include dependency graph for entrypoint.cpp:



Functions

- cae::IWindow * entryPoint ()

15.73.1 Function Documentation

15.73.1.1 entryPoint()

cae::IWindow ∗ entryPoint ()

Definition at line 8 of file entrypoint.cpp.

15.74 entrypoint.cpp

Go to the documentation of this file.

```
#include <memory>

#include "GLFW/GLFW.hpp"
#include "Interfaces/IWindow.hpp"

extern "C"
{
    cae::IWindow *entryPoint() { return std::make_unique<cae::GLFW>().release(); }
}
```

15.75 plugins/Window/Win32/src/entrypoint.cpp File Reference

15.76 entrypoint.cpp

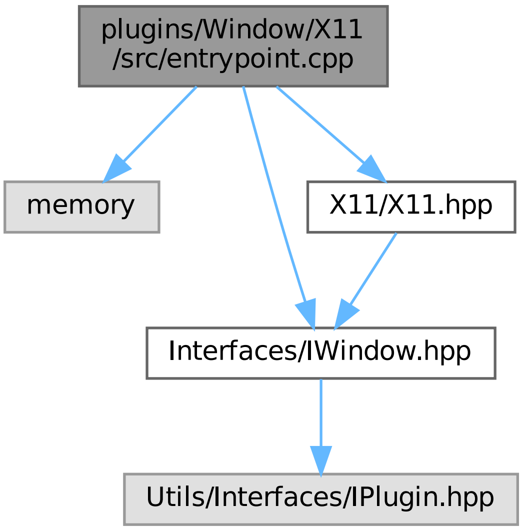
Go to the documentation of this file.

```

```

15.77 plugins/Window/X11/src/entrypoint.cpp File Reference

#include <memory>
#include "Interfaces/IWindow.hpp"
#include "X11/X11.hpp"

Include dependency graph for entrypoint.cpp:



Functions

- cae::IWindow ∗ entryPoint ()

15.77.1 Function Documentation

15.77.1.1 entryPoint()

cae::IWindow ∗ entryPoint ()

Definition at line 8 of file entrypoint.cpp.

15.78 entrypoint.cpp

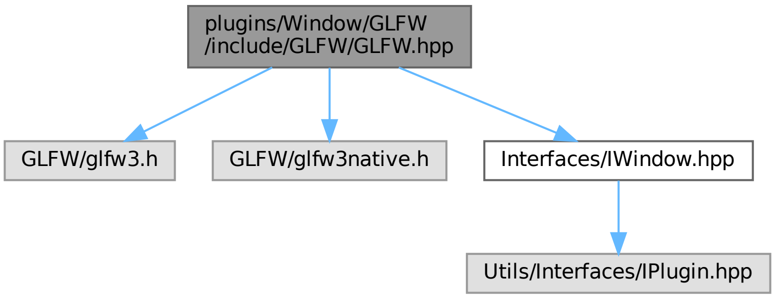
Go to the documentation of this file.

```
#include <memory>

#include "Interfaces/IWindow.hpp"
#include "X11/X11.hpp"

extern "C"
{
    cae::IWindow *entryPoint() { return std::make_unique<cae::X11>().release(); }
}
```

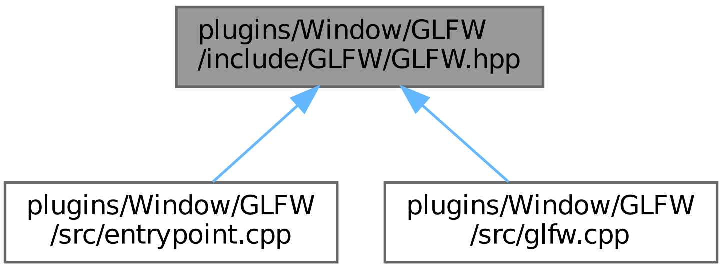
15.79 plugins/Window/GLFW/include/GLFW/GLFW.hpp File Reference

This file contains the GLFW class declaration.

#include <GLFW/glfw3.h>
#include <GLFW/glfw3native.h>
#include "Interfaces/IWindow.hpp"
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

15.79.1 Detailed Description

This file contains the GLFW class declaration.

Definition in file GLFW.hpp.

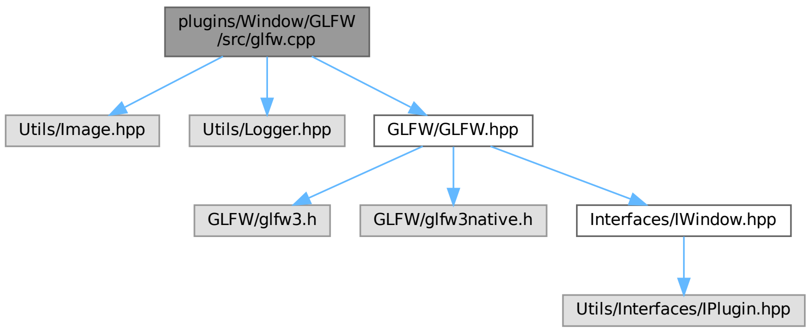
15.80 GLFW.hpp

Go to the documentation of this file.

```cpp
///
/// @file GLFW.hpp
/// @brief This file contains the GLFW class declaration
/// @namespace cae
///

#pragma once

#include <GLFW/glfw3.h>

#if defined(_WIN32)
#define GLFW_EXPOSE_NATIVE_WIN32
#elif defined(__linux__)
#define GLFW_EXPOSE_NATIVE_X11
#elif defined(__APPLE__)
#define GLFW_EXPOSE_NATIVE_COCOA
#endif
#include <GLFW/glfw3native.h>

#include "Interfaces/IWindow.hpp"

namespace cae
{

    ///
    /// @class GLFW
    /// @brief Class for the GLFW plugin
    /// @namespace cae
    ///
    class GLFW final : public IWindow
    {

        public:
            GLFW() = default;
            ~GLFW() override = default;

            GLFW(const GLFW &) = delete;
            GLFW &operator=(const GLFW &) = delete;
            GLFW(GLFW &&) = delete;
            GLFW &operator=(GLFW &&) = delete;

            [[nodiscard]] std::string getName() const override { return "GLFW"; }
            [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::WINDOW; }
            [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::ALL; }

            bool create(const std::string &name, WindowSize size) override;
            void close() override;

            [[nodiscard]] NativeWindowHandle getNativeHandle() const override;
            [[nodiscard]] WindowSize getWindowSize() const override;

            [[nodiscard]] bool setIcon(const std::string &path) const override;

            [[nodiscard]] bool shouldClose() const override { return glfwWindowShouldClose(m_window) != 0; }
            void pollEvents() override { glfwPollEvents(); }

            bool wasResized() const override { return m_frameBufferResized; }
            void resetResizedFlag() override { m_frameBufferResized = false; }

        private:
            static void frameBufferResizeCallback(GLFWwindow *window, int width, int height);

            GLFWwindow *m_window = nullptr;
            WindowSize m_frameBufferSize;
            bool m_frameBufferResized = false;

    }; // class GLFW
} // namespace cae
```

15.81 plugins/Window/GLFW/src/glfw.cpp File Reference

#include <Utils/Image.hpp>
#include <Utils/Logger.hpp>
#include "GLFW/GLFW.hpp"
Include dependency graph for glfw.cpp:



15.82 glfw.cpp

Go to the documentation of this file.

```
#include <Utils/Image.hpp>
#include <Utils/Logger.hpp>

#include "GLFW/GLFW.hpp"

void cae::GLFW::frameBufferResizeCallback(GLFWwindow *window, int width, int height)
{
    auto *const self = static_cast<GLFW *>(glfwGetWindowUserPointer(window));
    self->m_frameBufferResized = true;
    self->m_frameBufferSize = {static_cast<uint16_t>(width), static_cast<uint16_t>(height)};
}

bool cae::GLFW::create(const std::string &name, const WindowSize size)
{
    m_window = nullptr;
    if (glfwInit() == 0)
    {
        return false;
    }

    glfwWindowHint(GLFW_CLIENT_API, GLFW_NO_API);
    glfwWindowHint(GLFW_RESIZABLE, GLFW_TRUE);
    m_window = glfwCreateWindow(size.width, size.height, name.c_str(), nullptr, nullptr);
    if (m_window == nullptr)
    {
        glfwTerminate();
        utl::Logger::log("Failed to create GLFW window", utl::LogLevel::WARNING);

        return false;
    }
    glfwSetWindowUserPointer(m_window, this);
    glfwSetFramebufferSizeCallback(m_window, frameBufferResizeCallback);

    return true;
}

void cae::GLFW::close()
{
    if (m_window)
    {
        glfwDestroyWindow(m_window);
        m_window = nullptr;
    }
    glfwTerminate();
}

cae::WindowSize cae::GLFW::getWindowSize() const
{
    int width = 0;
    int height = 0;
    glfwGetWindowSize(m_window, &width, &height);
    return {.width = static_cast<uint16_t>(width), .height = static_cast<uint16_t>(height)};
}

cae::NativeWindowHandle cae::GLFW::getNativeHandle() const
{
    NativeWindowHandle handle{};
#if defined(_WIN32)
    handle.window = glfwGetWin32Window(m_glfwWindow);
    handle.display = GetModuleHandle(nullptr);
#elif defined(__linux__)
    handle.window = (void *)(uintptr_t)glfwGetX11Window(m_window);
    handle.display = glfwGetX11Display();
#elif defined(__APPLE__)
    handle.window = glfwGetCocoaWindow(m_glfwWindow);
    handle.display = nullptr;
#endif
    return handle;
}

bool cae::GLFW::setIcon(const std::string &path) const
{
    static const utl::Image image(path);
    if (image.pixels == nullptr)
    {
        return false;
    }
    static const GLFWimage appIcon{.width = image.width, .height = image.height, .pixels = image.pixels};
    glfwSetWindowIcon(m_window, 1, &appIcon);
    return true;
}
```

15.83 plugins/Audio/README.md File Reference

15.84 plugins/Input/README.md File Reference

15.85 plugins/Network/README.md File Reference

15.86 plugins/Renderer/README.md File Reference

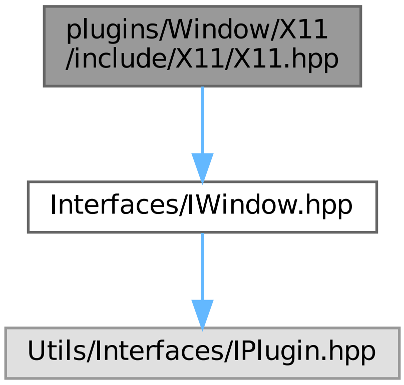
15.87 plugins/Window/README.md File Reference

15.88 README.md File Reference

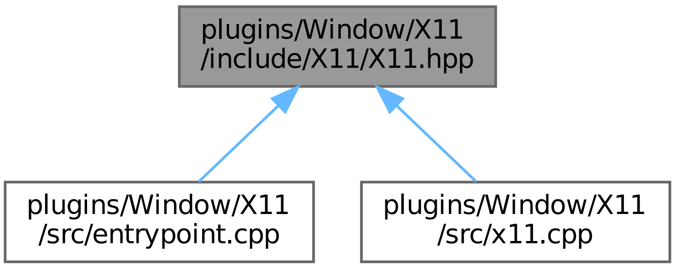
15.89 plugins/Window/X11/include/X11/X11.hpp File Reference

This file contains the X11 class declaration.

#include "Interfaces/IWindow.hpp"
Include dependency graph for X11.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class cae::X11
  Class for the X11 plugin.

Namespaces

- namespace cae

15.89.1 Detailed Description

This file contains the X11 class declaration.

Definition in file X11.hpp.

15.90 X11.hpp

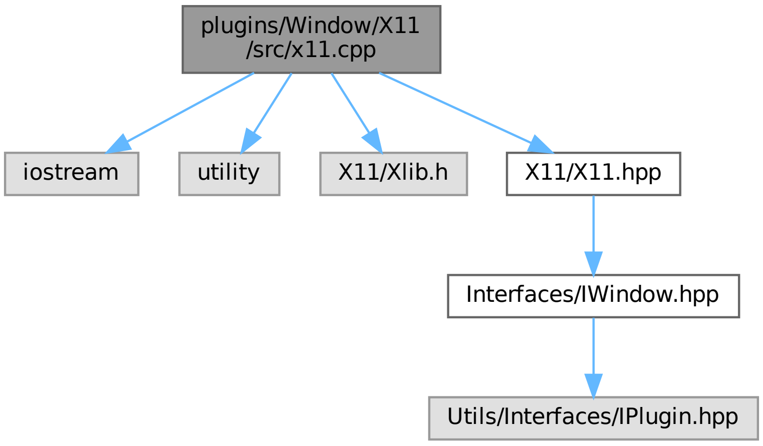
Go to the documentation of this file.

```
///
/// @file X11.hpp
/// @brief This file contains the X11 class declaration
/// @namespace cae
///

#pragma once

#include "Interfaces/IWindow.hpp"

namespace cae
{

    ///
    /// @class X11
    /// @brief Class for the X11 plugin
    /// @namespace cae
    ///
    class X11 final : public IWindow
    {

        public:
            X11() = default;
            ~X11() override = default;

            X11(const X11 &) = delete;
            X11 &operator=(const X11 &) = delete;
            X11(X11 &&) = delete;
            X11 &operator=(X11 &&) = delete;

            [[nodiscard]] std::string getName() const override { return "X11"; }
            [[nodiscard]] utl::PluginType getType() const override { return utl::PluginType::WINDOW; }
            [[nodiscard]] utl::PluginPlatform getPlatform() const override { return utl::PluginPlatform::LINUX; }

            bool create(const std::string &name, WindowSize size) override;
            void close() override;

            [[nodiscard]] NativeWindowHandle getNativeHandle() const override
            {
                return {.window = reinterpret_cast<void *>(m_window), .display = reinterpret_cast<void *>(m_display)};
            }
            [[nodiscard]] WindowSize getWindowSize() const override;

            [[nodiscard]] bool setIcon(const std::string &path) const override;

            [[nodiscard]] bool shouldClose() const override;
            void pollEvents() override;

            bool wasResized() const override { return m_frameBufferResized; }
            void resetResizedFlag() override { m_frameBufferResized = false; }

        private:
            WindowSize m_frameBufferSize;
            mutable bool m_frameBufferResized = false;

            Display *m_display = nullptr;
            Window m_window = 0;
            Atom m_wmDeleteMessage = 0;
            bool m_shouldClose = false;

    }; // class GLFW
} // namespace cae
```

15.91 plugins/Window/X11/src/x11.cpp File Reference

#include <iostream>
#include <utility>
#include <X11/Xlib.h>
#include "X11/X11.hpp"

Include dependency graph for x11.cpp:



15.92 x11.cpp

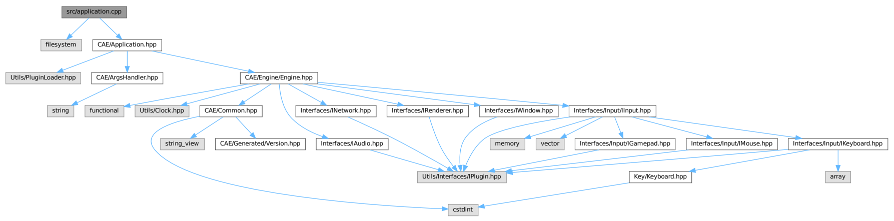
Go to the documentation of this file.

```
#include <iostream>
#include <utility>

#include <X11/Xlib.h>

#include "X11/X11.hpp"

bool cae::X11::create(const std::string &name, const WindowSize size)
{
    m_display = XOpenDisplay(nullptr);
    if (m_display == nullptr)
    {
        std::cerr « "[X11] Failed to open X display\n";
        return false;
    }

    const int screen = DefaultScreen(m_display);
    const Window root = RootWindow(m_display, screen);

    m_window = XCreateSimpleWindow(m_display, root, 0, 0, size.width, size.height, 1, BlackPixel(m_display, screen),
                                   WhitePixel(m_display, screen));

    if (m_window == 0U)
    {
        std::cerr « "[X11] Failed to create X11 window\n";
        return false;
    }

    XStoreName(m_display, m_window, name.c_str());

    XSelectInput(m_display, m_window, ExposureMask | KeyPressMask | StructureNotifyMask);

    m_wmDeleteMessage = XInternAtom(m_display, "WM_DELETE_WINDOW", False);
    XSetWMProtocols(m_display, m_window, &m_wmDeleteMessage, 1);

    XMapWindow(m_display, m_window);
    XFlush(m_display);

    m_frameBufferSize = size;

    return true;
}

void cae::X11::close()
{
    if (m_display != nullptr && m_window != 0U)
    {
        XDestroyWindow(m_display, m_window);
        XCloseDisplay(m_display);
        m_display = nullptr;
        m_window = 0;
    }
}

cae::WindowSize cae::X11::getWindowSize() const
{
    if (m_display == nullptr || m_window == 0U)
    {
        return m_frameBufferSize;
    }

    XWindowAttributes attrs;
    XGetWindowAttributes(m_display, m_window, &attrs);
    return {.width = static_cast<uint16_t>(attrs.width), .height = static_cast<uint16_t>(attrs.height)};
}

bool cae::X11::setIcon(const std::string &path) const
{
    std::cerr « ”[X11] setIcon() not implemented yet (” « path « ”)\n”;
    return false;
}

bool cae::X11::shouldClose() const { return m_shouldClose; }

void cae::X11::pollEvents()
{
    while (XPending(m_display) != 0)
    {
        XEvent event;
        XNextEvent(m_display, &event);

        switch (event.type)
        {
            case Expose:
            {
                XGCValues gcValues;
                GC gc = XCreateGC(m_display, m_window, 0, &gcValues);

                XColor color;
                const Colormap colormap = DefaultColormap(m_display, DefaultScreen(m_display));
                color.red = 0x0000;
                color.green = 0x0000;
                color.blue = 0x0000;
                color.flags = DoRed | DoGreen | DoBlue;
                XAllocColor(m_display, colormap, &color);

                XSetForeground(m_display, gc, color.pixel);

                XFillRectangle(m_display, m_window, gc, 0, 0, m_frameBufferSize.width, m_frameBufferSize.height);

                XFreeGC(m_display, gc);
                break;
            }
            case ConfigureNotify:
                m_frameBufferResized = true;
                m_frameBufferSize.width = event.xconfigure.width;
                m_frameBufferSize.height = event.xconfigure.height;
                break;
            case ClientMessage:
                if (std::cmp_equal(event.xclient.data.l[0], m_wmDeleteMessage))
                {
                    m_shouldClose = true;
                }
                break;
            default:
                break;
        }
    }

    XFlush(m_display);
}
```

15.93 src/application.cpp File Reference

#include <filesystem>
#include ”CAE/Application.hpp”

Include dependency graph for application.cpp:



Functions

- static std::vector< std::shared_ptr< utl::IPlugin > > loadPlugins (const std::unique_ptr< utl::↩ PluginLoader > &loader)

15.93.1 Function Documentation

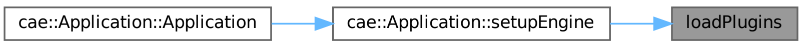
15.93.1.1 loadPlugins()

static std::vector< std::shared_ptr< utl::IPlugin > > loadPlugins (
const std::unique_ptr< utl::PluginLoader > & loader) [static]

Definition at line 5 of file application.cpp.

Referenced by cae::Application::setupEngine().

Here is the caller graph for this function:



15.94 application.cpp

Go to the documentation of this file.

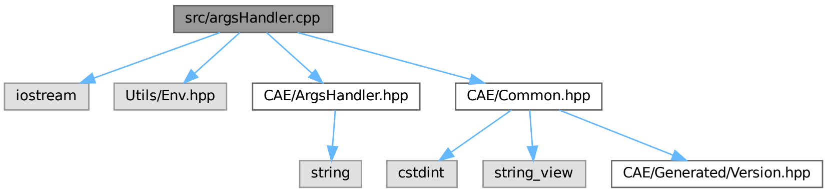
```
00001 #include <filesystem>
00002
00003 #include "CAE/Application.hpp"
00004
00005 static std::vector<std::shared_ptr<utl::IPlugin» loadPlugins(const std::unique_ptr<utl::PluginLoader> &loader)
00006 {
00007     const std::filesystem::path pluginDir{PLUGINS_DIR};
00008     std::vector<std::shared_ptr<utl::IPlugin» loadedPlugins;
00009
00010     for (const auto &entry : std::filesystem::directory_iterator(pluginDir))
00011     {
00012         if (!entry.is_regular_file() || entry.path().extension() != PLUGINS_EXTENSION)
00013         {
00014             continue;
00015         }
00016         const std::string pluginPath = entry.path().string();
00017         if (auto plugin = loader->loadPlugin<utl::IPlugin>(pluginPath); plugin != nullptr)
00018         {
```

```
            loadedPlugins.push_back(plugin);
        }
        else
        {
            utl::Logger::log("Failed to load plugin: " + pluginPath, utl::LogLevel::WARNING);
        }
    }
    if (loadedPlugins.empty())
    {
        utl::Logger::log("No plugins loaded from directory: " + pluginDir.string(), utl::LogLevel::WARNING);
    }

    return loadedPlugins;
}

cae::Application::Application(const ArgsConfig &argsConfig, const EnvConfig &envConfig)
    : m_pluginLoader(std::make_unique<utl::PluginLoader>())
{
    utl::Logger::log("PROJECT INFO:\n" + std::string(Message::VERSION_MSG), utl::LogLevel::INFO);

    try
    {
        m_appConfig.envConfig = envConfig;

        if (!argsConfig.config_path.empty())
        {
            m_appConfig.engineConfig = parseEngineConf(argsConfig.config_path);
        }
        setupEngine("Vulkan", "X11");
    }
    catch (const std::exception &e)
    {
        std::cerr « "Error: " « e.what() « '\n';
    }
}

void cae::Application::setupEngine(const std::string &rendererName, const std::string &windowName)
{
    std::shared_ptr<IWindow> windowPlugin = nullptr;
    std::shared_ptr<IRenderer> rendererPlugin = nullptr;

    for (auto &plugin : loadPlugins(m_pluginLoader))
    {
        if (const auto renderer = std::dynamic_pointer_cast<IRenderer>(plugin))
        {
            if (renderer->getName() == rendererName)
            {
                rendererPlugin = renderer;
            }
        }
        if (const auto window = std::dynamic_pointer_cast<IWindow>(plugin))
        {
            if (window->getName() == windowName)
            {
                windowPlugin = window;
            }
        }
    }
    m_engine = std::make_unique<Engine>(
        m_appConfig.engineConfig, []() { return nullptr; }, []() { return nullptr; }, []() { return nullptr; },
        [rendererPlugin]() { return rendererPlugin; }, [windowPlugin]() { return windowPlugin; });
}

void cae::Application::start() const { m_engine->run(); }

void cae::Application::stop()
{
    m_engine->stop();

    m_pluginLoader = nullptr;
    m_engine = nullptr;
}
```

15.95 src/argsHandler.cpp File Reference

```
#include <iostream>
#include <Utils/Env.hpp>
#include "CAE/ArgsHandler.hpp"
```

#include "CAE/Common.hpp"
Include dependency graph for argsHandler.cpp:



15.96 argsHandler.cpp

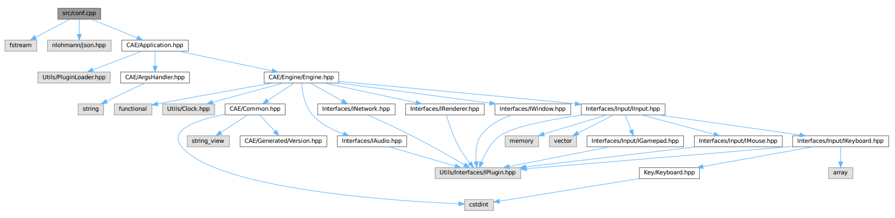
Go to the documentation of this file.

```
#include <iostream>

#include <Utils/Env.hpp>

#include "CAE/ArgsHandler.hpp"
#include "CAE/Common.hpp"

cae::ArgsConfig cae::ArgsHandler::ParseArgs(const int argc, const char *const *argv)
{
    ArgsConfig config;
    config.run = true;

    if (argc <= 1)
    {
        return config;
    }

    for (int i = 1; i < argc; ++i)
    {
        std::string arg = argv[i];

        if (arg == "-h" || arg == "--help")
        {
            std::cout « Message::HELP_MSG;
            config.run = false;
            return config;
        }
        if (arg == "-v" || arg == "--version")
        {
            std::cout « Message::VERSION_MSG;
            config.run = false;
            return config;
        }
        if (arg == "-c" || arg == "--config")
        {
            if (i + 1 >= argc)
            {
                throw std::runtime_error("Missing value for argument " + arg);
            }

            config.config_path = argv[++i];
        }
        else
        {
            throw std::runtime_error("Unknown argument: " + arg + ". Use -h or --help to see available options.");
        }
    }

    return config;
}

cae::EnvConfig cae::ArgsHandler::ParseEnv(const char *const *envp)
{
    EnvConfig config;
    const auto envMap = utl::getEnvMap(envp);

    if (envMap.contains("USER"))
    {
```

```
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 }
```

15.97 src/conf.cpp File Reference

#include <fstream>
#include <nlohmann/json.hpp>
#include "CAE/Application.hpp"
Include dependency graph for conf.cpp:



Typedefs

- using json = nlohmann::json

15.97.1 Typedef Documentation

15.97.1.1 json

using json = nlohmann::json

Definition at line 8 of file conf.cpp.

15.98 conf.cpp

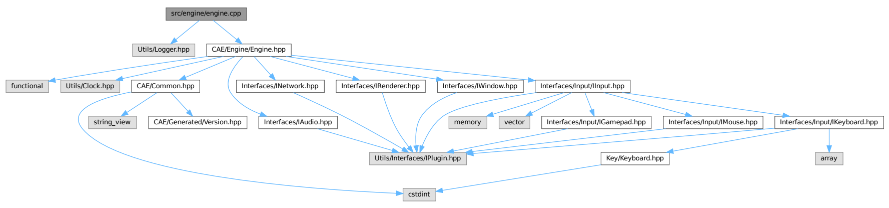
Go to the documentation of this file.

```
00001 #include <fstream>
00002
00003 #include <nlohmann/json.hpp>
00004
00005 #include "CAE/Application.hpp"
00006
00007 namespace fs = std::filesystem;
00008 using json = nlohmann::json;
00009
00010 cae::EngineConfig cae::Application::parseEngineConf(const std::string &path)
00011 {
00012     const fs::path filePath(path);
00013     if (!fs::exists(filePath))
00014     {
00015         std::cerr « "Config file not found: " « filePath « '\n';
00016         return {};
```

```
    }
    if (!fs::is_regular_file(filePath))
    {
        std::cerr « "Config path is not a regular file: " « filePath « '\n';
        return {};
    }

    std::ifstream file(filePath);
    if (!file.is_open())
    {
        std::cerr « "Failed to open config file: " « filePath « '\n';
        return {};
    }

    json j;
    try
    {
        file » j;
    }
    catch (const json::parse_error &e)
    {
        std::cerr « "Failed to parse JSON config (" « filePath « "): " + std::string(e.what()) « '\n';
        return {};
    }
    cae::EngineConfig config;
    utl::Logger::log("Loading config: " + filePath.string(), utl::LogLevel::INFO);
    if (j.contains("audio"))
    {
        const auto &audio = j["audio"];
        if (audio.contains("masterVolume") && audio["masterVolume"].is_number())
        {
            config.audio_master_volume = audio["masterVolume"];
        }
        if (audio.contains("muted") && audio["muted"].is_boolean())
        {
            config.audio_muted = audio["muted"];
        }
    }
    if (j.contains("network"))
    {
        const auto &network = j["network"];
        if (network.contains("host") && network["host"].is_string())
        {
            config.network_host = network["host"];
        }
        if (network.contains("port") && network["port"].is_number_unsigned())
        {
            config.network_port = network["port"];
        }
    }
    if (j.contains("renderer"))
    {
        const auto &renderer = j["renderer"];
        if (renderer.contains("vsync") && renderer["vsync"].is_boolean())
        {
            config.renderer_vsync = renderer["vsync"];
        }
        if (renderer.contains("frameRateLimit") && renderer["frameRateLimit"].is_number_unsigned())
        {
            config.renderer_frame_rate_limit = renderer["frameRateLimit"];
        }
    }
    if (j.contains("window"))
    {
        const auto &window = j["window"];
        if (window.contains("width") && window["width"].is_number_unsigned())
        {
            config.window_width = window["width"];
        }
        if (window.contains("height") && window["height"].is_number_unsigned())
        {
            config.window_height = window["height"];
        }
        if (window.contains("fullscreen") && window["fullscreen"].is_boolean())
        {
            config.window_fullscreen = window["fullscreen"];
        }
        if (window.contains("name") && window["name"].is_string())
        {
            config.window_name = window["name"];
        }
    }

    return config;
}
```

15.99 src/engine/engine.cpp File Reference

#include <Utils/Logger.hpp>
#include "CAE/Engine/Engine.hpp"
Include dependency graph for engine.cpp:



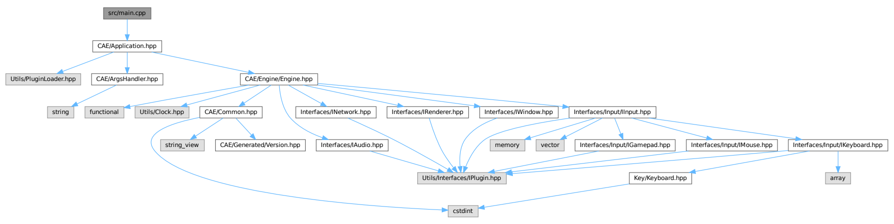
15.100 engine.cpp

Go to the documentation of this file.

```
#include <Utils/Logger.hpp>

#include "CAE/Engine/Engine.hpp"

cae::Engine::Engine(const EngineConfig &config, const std::function<std::shared_ptr<IAudio>()> &audioFactory,
                    const std::function<std::shared_ptr<IInput>()> &inputFactory,
                    const std::function<std::shared_ptr<INetwork>()> &networkFactory,
                    const std::function<std::shared_ptr<IRenderer>()> &rendererFactory,
                    const std::function<std::shared_ptr<IWindow>()> &windowFactory)
    : m_audioPlugin(audioFactory()), m_inputPlugin(inputFactory()), m_networkPlugin(networkFactory()),
      m_rendererPlugin(rendererFactory()), m_windowPlugin(windowFactory()),
m_clock(std::make_unique<utl::Clock>())
{
    utl::Logger::log("Loading engine with configuration:", utl::LogLevel::INFO);
    utl::Logger::log("\tAudio master volume: " + std::to_string(config.audio_master_volume), utl::LogLevel::INFO);
    utl::Logger::log("\tAudio muted: " + std::string(config.audio_muted ? "true" : "false"), utl::LogLevel::INFO);
    utl::Logger::log("\tNetwork host: " + config.network_host, utl::LogLevel::INFO);
    utl::Logger::log("\tNetwork port: " + std::to_string(config.network_port), utl::LogLevel::INFO);
    utl::Logger::log("\tRenderer vsync: " + std::string(config.renderer_vsync ? "true" : "false"), utl::LogLevel::INFO);
    utl::Logger::log("\tRenderer frame rate limit: " + std::to_string(config.renderer_frame_rate_limit),
                     utl::LogLevel::INFO);
    utl::Logger::log("\tWindow width: " + std::to_string(config.window_width), utl::LogLevel::INFO);
    utl::Logger::log("\tWindow height: " + std::to_string(config.window_height), utl::LogLevel::INFO);
    utl::Logger::log("\tWindow fullscreen: " + std::string(config.window_fullscreen ? "true" : "false"),
                     utl::LogLevel::INFO);
    utl::Logger::log("\tWindow name: " + config.window_name, utl::LogLevel::INFO);
    m_windowPlugin->create(config.window_name, {.width = config.window_width, .height = config.window_height});
}

void cae::Engine::run() const
{
    while (!m_windowPlugin->shouldClose())
    {
        m_windowPlugin->pollEvents();
    }
}

void cae::Engine::stop()
{
    utl::Logger::log("Stopping engine...", utl::LogLevel::INFO);
    m_windowPlugin->close();

    m_audioPlugin = nullptr;
    m_inputPlugin = nullptr;
    m_networkPlugin = nullptr;
    m_rendererPlugin = nullptr;
    m_windowPlugin = nullptr;
}
```

15.101 src/main.cpp File Reference

#include "CAE/Application.hpp"
Include dependency graph for main.cpp:



Functions

- int main (const int argc, const char ∗const ∗argv, const char ∗const ∗envp)

15.101.1 Function Documentation

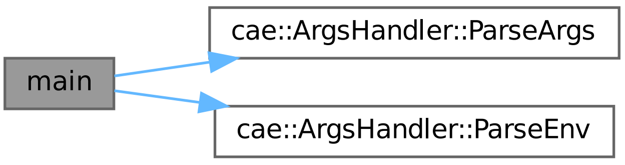
15.101.1.1 main()

```
int main (
        const int argc,
        const char ∗const ∗ argv,
        const char ∗const ∗ envp)
```

Definition at line 3 of file main.cpp.

References cae::ArgsHandler::ParseArgs(), cae::ArgsHandler::ParseEnv(), and cae::ArgsConfig::run.

Here is the call graph for this function:



15.102 main.cpp

Go to the documentation of this file.

```
#include "CAE/Application.hpp"

int main(const int argc, const char *const *argv, const char *const *envp)
{
    std::unique_ptr<cae::Application> app = nullptr;

    utl::Logger::init();
    try
    {
        cae::ArgsConfig argsConfig = cae::ArgsHandler::ParseArgs(argc, argv);
        cae::EnvConfig envConfig = cae::ArgsHandler::ParseEnv(envp);
        if (!argsConfig.run)
        {
            return EXIT_SUCCESS;
        }
        app = std::make_unique<cae::Application>(argsConfig, envConfig);
        app->start();
        app->stop();
    }
    catch (const std::exception &e)
    {
        std::cerr << "Error: " << e.what() << '\n';
        return EXIT_FAILURE;
    }
    catch (...)
    {
        std::cerr << "Unknown error occurred\n";
        return EXIT_FAILURE;
    }
    return EXIT_SUCCESS;
}
```

Index