Vesper

0.5.1

Generated by Doxygen 1.16.1

Chapter 1

Vesper

Vesper is a lightweight 2D/3D engine and editor inspired by TheCherno's Game Engine Architecture series for the Hazel engine.
It provides a renderer, scene & entity system, editor UI, input handling, and utilities to build simple games and interactive applications in C++ (C++17).

It is generally usable as an easily modifiable base line for creating graphics and software applications.

> The specific purpose is for the creation of Particle System demos, experiments, and visualizations.
>
> ***It is not intended to be a full-featured game engine,***
>
> But rather an application to create a range of visual effects and particle system simulations.

1.1 Quick overview

- Language: C++17
- Primary IDE: Visual Studio 2022
- Platforms: Windows (Visual Studio solution & VCXPROJ files provided)
  - Planned: Linux, macOS
- Build system: Premake5
- Third-party libraries:
  - entt (entity-component system)
  - Glad (OpenGL function loading)
  - GLFW (windowing and input)
  - glm (math library)
  - ImGui (editor UI)
  - spdlog (logging)
  - stb (image loading)
- Core features:
  - Scene / Entity / Component system
  - 2D renderer with orthographic and perspective cameras
  - ImGui-based editor layer
  - Input and event handling
  - Starter particle system and instrumentation

1.2 Repository layout (high level)

- `Vesper/` Engine source (core, renderer, scene, editor integration)
  - `src/Vesper` engine code
  - `vendor/` third-party libs (GLFW, Glad, ImGui)
- `Vesper-Editor/` Editor build that hosts engine in an editor UI
- `Sandbox/` Example application(s) that use the engine
- `README.md`, `LICENSE-Hazel_Apache2.txt` repo metadata and licensing

Notable engine files:

- `Vesper/src/Vesper.h` single include for engine public API
- `Vesper/src/Vesper/Scene/*` Scene, Entity, Components, Camera
- `Vesper-Editor/src/EditorLayer.*` Editor integration layer and UI panels

1.3 Getting started (clone + third-party libs)

```
git clone --recursive https://github.com/nomadiidamon/Vesper <desiredLocation>
```

1.4 Build (Visual Studio 2022)

1. Run the 'Win-GenProjects.bat' script from the repository root 'Vesper/scripts/Win-GenProjects.bat' to generate the Visual Studio solution file (`.sln`).
   - This runs Premake5 which is located in 'Vesper/Vendor/bin/premake'.
2. Open the solution in Visual Studio 2022.
3. Select the desired configuration (e.g. `Debug` or `Release`) and platform (`x64`).
4. Right-click the project you want to run (`Vesper-Editor` or `Sandbox`) and choose **Set as Startup Project**.
5. Build and run (F5).

Troubleshooting:

- Ensure the Windows SDK and Visual C++ toolset for VS2022 are installed.
- If include or linker errors appear, confirm vendor project builds (GLFW, Glad, ImGui, etc.) and that project dependencies are set correctly.
- For unresolved externals, confirm platform (x86/x64) consistency across projects.

1.5 Run the editor or sandbox

- Start the editor by launching the `Vesper-Editor` startup project.
- Use the `Sandbox` project to iterate small demos that use the engine API.
  - Modify or add new demo source files in the `Sandbox/src` folder.
- Create new projects that link against the Vesper engine.

1.6 New Project example code

```
#include <Vesper.h>
#include <Vesper/Core/EntryPoint.h> // can only be included in one source file

#include "imgui/imgui.h"

class ExampleLayer : public Vesper::Layer
{
public:
    ExampleLayer()
        : Layer("ExampleLayer")
    {
    }

    ~ExampleLayer() override = default;

    // Called once when the layer is attached to the layer stack
    void OnAttach() override {}

    // Called once when the layer is detached from the layer stack
    void OnDetach() override {}

    // Called every frame with the frame time
    void OnUpdate(Vesper::Timestep ts) override {}

    // Render ImGui UI for this layer (optional)
    void OnImGuiRender() override {
        ImGui::Begin("Example Layer");
        ImGui::Text("Hello from ExampleLayer!");
        ImGui::End();
    }

    // Receive events (keyboard/mouse/window/etc.)
    void OnEvent(Vesper::Event& e) override {}
};

class YourAppNameHere : public Vesper::Application
{
public:
    YourAppNameHere() {

        PushLayer(new ExampleLayer());
    }

    ~YourAppNameHere() {    }
};

Vesper::Application* Vesper::CreateApplication() {
    return new YourAppNameHere();
}
```

1.7 Contributing

Please open issues or pull requests.

1.8 Notes / TODO

- Pin third-party dependency versions and document them.
- Add a `CONTRIBUTING.md` with coding standards, branch workflow, and required checks (formatting, tests).

1.9 Licensing

License status: Proprietary (subject to change)

Vesper itself is not **currently** released under any specific license and is provided as-is. Copyright 2025 by Damon S. Green II (nomad_ii_damon). All rights reserved unless otherwise stated.

This project contains code derived from the Hazel Engine by TheCherno, licensed under the Apache License 2.0, obtained through public tutorialization and source code access.

Original Hazel code and derivative works thereof are licensed under the Apache License 2.0. Other original code in this repository is Copyright 2025 Damon S. Green II (nomad_ii_damon).

See the LICENSE file for details.

1.9.0.1 Fully-Modified File Header for Licensing

Copyright TheCherno Modifications Copyright 2025 Damon S. Green II (nomad_ii_damon)

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

```
[Apache](http://www.apache.org/licenses/LICENSE-2.0)
```

1.9.0.2 Modified File Header for Licensing

Based on Hazel Engine (Apache 2.0) Modified by Damon S. Green II, 2025

Chapter 2

Directory Hierarchy

2.1 Directories

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Directory Documentation

7.1 Vesper/src/Vesper/App Directory Reference

Files

- file Application.cpp
- file Application.h
- file EntryPoint.h
- file Layer.cpp
- file Layer.h
- file LayerStack.cpp
- file LayerStack.h
- file Window.h

7.2 Vesper/src/Vesper/Core Directory Reference

Files

- file Asserts.h
- file Base.h
- file Color.h
- file Config.h
- file Defines_Macros.h
- file Log.cpp
- file Log.h
- file Math.cpp
- file Math.h
- file PlatformDetection.h
- file Random.h
- file Timer.h
- file Timestep.h

7.3 Vesper/src/Vesper/Debug Directory Reference

Files

- file Instrumentor.h

7.4 Vesper/src/Vesper/Events Directory Reference

Files

- file ApplicationEvent.h
- file Event.h
- file KeyEvent.h
- file MouseEvent.h

7.5 Vesper/src/Vesper/ImGui Directory Reference

Files

- file ImGuiBuild.cpp
- file ImGuiLayer.cpp
- file ImGuiLayer.h
- file VesperImGui.h

7.6 Vesper/src/Vesper/ImGuizmo Directory Reference

Files

- file ImGuizmoBuild.cpp

7.7 Vesper/src/Vesper/Input Directory Reference

Files

- file Input.h
- file KeyCodes.h
- file MouseButtonCodes.h

7.8 Vesper/src/RenderAPI/OpenGL Directory Reference

Files

- file OpenGLBuffer.cpp
- file OpenGLBuffer.h
- file OpenGLContext.cpp
- file OpenGLContext.h
- file OpenGLFramebuffer.cpp
- file OpenGLFramebuffer.h
- file OpenGLImGuiLayer.cpp
- file OpenGLImGuiLayer.h
- file OpenGLRendererAPI.cpp
- file OpenGLRendererAPI.h
- file OpenGLShader.cpp
- file OpenGLShader.h
- file OpenGLTexture.cpp
- file OpenGLTexture.h
- file OpenGLUniformBuffer.cpp
- file OpenGLUniformBuffer.h
- file OpenGLVertexArray.cpp
- file OpenGLVertexArray.h

7.9 Vesper-Editor/src/Panels Directory Reference

Files

- file SceneHierarchyPanel.cpp
- file SceneHierarchyPanel.h

7.10 Vesper/src/Vesper/ParticleSystem Directory Reference

Files

- file ParticleSystem.cpp
- file ParticleSystem.h

7.11 Vesper/src/Platform Directory Reference

Directories

- directory Windows

7.12 Vesper/src/RenderAPI Directory Reference

Directories

- directory OpenGL

7.13 Vesper/src/Vesper/Renderer Directory Reference

Files

- file Buffer.cpp
- file Buffer.h
- file Camera.h
- file EditorCamera.cpp
- file EditorCamera.h
- file Framebuffer.cpp
- file Framebuffer.h
- file GraphicsContext.h
- file OrthographicCamera.cpp
- file OrthographicCamera.h
- file OrthographicCameraController.cpp
- file OrthographicCameraController.h
- file RenderCommand.cpp
- file RenderCommand.h
- file Renderer.cpp
- file Renderer.h

- file Renderer2D.cpp
- file Renderer2D.h
- file RendererAPI.cpp
- file RendererAPI.h
- file Shader.cpp
- file Shader.h
- file SubTexture2D.cpp
- file SubTexture2D.h
- file Texture.cpp
- file Texture.h
- file UniformBuffer.cpp
- file UniformBuffer.h
- file VertexArray.cpp
- file VertexArray.h

7.14 Vesper/src/Vesper/Scene Directory Reference

Files

- file Components.h
- file Entity.cpp
- file Entity.h
- file Scene.cpp
- file Scene.h
- file SceneCamera.cpp
- file SceneCamera.h
- file SceneSerializer.cpp
- file SceneSerializer.h
- file ScriptableEntity.h

7.15 Vesper-Editor/src Directory Reference

Directories

- directory Panels

Files

- file EditorLayer.cpp
- file EditorLayer.h
- file VesperEditorApp.cpp

7.16 Vesper/src Directory Reference

Directories

- directory Platform
- directory RenderAPI
- directory Vesper

Files

- file Vesper.h
- file vzpch.cpp
- file vzpch.h

7.17 Vesper/src/Vesper/Utils Directory Reference

Files

- file PlatformUtils.h

7.18 Vesper Directory Reference

Directories

- directory src

7.19 Vesper/src/Vesper Directory Reference

Directories

- directory App
- directory Core
- directory Debug
- directory Events
- directory ImGui
- directory ImGuizmo
- directory Input
- directory ParticleSystem
- directory Renderer
- directory Scene
- directory Utils

7.20 Vesper-Editor Directory Reference

Directories

- directory src

7.21 Vesper/src/Platform/Windows Directory Reference

Files

- file WindowsInput.cpp
- file WindowsPlatformUtils.cpp
- file WindowsWindow.cpp
- file WindowsWindow.h

Chapter 8

Namespace Documentation

8.1 InstrumentorUtils Namespace Reference

Classes

- struct ChangeResult

Functions

- template<size_t N, size_t K>
  constexpr auto CleanupOutputString (const char(&expr)[N], const char(&remove)[K])

8.1.1 Class Documentation

8.1.1.1 struct InstrumentorUtils::ChangeResult

template<size_t N>
struct InstrumentorUtils::ChangeResult< N >

Class Members

char	Data↩ [N]	

8.1.2 Function Documentation

8.1.2.1 CleanupOutputString()

```
template<size_t N, size_t K>
auto InstrumentorUtils::CleanupOutputString (
            const char(&) expr[N],
            const char(&) remove[K])  [constexpr]
```

```
00185     {
00186         ChangeResult<N> result = {};
00187 
00188         size_t srcIndex = 0;
00189         size_t dstIndex = 0;
00190         while (srcIndex < N)
00191         {
00192             size_t matchIndex = 0;
00193             while (matchIndex < K - 1 && srcIndex + matchIndex < N - 1 && expr[srcIndex + matchIndex]
      == remove[matchIndex])
00194                 matchIndex++;
00195             if (matchIndex == K - 1)
00196                 srcIndex += matchIndex;
00197             result.Data[dstIndex++] = expr[srcIndex] == '"' ? '\"' : expr[srcIndex];
00198             srcIndex++;
00199         }
00200         return result;
00201     }
```

8.2 Vesper Namespace Reference

TEMPORARY.

Namespaces

- namespace Math
- namespace Random
- namespace Color

Classes

- class Input
- class FileDialogs
- class FileSystem
- struct ApplicationSettings

  WIP. More...
- class Application
- class WindowsWindow
- class WindowResizeEvent
- class WindowCloseEvent
- class AppTickEvent
- class AppUpdateEvent
- class AppRenderEvent
- class MouseMovedEvent
- class MouseScrolledEvent
- class MouseButtonEvent
- class MouseButtonPressedEvent
- class MouseButtonReleasedEvent

- class KeyEvent
- class KeyPressedEvent
- class KeyReleasedEvent
- class KeyTypedEvent
- class OpenGLContext
- class GraphicsContext
- class OpenGLFramebuffer
- class OpenGLImGuiLayer
- class Layer
- class RendererAPI
- class OpenGLRendererAPI
- class OpenGLShader
- class OpenGLTexture2D
- class OpenGLUniformBuffer
- class OpenGLVertexArray
- class LayerStack
- class Log
- class ImGuiLayer
- struct ParticleProps
- class ParticleSystem
- class Renderer2D
- class OrthographicCamera
- struct BufferElement
- class BufferLayout
- class VertexBuffer
- class IndexBuffer
- class Renderer
- class OpenGLVertexBuffer
- class OpenGLIndexBuffer
- class EditorCamera
- struct FramebufferSpecification
- class Framebuffer
- struct ProfileResult
- struct InstrumentationSession
- class Instrumentor
- class InstrumentationTimer
- struct OrthographicCameraBounds
- class OrthographicCameraController
- class RenderCommand
- class VertexArray
- class SubTexture2D
- class Camera
- struct QuadVertex
- struct Renderer2DData
- class UniformBuffer
- class Texture
- class Texture2D
- class TextureLibrary
- class Shader
- class ShaderLibrary
- class Entity
- class SceneCamera
- class EditorLayer
- class SceneSerializer
- class SceneHierarchyPanel

- struct UUID
- struct UUIDComponent
- struct NameComponent
- struct TransformComponent
- struct SpriteRendererComponent
- struct SubTextureComponent
- struct TextureAnimationComponent
- struct CameraComponent
- struct NativeScriptComponent
- class VesperEditor
- struct WindowProps
- class Window
- class Timestep
- class Event
- class EventDispatcher
- class ScriptableEntity
- class Scene

Typedefs

- using FloatingPointMicroseconds = std::chrono::duration<double, std::micro>
- template<typename T>
  using Scope = std::unique_ptr<T>
- template<typename T>
  using Ref = std::shared_ptr<T>

Enumerations

- enum class WindowMode { Windowed = 0 , Fullscreen = 1 , Borderless = 2 }
  WIP.
- enum class ShaderDataType {
  None = 0 , Float , Float2 , Float3 ,
  Float4 , Mat3 , Mat4 , Int ,
  Int2 , Int3 , Int4 , Bool }
- enum class EventType {
  None = 0 , WindowClose , WindowResize , WindowFocus ,
  WindowLostFocus , WindowMoved , AppTick , AppUpdate ,
  AppRender , KeyPressed , KeyReleased , KeyTyped ,
  MouseButtonPressed , MouseButtonReleased , MouseMoved , MouseScrolled }
- enum EventCategory {
  None = 0 , EventCategoryApplication = BIT(0) , EventCategoryInput = BIT(1) , EventCategoryKeyboard = BIT(2) ,
  EventCategoryMouse = BIT(3) , EventCategoryMouseButton = BIT(4) }

Functions

- Application * CreateApplication ()
- static void GLFWErrorCallback (int error, const char *description)
- static GLenum ShaderTypeFromString (const std::string &type)
- static GLenum ShaderDataTypeToOpenGLBaseType (ShaderDataType type)
- static uint32_t ShaderDataTypeSize (ShaderDataType type)
- static void SerializeEntity (YAML::Emitter &out, Entity entity)
- static void DisplayVesperInfo_ImGui ()

- static void DrawVec3Control (const std::string &label, glm::vec3 &values, float resetValue=0.0f, float columnWidth=100.0f)
- static void DrawVec2Control (const std::string &label, glm::vec2 &values, float resetValue=0.0f, float columnWidth=100.0f)
- static void SubTextureEdit (const std::string &label, SubTextureComponent &subTexture)
- template<typename T, typename UIFunction>
  static void DrawComponent (const std::string &name, Entity entity, UIFunction uiFunction)
- std::string format_as (const Event &e)
- template<typename T, typename... Args>
  constexpr Scope< T > CreateScope (Args &&... args)
- template<typename T, typename... Args>
  constexpr Ref< T > CreateRef (Args &&... args)

Variables

- static bool s_GLFWInitialized = false
- static const uint32_t s_MaxFramebufferSize = 8192

  TODO: Get the actual maximum size from the GPU!
- static Renderer2DData s_Data

8.2.1 Detailed Description

TEMPORARY.

TYPE ALIASES.

Temporary.

TODO: Abstract this to OpenGL/DirectX/Vulkan etc ImGui layers.

8.2.2 Class Documentation

8.2.2.1 struct Vesper::ApplicationSettings

WIP.

Class Members

string	ApplicationName = "Vesper Application"	
bool	EnableImGui = true	
bool	EnableVSync = false	
uint32_t	Height = 720	
WindowMode	Mode = WindowMode::Windowed	
API	RendererAPI = RendererAPI::API::OpenGL	
uint32_t	Width = 1280	
string	WorkingDirectory	

8.2.2.2 struct Vesper::ParticleProps

Class Members

vec4	ColorBegin = { 1.0f, 1.0f, 1.0f, 1.0f }	
vec4	ColorEnd = { 1.0f, 1.0f, 1.0f, 1.0f }	
float	LifeTime = 1.0f	
float	LifetimeVariation = 0.0f	
vec3	Position = { 0.0f, 0.0f, 0.0f }	
float	Rotation = 0.0f	
float	RotationVariation = 0.0f	
float	SizeBegin = 1.0f	
float	SizeEnd = 0.0f	
float	SizeVariation = 0.0f	
vec3	Velocity = { 0.0f, 0.0f, 0.0f }	
vec3	VelocityVariation = { 0.0f, 0.0f, 0.0f }	

8.2.2.3 struct Vesper::FramebufferSpecification

Class Members

uint32_t	Height	
uint32_t	Samples = 1	
bool	SwapChainTarget = false	
uint32_t	Width	

8.2.2.4 struct Vesper::ProfileResult

Class Members

long long	End	
string	Name	
long long	Start	
uint32_t	ThreadID	

8.2.2.5 struct Vesper::InstrumentationSession

Class Members

string	Name	

8.2.2.6 struct Vesper::QuadVertex

Class Members

vec4	Color	
vec3	Position	
vec2	TexCoord	
float	TexIndex	
float	TilingFactor	

8.2.3 Typedef Documentation

8.2.3.1 FloatingPointMicroseconds

```
using Vesper::FloatingPointMicroseconds = std::chrono::duration<double, std::micro>
```

8.2.3.2 Ref

```
template<typename T>
using Vesper::Ref = std::shared_ptr<T>
```

8.2.3.3 Scope

```
template<typename T>
using Vesper::Scope = std::unique_ptr<T>
```

8.2.4 Enumeration Type Documentation

8.2.4.1 EventCategory

```
enum Vesper::EventCategory
```

Enumerator

None	
EventCategoryApplication	
EventCategoryInput	
EventCategoryKeyboard	
EventCategoryMouse	
EventCategoryMouseButton	

```
00018     {
00019         None = 0,
00020         EventCategoryApplication    = BIT(0),
00021         EventCategoryInput          = BIT(1),
00022         EventCategoryKeyboard       = BIT(2),
00023         EventCategoryMouse          = BIT(3),
00024         EventCategoryMouseButton    = BIT(4)
00025     };
```

8.2.4.2 EventType

enum class Vesper::EventType [strong]

Enumerator

None	
WindowClose	
WindowResize	
WindowFocus	
WindowLostFocus	
WindowMoved	
AppTick	
AppUpdate	
AppRender	
KeyPressed	
KeyReleased	
KeyTyped	
MouseButtonPressed	
MouseButtonReleased	
MouseMoved	
MouseScrolled	

```
00010     {
00011         None = 0,
00012         WindowClose, WindowResize, WindowFocus, WindowLostFocus, WindowMoved,
00013         AppTick, AppUpdate, AppRender,
00014         KeyPressed, KeyReleased, KeyTyped,
00015         MouseButtonPressed, MouseButtonReleased, MouseMoved, MouseScrolled
00016     };
```

8.2.4.3 ShaderDataType

enum class Vesper::ShaderDataType [strong]

Enumerator

None	
Float	
Float2	
Float3	
Float4	
Mat3	
Mat4	
Int	
Int2	
Int3	
Int4	

Bool	

```
00005                                {
00006         None = 0,
00007         Float, Float2, Float3, Float4,
00008         Mat3, Mat4,
00009         Int, Int2, Int3, Int4,
00010         Bool
00011     };
```

8.2.4.4 WindowMode

```
enum class Vesper::WindowMode  [strong]
```

WIP.

Enumerator

Windowed	
Fullscreen	
Borderless	

```
00017     {
00018         Windowed = 0,
00019         Fullscreen = 1,
00020         Borderless = 2
00021     };
```

8.2.5 Function Documentation

8.2.5.1 CreateApplication()

```
Application * Vesper::CreateApplication ()
00024     {
00025         return new VesperEditor();
00026     }
```

References Vesper::VesperEditor::VesperEditor().

8.2.5.2 CreateRef()

```
template<typename T, typename...  Args>
Ref< T > Vesper::CreateRef (
             Args &&...  args)  [constexpr]
00024     {
00025         return std::make_shared<T>(std::forward<Args>(args)...);
00026     }
```

8.2.5.3 CreateScope()

```
template<typename T, typename...  Args>
Scope< T > Vesper::CreateScope (
             Args &&...  args)  [constexpr]
00016     {
00017         return std::make_unique<T>(std::forward<Args>(args)...);
00018     }
```

8.2.5.4 DisplayVesperInfo_ImGui()

```
void Vesper::DisplayVesperInfo_ImGui ()  [static]
```

```
00008     {
00009         ImGui::Begin("Vesper Info");
00010 
00011         if (ImGui::TreeNode("About Vesper"))
00012         {
00013             ImGui::Text("Vesper Engine");
00014             ImGui::Text("Version: 0.1.0");
00015             ImGui::Text("Author: Damon Green II");
00016             ImGui::Text("GitHub: https://github.com/nomadiidamon/Vesper");
00017             ImGui::Separator();
00018 
00019             ImGui::Text("Status: ");
00020             ImGui::Text("\tEarly Development of API and 2D Renderer");
00021             ImGui::Separator();
00022 
00023             ImGui::TextWrapped("Vesper is a cross-platform game engine currently in early development.
      The engine is being built from the ground up with a focus on modularity, performance, and ease of use.
      The goal of Vesper is to provide developers with a powerful and flexible toolset for creating games
      and interactive applications.");
00024             ImGui::Separator();
00025 
00026             if (ImGui::TreeNode("Controls:"))
00027             {
00028                 ImGui::Text("\tWASD: Move Camera");
00029                 ImGui::Text("\tQ/E: Rotate Camera (if enabled {see settings})");
00030                 ImGui::Text("\tScroll Wheel: Zoom Camera");
00031                 ImGui::TreePop();
00032             }
00033             ImGui::Separator();
00034 
00035             if (ImGui::TreeNode("RoadMap")) {
00036 
00037                 if (ImGui::TreeNode("Current Features:"))
00038                 {
00039                     ImGui::Text("\t- Cross-Platform Design");
00040                     ImGui::Text("\t\t- Currently Windows only");
00041                     ImGui::Text("\t- OpenGL Renderer");
00042                     ImGui::Text("\t- Orthographic Camera");
00043                     ImGui::Text("\t- Shader System");
00044                     ImGui::Text("\t- Texture Loading");
00045                     ImGui::Text("\t- ImGui Integration");
00046                     ImGui::Text("\t\t- Current settings panel adjusts camera parameters!");
00047 
00048                     ImGui::TreePop();
00049                 }
00050                 ImGui::Separator();
00051 
00052                 if (ImGui::TreeNode("In Progress:"))
00053                 {
00054                     ImGui::Text("\t- 2D Rendering Features");
00055                     ImGui::Text("\t\t- Sprites");
00056                     ImGui::Text("\t\t- Sprite Sheets");
00057                     ImGui::Text("\t\t- Animation");
00058                     ImGui::TreePop();
00059                 }
00060                 ImGui::Separator();
00061 
00062                 if (ImGui::TreeNode("Planned Features:"))
00063                 {
00064                     ImGui::Text("\t- Vulkan Renderer");
00065                     ImGui::Text("\t- 2D Editor");
00066                     ImGui::Text("\t- 2D Particles");
00067                     ImGui::Text("\t- Audio");
00068                     ImGui::Text("\t- Timelining");
00069                     ImGui::Text("\t- Video Playback");
00070                     ImGui::Text("\t- 3D Renderer");
00071                     ImGui::Text("\t- 3D Particles");
00072                     ImGui::TreePop();
00073                 }
00074                 ImGui::TreePop();
00075             }
00076 
00077             ImGui::TreePop();
00078         }
00079         ImGui::End();
00080     }
```

Referenced by Vesper::EditorLayer::OnImGuiRender().

8.2.5.5 DrawComponent()

```
template<typename T, typename UIFunction>
void Vesper::DrawComponent (
            const std::string & name,
            Entity entity,
            UIFunction uiFunction)  [static]
```

```
00333     {
00334         const ImGuiTreeNodeFlags treeNodeFlags = ImGuiTreeNodeFlags_DefaultOpen |
      ImGuiTreeNodeFlags_Framed | ImGuiTreeNodeFlags_SpanAvailWidth | ImGuiTreeNodeFlags_AllowItemOverlap |
      ImGuiTreeNodeFlags_FramePadding;
00335         if (entity.HasComponent<T>())
00336         {
00337             auto& component = entity.GetComponent<T>();
00338             ImVec2 contentRegionAvailable = ImGui::GetContentRegionAvail();
00339 
00340             ImGui::PushStyleVar(ImGuiStyleVar_FramePadding, ImVec2{ 4, 4 });
00341             float lineHeight = ImGui::GetFontSize() + ImGui::GetStyle().FramePadding.y * 2.0f;
00342             ImGui::Separator();
00343             bool open = ImGui::TreeNodeEx((void*)typeid(T).hash_code(), treeNodeFlags, name.c_str());
00344             ImGui::PopStyleVar(
00345             );
00346             ImGui::SameLine(contentRegionAvailable.x - lineHeight * 0.5f);
00347             if (ImGui::Button("+", ImVec2{ lineHeight, lineHeight }))
00348             {
00349                 ImGui::OpenPopup("ComponentSettings");
00350             }
00351 
00352             bool removeComponent = false;
00353             if (ImGui::BeginPopup("ComponentSettings"))
00354             {
00355                 if (ImGui::MenuItem("Remove component"))
00356                     removeComponent = true;
00357 
00358                 ImGui::EndPopup();
00359             }
00360 
00361             if (open)
00362             {
00363                 uiFunction(component);
00364                 ImGui::TreePop();
00365             }
00366 
00367             if (removeComponent)
00368                 entity.RemoveComponent<T>();
00369         }
00370     }
```

8.2.5.6 DrawVec2Control()

```
void Vesper::DrawVec2Control (
            const std::string & label,
            glm::vec2 & values,
            float resetValue = 0.0f,
            float columnWidth = 100.0f)  [static]
```

```
00234     {
00235         ImGuiIO& io = ImGui::GetIO();
00236         auto boldFont = io.Fonts->Fonts[0];
00237 
00238         ImGui::PushID(label.c_str());
00239 
00240         ImGui::Columns(2);
00241         ImGui::SetColumnWidth(0, columnWidth);
00242         ImGui::Text(label.c_str());
00243         ImGui::NextColumn();
00244 
00245         ImGui::PushStyleVar(ImGuiStyleVar_ItemSpacing, ImVec2{ 0, 0 });
00246 
00247         float lineHeight = ImGui::GetFontSize() + ImGui::GetStyle().FramePadding.y * 2.0f;
00248         ImVec2 buttonSize = { lineHeight + 3.0f, lineHeight };
00249 
00250         // Compute available width for the three float controls in the right column
00251         float availableWidth = ImGui::GetContentRegionAvail().x;
00252         float itemSpacing = ImGui::GetStyle().ItemSpacing.x;
00253         float totalButtonWidth = buttonSize.x * 2.0f;
00254         // Account for SameLine() spacings between button+control pairs (conservative estimate)
```

```
00255         float totalSpacing = itemSpacing * 4.0f;
00256         float itemWidth = (availableWidth - totalButtonWidth - totalSpacing) / 2.0f;
00257         if (itemWidth <= 0.0f)
00258             itemWidth = ImGui::CalcItemWidth();
00259 
00260         ImGui::PushStyleColor(ImGuiCol_Button, ImVec4{ 0.8f, 0.1f, 0.15f, 1.0f });
00261         ImGui::PushStyleColor(ImGuiCol_ButtonHovered, ImVec4{ 0.9f, 0.2f, 0.2f, 1.0f });
00262         ImGui::PushStyleColor(ImGuiCol_ButtonActive, ImVec4{ 0.8f, 0.1f, 0.15f, 1.0f });
00263         ImGui::PushFont(boldFont);
00264         if (ImGui::Button("X", buttonSize))
00265             values.x = resetValue;
00266         ImGui::PopFont();
00267         ImGui::PopStyleColor(3);
00268 
00269         ImGui::SameLine();
00270         ImGui::PushItemWidth(itemWidth);
00271         ImGui::DragFloat("##X", &values.x, 0.1f, 0.0f, 0.0f, "%.2f");
00272         ImGui::PopItemWidth();
00273         ImGui::SameLine();
00274 
00275         ImGui::PushStyleColor(ImGuiCol_Button, ImVec4{ 0.2f, 0.7f, 0.2f, 1.0f });
00276         ImGui::PushStyleColor(ImGuiCol_ButtonHovered, ImVec4{ 0.3f, 0.8f, 0.3f, 1.0f });
00277         ImGui::PushStyleColor(ImGuiCol_ButtonActive, ImVec4{ 0.2f, 0.7f, 0.2f, 1.0f });
00278         ImGui::PushFont(boldFont);
00279         if (ImGui::Button("Y", buttonSize))
00280             values.y = resetValue;
00281         ImGui::PopFont();
00282         ImGui::PopStyleColor(3);
00283 
00284         ImGui::SameLine();
00285         ImGui::PushItemWidth(itemWidth);
00286         ImGui::DragFloat("##Y", &values.y, 0.1f, 0.0f, 0.0f, "%.2f");
00287         ImGui::PopItemWidth();
00288         ImGui::SameLine();
00289 
00290         ImGui::PopStyleVar();
00291 
00292         ImGui::Columns(1);
00293 
00294         ImGui::PopID();
00295     }
```

8.2.5.7 DrawVec3Control()

```
void Vesper::DrawVec3Control (
            const std::string & label,
            glm::vec3 & values,
            float resetValue = 0.0f,
            float columnWidth = 100.0f)  [static]
```

```
00156     {
00157         ImGuiIO& io = ImGui::GetIO();
00158         auto boldFont = io.Fonts->Fonts[0];
00159 
00160         ImGui::PushID(label.c_str());
00161 
00162         ImGui::Columns(2);
00163         ImGui::SetColumnWidth(0, columnWidth);
00164         ImGui::Text(label.c_str());
00165         ImGui::NextColumn();
00166 
00167         ImGui::PushStyleVar(ImGuiStyleVar_ItemSpacing, ImVec2{ 0, 0 });
00168 
00169         float lineHeight = ImGui::GetFontSize() + ImGui::GetStyle().FramePadding.y * 2.0f;
00170         ImVec2 buttonSize = { lineHeight + 3.0f, lineHeight };
00171 
00172         // Compute available width for the three float controls in the right column
00173         float availableWidth = ImGui::GetContentRegionAvail().x;
00174         float itemSpacing = ImGui::GetStyle().ItemSpacing.x;
00175         float totalButtonWidth = buttonSize.x * 3.0f;
00176         // Account for SameLine() spacings between button+control pairs (conservative estimate)
00177         float totalSpacing = itemSpacing * 6.0f;
00178         float itemWidth = (availableWidth - totalButtonWidth - totalSpacing) / 3.0f;
00179         if (itemWidth <= 0.0f)
00180             itemWidth = ImGui::CalcItemWidth();
00181 
00182         ImGui::PushStyleColor(ImGuiCol_Button, ImVec4{ 0.8f, 0.1f, 0.15f, 1.0f });
00183         ImGui::PushStyleColor(ImGuiCol_ButtonHovered, ImVec4{ 0.9f, 0.2f, 0.2f, 1.0f });
00184         ImGui::PushStyleColor(ImGuiCol_ButtonActive, ImVec4{ 0.8f, 0.1f, 0.15f, 1.0f });
00185         ImGui::PushFont(boldFont);
00186         if (ImGui::Button("X", buttonSize))
```

```
00187             values.x = resetValue;
00188         ImGui::PopFont();
00189         ImGui::PopStyleColor(3);
00190 
00191         ImGui::SameLine();
00192         ImGui::PushItemWidth(itemWidth);
00193         ImGui::DragFloat("##X", &values.x, 0.1f, 0.0f, 0.0f, "%.2f");
00194         ImGui::PopItemWidth();
00195         ImGui::SameLine();
00196 
00197         ImGui::PushStyleColor(ImGuiCol_Button, ImVec4{ 0.2f, 0.7f, 0.2f, 1.0f });
00198         ImGui::PushStyleColor(ImGuiCol_ButtonHovered, ImVec4{ 0.3f, 0.8f, 0.3f, 1.0f });
00199         ImGui::PushStyleColor(ImGuiCol_ButtonActive, ImVec4{ 0.2f, 0.7f, 0.2f, 1.0f });
00200         ImGui::PushFont(boldFont);
00201         if (ImGui::Button("Y", buttonSize))
00202             values.y = resetValue;
00203         ImGui::PopFont();
00204         ImGui::PopStyleColor(3);
00205 
00206         ImGui::SameLine();
00207         ImGui::PushItemWidth(itemWidth);
00208         ImGui::DragFloat("##Y", &values.y, 0.1f, 0.0f, 0.0f, "%.2f");
00209         ImGui::PopItemWidth();
00210         ImGui::SameLine();
00211 
00212         ImGui::PushStyleColor(ImGuiCol_Button, ImVec4{ 0.1f, 0.25f, 0.8f, 1.0f });
00213         ImGui::PushStyleColor(ImGuiCol_ButtonHovered, ImVec4{ 0.2f, 0.35f, 0.9f, 1.0f });
00214         ImGui::PushStyleColor(ImGuiCol_ButtonActive, ImVec4{ 0.1f, 0.25f, 0.8f, 1.0f });
00215         ImGui::PushFont(boldFont);
00216         if (ImGui::Button("Z", buttonSize))
00217             values.z = resetValue;
00218         ImGui::PopFont();
00219         ImGui::PopStyleColor(3);
00220 
00221         ImGui::SameLine();
00222         ImGui::PushItemWidth(itemWidth);
00223         ImGui::DragFloat("##Z", &values.z, 0.1f, 0.0f, 0.0f, "%.2f");
00224         ImGui::PopItemWidth();
00225 
00226         ImGui::PopStyleVar();
00227 
00228         ImGui::Columns(1);
00229 
00230         ImGui::PopID();
00231     }
```

8.2.5.8 format_as()

```
std::string Vesper::format_as (
            const Event & e)  [inline]
00075     {
00076         return e.ToString();
00077     }
```

8.2.5.9 GLFWErrorCallback()

```
void Vesper::GLFWErrorCallback (
            int error,
            const char * description)  [static]
00016     {
00017         VZ_CORE_ERROR("GLFW Error ({0}): {1}", error, description);
00018     }
```

Referenced by Vesper::WindowsWindow::Init().

8.2.5.10 SerializeEntity()

```
void Vesper::SerializeEntity (
            YAML::Emitter & out,
            Entity entity)  [static]
```

```
00121                                                                   {
00122
00123         VZ_CORE_ASSERT(entity.HasComponent<UUIDComponent>(), "Entity has no UUIDComponent!");
00124         VZ_CORE_ASSERT(entity.HasComponent<NameComponent>(), "Entity has no NameComponent!");
00125
00126         out « YAML::BeginMap; // Entity
00127         out « YAML::Key « "Entity" « YAML::Value « entity.GetID(); // UUIDComponent
00128         out « YAML::Key « "NameComponent" « YAML::Value « entity.GetName();
00129
00130         if (entity.HasComponent<TransformComponent>()) {
00131             out « YAML::Key « "TransformComponent";
00132             out « YAML::BeginMap; // TransformComponent
00133
00134             auto& tc = entity.GetComponent<TransformComponent>();
00135             out « YAML::Key « "Translation" « YAML::Value « tc.Translation;
00136             out « YAML::Key « "Rotation" « YAML::Value « tc.Rotation;
00137             out « YAML::Key « "Scale" « YAML::Value « tc.Scale;
00138
00139             out « YAML::EndMap; // TransformComponent
00140         }
00141
00142         if (entity.HasComponent<CameraComponent>()) {
00143             out « YAML::Key « "CameraComponent";
00144             out « YAML::BeginMap; // CameraComponent
00145
00146             auto& cameraComp = entity.GetComponent<CameraComponent>();
00147             auto& camera = cameraComp.Camera;
00148
00149
00150             out « YAML::Key « "Camera" « YAML::Value;
00151             out « YAML::BeginMap; // Camera
00152             out « YAML::Key « "PerspectiveFOV" « YAML::Value « camera.GetPerspectiveVerticalFOV();
00153             out « YAML::Key « "PerspectiveNear" « YAML::Value « camera.GetPerspectiveNearClip();
00154             out « YAML::Key « "PerspectiveFar" « YAML::Value « camera.GetPerspectiveFarClip();
00155             out « YAML::Key « "OrthographicSize" « YAML::Value « camera.GetOrthographicSize();
00156             out « YAML::Key « "OrthographicNear" « YAML::Value « camera.GetOrthographicNearClip();
00157             out « YAML::Key « "OrthographicFar" « YAML::Value « camera.GetOrthographicFarClip();
00158             out « YAML::EndMap;
00159
00160             out « YAML::Key « "Primary" « YAML::Value « cameraComp.Primary;
00161             out « YAML::Key « "ProjectionType" « YAML::Value « (int)camera.GetProjectionType();
00162             out « YAML::Key « "FixedAspectRatio" « YAML::Value « cameraComp.FixedAspectRatio;
00163
00164             out « YAML::EndMap; // CameraComponent
00165         }
00166
00167         if (entity.HasComponent<SpriteRendererComponent>()) {
00168             out « YAML::Key « "SpriteRendererComponent";
00169             out « YAML::BeginMap; // SpriteRendererComponent
00170
00171             auto& src = entity.GetComponent<SpriteRendererComponent>();
00172             out « YAML::Key « "Color" « YAML::Value « src.Color;
00173             // Texture serialization can be added here in the future
00174             out « YAML::EndMap; // SpriteRendererComponent
00175         }
00176         out « YAML::EndMap; // Entity
00177     }
```

8.2.5.11 ShaderDataTypeSize()

```
uint32_t Vesper::ShaderDataTypeSize (
             ShaderDataType type)  [static]
00013                                                                   {
00014         switch (type) {
00015         case ShaderDataType::Float:     return 4;
00016         case ShaderDataType::Float2:    return 4 * 2;
00017         case ShaderDataType::Float3:    return 4 * 3;
00018         case ShaderDataType::Float4:    return 4 * 4;
00019         case ShaderDataType::Mat3:      return 4 * 3 * 3;
00020         case ShaderDataType::Mat4:      return 4 * 4 * 4;
00021         case ShaderDataType::Int:       return 4;
00022         case ShaderDataType::Int2:      return 4 * 2;
00023         case ShaderDataType::Int3:      return 4 * 3;
00024         case ShaderDataType::Int4:      return 4 * 4;
00025         case ShaderDataType::Bool:      return 1;
00026         }
00027         VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00028         return 0;
00029     }
```

References Bool, Float, Float2, Float3, Float4, Int, Int2, Int3, Int4, Mat3, and Mat4.

Referenced by Vesper::BufferElement::BufferElement().

8.2.5.12 ShaderDataTypeToOpenGLBaseType()

```
GLenum Vesper::ShaderDataTypeToOpenGLBaseType (
            ShaderDataType type)  [static]
```

```
00010     {
00011         switch (type)
00012         {
00013         case ShaderDataType::Float:     return GL_FLOAT;
00014         case ShaderDataType::Float2:    return GL_FLOAT;
00015         case ShaderDataType::Float3:    return GL_FLOAT;
00016         case ShaderDataType::Float4:    return GL_FLOAT;
00017         case ShaderDataType::Mat3:      return GL_FLOAT;
00018         case ShaderDataType::Mat4:      return GL_FLOAT;
00019         case ShaderDataType::Int:       return GL_INT;
00020         case ShaderDataType::Int2:      return GL_INT;
00021         case ShaderDataType::Int3:      return GL_INT;
00022         case ShaderDataType::Int4:      return GL_INT;
00023         case ShaderDataType::Bool:      return GL_BOOL;
00024         }
00025         VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00026         return 0;
00027     }
```

8.2.5.13 ShaderTypeFromString()

```
GLenum Vesper::ShaderTypeFromString (
            const std::string & type)  [static]
```

```
00011     {
00012         VZ_PROFILE_FUNCTION();
00013         if (type == "vertex")
00014             return GL_VERTEX_SHADER;
00015         if (type == "fragment" || type == "pixel")
00016             return GL_FRAGMENT_SHADER;
00017         VZ_CORE_ASSERT(false, "Unknown shader type!");
00018         return 0;
00019     }
```

8.2.5.14 SubTextureEdit()

```
void Vesper::SubTextureEdit (
            const std::string & label,
            SubTextureComponent & subTexture)  [static]
```

```
00298     {
00299         ImGui::Text(label.c_str());
00300
00301         auto& subTexRef = subTexture.GetSubTexture();
00302         if (subTexRef && subTexRef->GetTexture()) {
00303             glm::vec2 oldOffset = subTexture.Offset;
00304             glm::vec2 oldTiling = subTexture.TilingFactor;
00305
00306             DrawVec2Control("Offset", subTexture.Offset);
00307             DrawVec2Control("Scale", subTexture.TilingFactor, 1.0f);
00308
00309             if (subTexRef->GetTexture())
00310             {
00311                 //if (oldOffset != subTexture.Offset || oldTiling != subTexture.TilingFactor) {
00312                 //  subTexture.SetOffset(subTexture.Offset);
00313                 //  subTexture.SetTilingFactor(subTexture.TilingFactor);
00314
00315                     auto tex = subTexRef->GetTexture();
00316                     if (tex) {
00317                         subTexture.SubTexture = SubTexture2D::CreateFromCoords(
00318                             tex, subTexture.Offset,
00319                             glm::vec2(static_cast<float>(tex->GetWidth()) * subTexture.TilingFactor.x,
00320                                 static_cast<float>(tex->GetHeight()) * subTexture.TilingFactor.y));
00321                     }
00322                 //}
00323             }
00324         }
00325         else {
00326             ImGui::Text("No texture assigned.");
00327         }
00328
00329     }
```

8.2.6 Variable Documentation

8.2.6.1 s_Data

```
Renderer2DData Vesper::s_Data  [static]
```

Referenced by Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::DrawQuad(), Vesper::Renderer2D::DrawQuadRotated(), Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::DrawQu, Vesper::Renderer2D::EndScene(), Vesper::Renderer2D::Flush(), Vesper::Renderer2D::FlushAndReset(), Vesper::Renderer2D::GetSt, Vesper::Renderer2D::GetWhiteTexture(), Vesper::Renderer2D::Init(), Vesper::Renderer2D::ResetStats(), Vesper::Renderer2D::Shutd, and Vesper::Renderer2D::StartBatch().

8.2.6.2 s_GLFWInitialized

```
bool Vesper::s_GLFWInitialized = false  [static]
```

Referenced by Vesper::WindowsWindow::Init().

8.2.6.3 s_MaxFramebufferSize

```
const uint32_t Vesper::s_MaxFramebufferSize = 8192  [static]
```

TODO: Get the actual maximum size from the GPU!

8.3 Vesper::Color Namespace Reference

Functions

- static glm::vec4 White ()
- static glm::vec4 Black ()
- static glm::vec4 Gray ()
- static glm::vec4 Red ()
- static glm::vec4 Orange ()
- static glm::vec4 Yellow ()
- static glm::vec4 Green ()
- static glm::vec4 Blue ()
- static glm::vec4 Indigo ()
- static glm::vec4 Purple ()
- static glm::vec4 Cyan ()
- static glm::vec4 Magenta ()
- static glm::vec4 Pink ()
- static glm::vec4 Brown ()
- static glm::vec4 Transparent ()
- static glm::vec4 StripAlpha (const glm::vec4 &color)
- static glm::vec4 SetAlpha (const glm::vec4 &color, float alpha=0.0f)

8.3.1 Function Documentation

8.3.1.1 Black()

```
glm::vec4 Vesper::Color::Black ()  [static]
00010 { return glm::vec4(0.0f, 0.0f, 0.0f, 1.0f); }
```

8.3.1.2 Blue()

```
glm::vec4 Vesper::Color::Blue ()  [static]
00017 { return glm::vec4(0.0f, 0.0f, 1.0f, 1.0f); }
```

8.3.1.3 Brown()

```
glm::vec4 Vesper::Color::Brown ()  [static]
00024 { return glm::vec4(0.6f, 0.4f, 0.2f, 1.0f); }
```

8.3.1.4 Cyan()

```
glm::vec4 Vesper::Color::Cyan ()  [static]
00021 { return glm::vec4(0.0f, 1.0f, 1.0f, 1.0f); }
```

8.3.1.5 Gray()

```
glm::vec4 Vesper::Color::Gray ()  [static]
00011 { return glm::vec4(0.5f, 0.5f, 0.5f, 1.0f); }
```

8.3.1.6 Green()

```
glm::vec4 Vesper::Color::Green ()  [static]
00016 { return glm::vec4(0.0f, 1.0f, 0.0f, 1.0f); }
```

8.3.1.7 Indigo()

```
glm::vec4 Vesper::Color::Indigo ()  [static]
00018 { return glm::vec4(0.29f, 0.0f, 0.51f, 1.0f); }
```

8.3.1.8 Magenta()

```
glm::vec4 Vesper::Color::Magenta ()  [static]
00022 { return glm::vec4(1.0f, 0.0f, 1.0f, 1.0f); }
```

8.3.1.9 Orange()

```
glm::vec4 Vesper::Color::Orange ()  [static]
00014 { return glm::vec4(1.0f, 0.5f, 0.0f, 1.0f); }
```

8.3.1.10 Pink()

```
glm::vec4 Vesper::Color::Pink ()  [static]
00023 { return glm::vec4(1.0f, 0.75f, 0.8f, 1.0f); }
```

8.3.1.11 Purple()

```
glm::vec4 Vesper::Color::Purple ()  [static]
00019 { return glm::vec4(0.5f, 0.0f, 0.5f, 1.0f); }
```

8.3.1.12 Red()

```
glm::vec4 Vesper::Color::Red ()  [static]
00013 { return glm::vec4(1.0f, 0.0f, 0.0f, 1.0f); }
```

8.3.1.13 SetAlpha()

```
glm::vec4 Vesper::Color::SetAlpha (
            const glm::vec4 & color,
            float alpha = 0.0f)  [static]
00028 {   return glm::vec4(color.x, color.y, color.z, alpha);}
```

8.3.1.14 StripAlpha()

```
glm::vec4 Vesper::Color::StripAlpha (
            const glm::vec4 & color)  [static]
00027 {   return glm::vec4(color.x, color.y, color.z, 1.0f);  }
```

8.3.1.15 Transparent()

```
glm::vec4 Vesper::Color::Transparent ()  [static]
00025 { return glm::vec4(0.0f, 0.0f, 0.0f, 0.0f); }
```

8.3.1.16 White()

```
glm::vec4 Vesper::Color::White ()  [static]
00009 { return glm::vec4(1.0f, 1.0f, 1.0f, 1.0f); }
```

8.3.1.17 Yellow()

```
glm::vec4 Vesper::Color::Yellow ()  [static]
00015 { return glm::vec4(1.0f, 1.0f, 0.0f, 1.0f); }
```

8.4 Vesper::Math Namespace Reference

Functions

- bool DecomposeTransform (const glm::mat4 &transform, glm::vec3 &translation, glm::vec3 &rotation, glm::vec3 &scale)

8.4.1 Function Documentation

8.4.1.1 DecomposeTransform()

```
bool Vesper::Math::DecomposeTransform (
            const glm::mat4 & transform,
            glm::vec3 & translation,
            glm::vec3 & rotation,
            glm::vec3 & scale)
00011     {
00012         // From glm::decompose in matrix_decompose.inl
00013 
00014         using namespace glm;
00015         using T = float;
00016 
00017         mat4 LocalMatrix(transform);
00018 
00019         // Normalize the matrix.
00020         if (epsilonEqual(LocalMatrix[3][3], static_cast<float>(0), epsilon<T>()))
00021             return false;
00022 
00023         // First, isolate perspective.  This is the messiest.
00024         if (
00025             epsilonNotEqual(LocalMatrix[0][3], static_cast<T>(0), epsilon<T>()) ||
00026             epsilonNotEqual(LocalMatrix[1][3], static_cast<T>(0), epsilon<T>()) ||
00027             epsilonNotEqual(LocalMatrix[2][3], static_cast<T>(0), epsilon<T>()))
00028         {
00029             // Clear the perspective partition
00030             LocalMatrix[0][3] = LocalMatrix[1][3] = LocalMatrix[2][3] = static_cast<T>(0);
00031             LocalMatrix[3][3] = static_cast<T>(1);
00032         }
00033 
00034         // Next take care of translation (easy).
00035         translation = vec3(LocalMatrix[3]);
00036         LocalMatrix[3] = vec4(0, 0, 0, LocalMatrix[3].w);
00037 
00038         vec3 Row[3], Pdum3;
00039 
00040         // Now get scale and shear.
00041         for (length_t i = 0; i < 3; ++i)
00042             for (length_t j = 0; j < 3; ++j)
00043                 Row[i][j] = LocalMatrix[i][j];
00044 
00045         // Compute X scale factor and normalize first row.
00046         scale.x = length(Row[0]);
00047         Row[0] = detail::scale(Row[0], static_cast<T>(1));
00048         scale.y = length(Row[1]);
00049         Row[1] = detail::scale(Row[1], static_cast<T>(1));
00050         scale.z = length(Row[2]);
00051         Row[2] = detail::scale(Row[2], static_cast<T>(1));
00052 
00053         // At this point, the matrix (in rows[]) is orthonormal.
00054         // Check for a coordinate system flip.  If the determinant
00055         // is -1, then negate the matrix and the scaling factors.
00056 #if 0
```

```
00057         Pdum3 = cross(Row[1], Row[2]); // v3Cross(row[1], row[2], Pdum3);
00058         if (dot(Row[0], Pdum3) < 0)
00059         {
00060             for (length_t i = 0; i < 3; i++)
00061             {
00062                 scale[i] *= static_cast<T>(-1);
00063                 Row[i] *= static_cast<T>(-1);
00064             }
00065         }
00066 #endif
00067
00068         rotation.y = asin(-Row[0][2]);
00069         if (cos(rotation.y) != 0) {
00070             rotation.x = atan2(Row[1][2], Row[2][2]);
00071             rotation.z = atan2(Row[0][1], Row[0][0]);
00072         }
00073         else {
00074             rotation.x = atan2(-Row[2][0], Row[1][1]);
00075             rotation.z = 0;
00076         }
00077
00078
00079         return true;
00080     }
```

8.5 Vesper::Random Namespace Reference

Functions

- std::mt19937 & GetRNG ()
- void Seed (uint32_t seed)
- uint32_t UInt1 (uint32_t max)
- bool Bool1 (float trueChance)
- unsigned char Char ()
- std::string String (size_t length)
- std::string HexString (size_t length)
- std::string UUID ()
- float Float1 ()
- float RangeF1 (float min, float max)
- float RangeF1_Inclusive (float min, float max)
- glm::vec2 Float2 ()
- glm::vec2 RangeF2 (float min, float max)
- glm::vec2 RangeF2 (float min1, float max1, float min2, float max2)
- glm::vec2 RangeF2 (const glm::vec2 &minRange, const glm::vec2 &maxRange)
- glm::vec3 Float3 ()
- glm::vec3 RangeF3 (float min, float max)
- glm::vec3 RangeF3 (float min1, float max1, float min2, float max2, float min3, float max3)
- glm::vec3 RangeF3 (const glm::vec2 &range1, const glm::vec2 &range2, const glm::vec2 &range3)
- glm::vec4 Float4 ()
- glm::vec4 RangeF4 (float min, float max)

8.5.1 Function Documentation

8.5.1.1 Bool1()

```
bool Vesper::Random::Bool1 (
            float trueChance)  [inline]
00027                                           {
00028             VZ_PROFILE_FUNCTION();
00029             std::uniform_real_distribution<float> dist(0.0f, 1.0f);
00030             return dist(GetRNG()) < trueChance;
00031         }
```

8.5.1.2 Char()

```
unsigned char Vesper::Random::Char ()  [inline]
```

```
00033                                    {
00034             VZ_PROFILE_FUNCTION();
00035             std::uniform_int_distribution<int> dist(0, 255);
00036             return static_cast<unsigned char>(dist(GetRNG()));
00037         }
```

8.5.1.3 Float1()

```
float Vesper::Random::Float1 ()  [inline]
```

```
00072                                {
00073             VZ_PROFILE_FUNCTION();
00074             static thread_local std::uniform_real_distribution<float> dist(0.0f, 1.0f);
00075             return dist(GetRNG());
00076         }
```

Referenced by Vesper::ParticleSystem::Emit().

8.5.1.4 Float2()

```
glm::vec2 Vesper::Random::Float2 ()  [inline]
```

```
00096                                {
00097             return glm::vec2{ Float1(), Float1() };
00098         }
```

8.5.1.5 Float3()

```
glm::vec3 Vesper::Random::Float3 ()  [inline]
```

```
00117                                {
00118             return glm::vec3{ Float1(), Float1(), Float1() };
00119         }
```

8.5.1.6 Float4()

```
glm::vec4 Vesper::Random::Float4 ()  [inline]
```

```
00137                                {
00138             return glm::vec4{ Float1(), Float1(), Float1(), Float1() };
00139         }
```

8.5.1.7 GetRNG()

```
std::mt19937 & Vesper::Random::GetRNG ()  [inline]
```

```
00012                                    {
00013             static thread_local std::mt19937 rng{ std::random_device{}() };
00014             return rng;
00015         }
```

Referenced by Seed().

8.5.1.8 HexString()

```
std::string Vesper::Random::HexString (
            size_t length)  [inline]
```

```
00053                                                {
00054             VZ_PROFILE_FUNCTION();
00055             const char charset[] =
00056                 "0123456789"
00057                 "ABCDEF";
00058             const size_t max_index = (sizeof(charset) - 1);
00059             std::string str(length, 0);
00060             for (size_t i = 0; i < length; ++i) {
00061                 str[i] = charset[UInt1(static_cast<uint32_t>(max_index))];
00062             }
00063             return str;
00064         }
```

References UInt1().

8.5.1.9 RangeF1()

```
float Vesper::Random::RangeF1 (
            float min,
            float max)  [inline]
```

```
00080                                                 {
00081             VZ_PROFILE_FUNCTION();
00082             if (min > max) std::swap(min, max);
00083             std::uniform_real_distribution<float> dist(min, max);
00084             return dist(GetRNG());
00085         }
```

8.5.1.10 RangeF1_Inclusive()

```
float Vesper::Random::RangeF1_Inclusive (
            float min,
            float max)  [inline]
```

```
00088                                                        {
00089             if (min > max) std::swap(min, max);
00090             float upper = std::nextafter(max, std::numeric_limits<float>::infinity());
00091             std::uniform_real_distribution<float> dist(min, upper);
00092             return dist(GetRNG());
00093         }
```

8.5.1.11 RangeF2() [1/3]

```
glm::vec2 Vesper::Random::RangeF2 (
            const glm::vec2 & minRange,
            const glm::vec2 & maxRange)  [inline]
```

```
00111                                                                                {
00112             return glm::vec2{ RangeF1(minRange.x, maxRange.x), RangeF1(minRange.y, maxRange.y) };
00113         }
```

8.5.1.12 RangeF2() [2/3]

```
glm::vec2 Vesper::Random::RangeF2 (
            float min,
            float max)  [inline]
```

```
00101                                                 {
00102             return glm::vec2{ RangeF1(min, max), RangeF1(min, max) };
00103         }
```

8.5.1.13 RangeF2() [3/3]

```
glm::vec2 Vesper::Random::RangeF2 (
            float min1,
            float max1,
            float min2,
            float max2)  [inline]
00106                                                                                {
00107             return glm::vec2{ RangeF1(min1, max1), RangeF1(min2, max2) };
00108         }
```

8.5.1.14 RangeF3() [1/3]

```
glm::vec3 Vesper::Random::RangeF3 (
            const glm::vec2 & range1,
            const glm::vec2 & range2,
            const glm::vec2 & range3)  [inline]
00132                                                                                                                {
00133             return glm::vec3{ RangeF1(range1.x, range1.y), RangeF1(range2.x, range2.y),
      RangeF1(range3.x, range3.y) };
00134         }
```

8.5.1.15 RangeF3() [2/3]

```
glm::vec3 Vesper::Random::RangeF3 (
            float min,
            float max)  [inline]
00122                                                {
00123             return glm::vec3{ RangeF1(min, max), RangeF1(min, max), RangeF1(min, max) };
00124         }
```

8.5.1.16 RangeF3() [3/3]

```
glm::vec3 Vesper::Random::RangeF3 (
            float min1,
            float max1,
            float min2,
            float max2,
            float min3,
            float max3)  [inline]
00127
      {
00128             return glm::vec3{ RangeF1(min1, max1), RangeF1(min2, max2), RangeF1(min3, max3) };
00129         }
```

8.5.1.17 RangeF4()

```
glm::vec4 Vesper::Random::RangeF4 (
            float min,
            float max)  [inline]
00142                                                {
00143             return glm::vec4{ RangeF1(min, max), RangeF1(min, max), RangeF1(min, max), RangeF1(min,
      max) };
00144         }
```

8.5.1.18 Seed()

```
void Vesper::Random::Seed (
            uint32_t seed)  [inline]
00017                                       {
00018             GetRNG().seed(seed);
00019         }
```

References GetRNG().

8.5.1.19 String()

```
std::string Vesper::Random::String (
            size_t length)  [inline]
00039                                              {
00040             VZ_PROFILE_FUNCTION();
00041             const char charset[] =
00042                 "0123456789"
00043                 "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
00044                 "abcdefghijklmnopqrstuvwxyz";
00045             const size_t max_index = (sizeof(charset) - 1);
00046             std::string str(length, 0);
00047             for (size_t i = 0; i < length; ++i) {
00048                 str[i] = charset[UInt1(static_cast<uint32_t>(max_index))];
00049             }
00050             return str;
00051         }
```

References UInt1().

8.5.1.20 UInt1()

```
uint32_t Vesper::Random::UInt1 (
            uint32_t max)  [inline]
00021                                           {
00022             VZ_PROFILE_FUNCTION();
00023             std::uniform_int_distribution<uint32_t> dist(0, max - 1);
00024             return dist(GetRNG());
00025         }
```

Referenced by HexString(), and String().

8.5.1.21 UUID()

```
std::string Vesper::Random::UUID ()  [inline]
00066                                  {
00067             VZ_PROFILE_FUNCTION();
00068             return HexString(8) + "-" + HexString(4) + "-" + HexString(4) + "-" + HexString(4) + "-" +
      HexString(12);
00069         }
```

8.6 YAML Namespace Reference

Classes

- struct convert< glm::vec2 >
- struct convert< glm::vec3 >
- struct convert< glm::vec4 >

Functions

- YAML::Emitter & operator<< (YAML::Emitter &out, const glm::vec2 &v)
- YAML::Emitter & operator<< (YAML::Emitter &out, const glm::vec3 &v)
- YAML::Emitter & operator<< (YAML::Emitter &out, const glm::vec4 &v)

8.6.1 Function Documentation

8.6.1.1 operator<<() [1/3]

```
YAML::Emitter & YAML::operator<< (
            YAML::Emitter & out,
            const glm::vec2 & v)
```

```
00091     {
00092         out « YAML::Flow;
00093         out « YAML::BeginSeq « v.x « v.y « YAML::EndSeq;
00094         return out;
00095     }
```

8.6.1.2 operator<<() [2/3]

```
YAML::Emitter & YAML::operator<< (
            YAML::Emitter & out,
            const glm::vec3 & v)
```

```
00098     {
00099         out « YAML::Flow;
00100         out « YAML::BeginSeq « v.x « v.y « v.z « YAML::EndSeq;
00101         return out;
00102     }
```

8.6.1.3 operator<<() [3/3]

```
YAML::Emitter & YAML::operator<< (
            YAML::Emitter & out,
            const glm::vec4 & v)
```

```
00105     {
00106         out « YAML::Flow;
00107         out « YAML::BeginSeq « v.x « v.y « v.z « v.w « YAML::EndSeq;
00108         return out;
00109     }
```

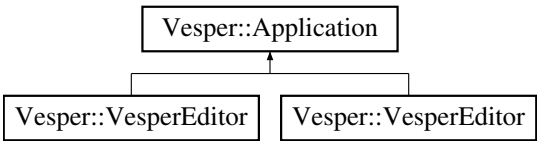
Chapter 9

Class Documentation

9.1 Vesper::Application Class Reference

`#include <Application.h>`

Inheritance diagram for Vesper::Application:



Public Member Functions

- Application (const std::string &name="")
- virtual ~Application ()
- void Run ()
- void OnEvent (Event &e)
- void PushLayer (Layer *layer)
- void PushOverlay (Layer *overlay)
- void Close ()
- ImGuiLayer * GetImGuiLayer ()
- Window & GetWindow ()

Static Public Member Functions

- static Application & Get ()

Private Member Functions

- bool OnWindowClose (WindowCloseEvent &e)
- bool OnWindowResize (WindowResizeEvent &e)

Private Attributes

- Scope< Window > m_Window
- ImGuiLayer * m_ImGuiLayer
- bool m_Running = true
- bool m_Minimized = false
- LayerStack m_LayerStack
- float m_LastFrameTime = 0.0f

Static Private Attributes

- static Application * s_Instance = nullptr

9.1.1 Constructor & Destructor Documentation

9.1.1.1 Application()

```
Vesper::Application::Application (
            const std::string & name = "")
```

```
00018     {
00019         VZ_PROFILE_FUNCTION();
00020
00021         VZ_CORE_ASSERT(!s_Instance, "Application already exists!");
00022         s_Instance = this;
00023
00024         m_Window = Window::Create(WindowProps(name));
00025         m_Window->SetEventCallback(BIND_EVENT_FN(OnEvent));
00026         m_Window->SetVSync(false);
00027
00028
00029         Renderer::Init();
00030
00031
00032         m_ImGuiLayer = new ImGuiLayer();
00033         PushOverlay(m_ImGuiLayer);
00034
00035     }
```

References Vesper::ImGuiLayer::ImGuiLayer(), Vesper::Renderer::Init(), m_ImGuiLayer, PushOverlay(), and s_Instance.

9.1.1.2 ~Application()

```
Vesper::Application::~Application ()  [virtual]
```

```
00038     {
00039     }
```

9.1.2 Member Function Documentation

9.1.2.1 Close()

```
void Vesper::Application::Close ()
```

```
00056     {
00057         m_Running = false;
00058     }
```

References m_Running.

Referenced by Vesper::EditorLayer::OnImGuiRender().

9.1.2.2 Get()

```
Application & Vesper::Application::Get ()  [inline], [static]
00052 { return *s_Instance; }
```

References s_Instance.

Referenced by Vesper::ImGuiLayer::End(), Vesper::Input::GetMousePosition(), Vesper::Input::IsKeyPressed(), Vesper::Input::IsMouseButtonPressed(), Vesper::ImGuiLayer::OnAttach(), and Vesper::EditorLayer::OnImGuiRender().

9.1.2.3 GetImGuiLayer()

```
ImGuiLayer * Vesper::Application::GetImGuiLayer ()  [inline]
00050 { return m_ImGuiLayer; }
```

References m_ImGuiLayer.

Referenced by Vesper::EditorLayer::OnImGuiRender().

9.1.2.4 GetWindow()

```
Window & Vesper::Application::GetWindow ()  [inline]
00053 { return *m_Window; }
```

Referenced by Vesper::ImGuiLayer::End(), Vesper::Input::GetMousePosition(), Vesper::Input::IsKeyPressed(), and Vesper::Input::IsMouseButtonPressed().

9.1.2.5 OnEvent()

```
void Vesper::Application::OnEvent (
            Event & e)
00061     {
00062         VZ_PROFILE_FUNCTION();
00063         EventDispatcher dispatcher(e);
00064         dispatcher.Dispatch<WindowCloseEvent>(BIND_EVENT_FN(OnWindowClose));
00065         dispatcher.Dispatch<WindowResizeEvent>(BIND_EVENT_FN(OnWindowResize));
00066
00067         for (auto it = m_LayerStack.rbegin(); it != m_LayerStack.rend(); ++it)
00068         {
00069             if (e.Handled)
00070                 break;
00071             (*it)->OnEvent(e);
00072         }
00073
00074     }
```

References Vesper::EventDispatcher::EventDispatcher(), OnWindowClose(), and OnWindowResize().

9.1.2.6 OnWindowClose()

```
bool Vesper::Application::OnWindowClose (
            WindowCloseEvent & e)  [private]
00112     {
00113         VZ_PROFILE_FUNCTION();
00114         m_Running = false;
00115         return true;
00116     }
```

References m_Running.

Referenced by OnEvent().

9.1.2.7 OnWindowResize()

```
bool Vesper::Application::OnWindowResize (
            WindowResizeEvent & e)  [private]
```

```
00119     {
00120         VZ_PROFILE_FUNCTION();
00121         if (e.GetWidth() == 0 || e.GetHeight() == 0)
00122         {
00123             m_Minimized = true;
00124             return false;
00125         }
00126 
00127         m_Minimized = false;
00128         Renderer::OnWindowResize(e.GetWidth(), e.GetHeight());
00129         return false;
00130     }
```

References Vesper::WindowResizeEvent::GetHeight(), Vesper::WindowResizeEvent::GetWidth(), m_Minimized, and Vesper::Renderer::OnWindowResize().

Referenced by OnEvent().

9.1.2.8 PushLayer()

```
void Vesper::Application::PushLayer (
            Layer * layer)
```

```
00042     {
00043         VZ_PROFILE_FUNCTION();
00044         m_LayerStack.PushLayer(layer);
00045         layer->OnAttach();
00046     }
```

References Vesper::Layer::OnAttach().

Referenced by Vesper::VesperEditor::VesperEditor().

9.1.2.9 PushOverlay()

```
void Vesper::Application::PushOverlay (
            Layer * overlay)
```

```
00049     {
00050         VZ_PROFILE_FUNCTION();
00051         m_LayerStack.PushOverlay(overlay);
00052         overlay->OnAttach();
00053     }
```

References Vesper::Layer::OnAttach().

Referenced by Application().

9.1.2.10 Run()

```
void Vesper::Application::Run ()
```

```
00078     {
00079         VZ_PROFILE_FUNCTION();
00080         while (m_Running)
00081         {
00082             VZ_PROFILE_SCOPE("RunLoop");
00083             float time = (float)glfwGetTime(); // TODO: Platform::GetTime()
00084             Timestep timestep = time - m_LastFrameTime;
00085             m_LastFrameTime = time;
00086 
00087             if (!m_Minimized)
00088             {
00089                 VZ_PROFILE_SCOPE("LayerStack OnUpdate");
00090                 // Update layers
00091                 for (auto layer : m_LayerStack)
00092                     layer->OnUpdate(timestep);
00093             }
00094 
00095             {
00096                 VZ_PROFILE_SCOPE("ImGuiLayer OnImGuiRender");
00097                 m_ImGuiLayer->Begin();
00098                 for (auto layer : m_LayerStack)
00099                     layer->OnImGuiRender();
00100                 m_ImGuiLayer->End();
00101             }
00102 
00103             {
00104                 VZ_PROFILE_SCOPE("Window OnUpdate");
00105                 // Update window second
00106                 m_Window->OnUpdate();
00107             }
00108         };
00109     }
```

References Vesper::ImGuiLayer::Begin(), Vesper::ImGuiLayer::End(), m_ImGuiLayer, m_LastFrameTime, m_Minimized, and m_Running.

9.1.3 Member Data Documentation

9.1.3.1 m_ImGuiLayer

```
ImGuiLayer* Vesper::Application::m_ImGuiLayer  [private]
```

Referenced by Application(), GetImGuiLayer(), and Run().

9.1.3.2 m_LastFrameTime

```
float Vesper::Application::m_LastFrameTime = 0.0f  [private]
```

Referenced by Run().

9.1.3.3 m_LayerStack

```
LayerStack Vesper::Application::m_LayerStack  [private]
```

9.1.3.4 m_Minimized

```
bool Vesper::Application::m_Minimized = false  [private]
```

Referenced by OnWindowResize(), and Run().

9.1.3.5 m_Running

```
bool Vesper::Application::m_Running = true  [private]
```

Referenced by Close(), OnWindowClose(), and Run().

9.1.3.6 m_Window

```
Scope<Window> Vesper::Application::m_Window  [private]
```

9.1.3.7 s_Instance

```
Application * Vesper::Application::s_Instance = nullptr  [static], [private]
```

Referenced by Application(), and Get().

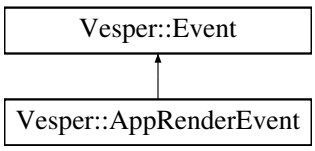
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/App/Application.h
- Vesper/src/Vesper/App/Application.cpp

9.2 Vesper::AppRenderEvent Class Reference

```
#include <ApplicationEvent.h>
```

Inheritance diagram for Vesper::AppRenderEvent:



Public Member Functions

- AppRenderEvent ()

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- virtual std::string ToString () const
- bool IsInCategory (EventCategory category)

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.2.1 Constructor & Destructor Documentation

9.2.1.1 AppRenderEvent()

```
Vesper::AppRenderEvent::AppRenderEvent () [inline]
00062 {}
```

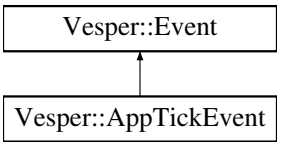
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/ApplicationEvent.h

9.3 Vesper::AppTickEvent Class Reference

```
#include <ApplicationEvent.h>
```

Inheritance diagram for Vesper::AppTickEvent:



Public Member Functions

- AppTickEvent ()

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- virtual std::string ToString () const
- bool IsInCategory (EventCategory category)

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.3.1 Constructor & Destructor Documentation

9.3.1.1 AppTickEvent()

```
Vesper::AppTickEvent::AppTickEvent () [inline]
00044 {}
```

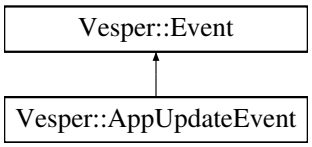
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/ApplicationEvent.h

9.4 Vesper::AppUpdateEvent Class Reference

```
#include <ApplicationEvent.h>
```

Inheritance diagram for Vesper::AppUpdateEvent:



Public Member Functions

- AppUpdateEvent ()

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- virtual std::string ToString () const
- bool IsInCategory (EventCategory category)

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.4.1 Constructor & Destructor Documentation

9.4.1.1 AppUpdateEvent()

```
Vesper::AppUpdateEvent::AppUpdateEvent ()  [inline]
00053 {}
```

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

- Vesper/src/Vesper/Events/ApplicationEvent.h

9.5 Vesper::BufferElement Struct Reference

```
#include <Buffer.h>
```

Public Member Functions

- BufferElement ()
- BufferElement (ShaderDataType type, const std::string &name, bool normalized=false)
- uint32_t GetComponentCount () const

Public Attributes

- std::string Name
- ShaderDataType Type
- uint32_t Size
- uint32_t Offset
- bool Normalized

9.5.1 Constructor & Destructor Documentation

9.5.1.1 BufferElement() [1/2]

```
Vesper::BufferElement::BufferElement ()  [inline]
00039 {}
```

9.5.1.2 BufferElement() [2/2]

```
Vesper::BufferElement::BufferElement (
            ShaderDataType type,
            const std::string & name,
            bool normalized = false)  [inline]
00041             : Name(name), Type(type), Size(ShaderDataTypeSize(type)), Offset(0),
      Normalized(normalized)
00042         {
00043         }
```

References BufferElement(), Normalized, Offset, Vesper::ShaderDataTypeSize(), Size, and Type.

Referenced by BufferElement().

9.5.2 Member Function Documentation

9.5.2.1 GetComponentCount()

```
uint32_t Vesper::BufferElement::GetComponentCount () const  [inline]
```

```
00045                                           {
00046             switch (Type) {
00047             case ShaderDataType::Float:     return 1;
00048             case ShaderDataType::Float2:    return 2;
00049             case ShaderDataType::Float3:    return 3;
00050             case ShaderDataType::Float4:    return 4;
00051             case ShaderDataType::Mat3:      return 3 * 3;
00052             case ShaderDataType::Mat4:      return 4 * 4;
00053             case ShaderDataType::Int:       return 1;
00054             case ShaderDataType::Int2:      return 2;
00055             case ShaderDataType::Int3:      return 3;
00056             case ShaderDataType::Int4:      return 4;
00057             case ShaderDataType::Bool:      return 1;
00058             }
00059             VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00060             return 0;
00061         }
```

References Vesper::Bool, Vesper::Float, Vesper::Float2, Vesper::Float3, Vesper::Float4, Vesper::Int, Vesper::Int2, Vesper::Int3, Vesper::Int4, Vesper::Mat3, Vesper::Mat4, and Type.

9.5.3 Member Data Documentation

9.5.3.1 Name

```
std::string Vesper::BufferElement::Name
```

9.5.3.2 Normalized

```
bool Vesper::BufferElement::Normalized
```

Referenced by BufferElement().

9.5.3.3 Offset

```
uint32_t Vesper::BufferElement::Offset
```

Referenced by BufferElement().

9.5.3.4 Size

```
uint32_t Vesper::BufferElement::Size
```

Referenced by BufferElement().

9.5.3.5 Type

```
ShaderDataType Vesper::BufferElement::Type
```

Referenced by BufferElement(), and GetComponentCount().

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

- Vesper/src/Vesper/Renderer/Buffer.h

9.6 Vesper::BufferLayout Class Reference

```
#include <Buffer.h>
```

Public Member Functions

- BufferLayout ()
- BufferLayout (const std::initializer_list< BufferElement > &elements)
- const std::vector< BufferElement > & GetElements () const
- uint32_t GetStride () const
- std::vector< BufferElement >::iterator begin ()
- std::vector< BufferElement >::const_iterator begin () const
- std::vector< BufferElement >::iterator end ()
- std::vector< BufferElement >::const_iterator end () const

Private Member Functions

- void CalculateOffsetsAndStride ()

Private Attributes

- std::vector< BufferElement > m_Elements
- uint32_t m_Stride = 0

9.6.1 Constructor & Destructor Documentation

9.6.1.1 BufferLayout() [1/2]

```
Vesper::BufferLayout::BufferLayout ()  [inline]
00067 {}
```

9.6.1.2 BufferLayout() [2/2]

```
Vesper::BufferLayout::BufferLayout (
            const std::initializer_list< BufferElement > & elements)  [inline]
00069           : m_Elements(elements), m_Stride(0)
00070       {
00071           CalculateOffsetsAndStride();
00072       }
```

References BufferLayout(), and m_Stride.

Referenced by BufferLayout().

9.6.2 Member Function Documentation

9.6.2.1 begin() [1/2]

```
std::vector< BufferElement >::iterator Vesper::BufferLayout::begin ()  [inline]
00078 { return m_Elements.begin(); }
```

9.6.2.2 begin() [2/2]

```
std::vector< BufferElement >::const_iterator Vesper::BufferLayout::begin () const  [inline]
00079 { return m_Elements.begin(); }
```

9.6.2.3 CalculateOffsetsAndStride()

```
void Vesper::BufferLayout::CalculateOffsetsAndStride ()  [inline], [private]
00086                                       {
00087           uint32_t offset = 0;
00088           m_Stride = 0;
00089           for (auto& element : m_Elements) {
00090               element.Offset = offset;
00091               offset += element.Size;
00092               m_Stride += element.Size;
00093           }
00094       }
```

9.6.2.4 end() [1/2]

```
std::vector< BufferElement >::iterator Vesper::BufferLayout::end ()  [inline]
00080 { return m_Elements.end(); }
```

9.6.2.5 end() [2/2]

```
std::vector< BufferElement >::const_iterator Vesper::BufferLayout::end () const  [inline]
00081 { return m_Elements.end(); }
```

9.6.2.6 GetElements()

```
const std::vector< BufferElement > & Vesper::BufferLayout::GetElements () const  [inline]
00075 { return m_Elements; }
```

9.6.2.7 GetStride()

```
uint32_t Vesper::BufferLayout::GetStride () const  [inline]
00076 { return m_Stride; }
```

References m_Stride.

9.6.3 Member Data Documentation

9.6.3.1 m_Elements

```
std::vector<BufferElement> Vesper::BufferLayout::m_Elements  [private]
```

9.6.3.2 m_Stride

```
uint32_t Vesper::BufferLayout::m_Stride = 0  [private]
```

Referenced by BufferLayout(), and GetStride().

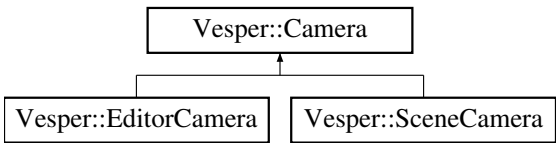
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Renderer/Buffer.h

9.7 Vesper::Camera Class Reference

```
#include <Camera.h>
```

Inheritance diagram for Vesper::Camera:



Public Member Functions

- Camera ()=default
- Camera (const glm::mat4 &projection)
- ~Camera ()=default
- const glm::mat4 & GetProjection () const

Protected Attributes

- glm::mat4 m_Projection = glm::mat4(1.0f)

9.7.1 Constructor & Destructor Documentation

9.7.1.1 Camera() [1/2]

```
Vesper::Camera::Camera ()  [default]
```

9.7.1.2 Camera() [2/2]

```
Vesper::Camera::Camera (
            const glm::mat4 & projection)  [inline]
00011             : m_Projection(projection) {
00012         }
```

References Camera().

Referenced by Camera().

9.7.1.3 ~Camera()

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

9.7.2 Member Function Documentation

9.7.2.1 GetProjection()

```
const glm::mat4 & Vesper::Camera::GetProjection () const  [inline]
00015 { return m_Projection; }
```

9.7.3 Member Data Documentation

9.7.3.1 m_Projection

```
glm::mat4 Vesper::Camera::m_Projection = glm::mat4(1.0f)  [protected]
```

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

- Vesper/src/Vesper/Renderer/Camera.h

9.8 Vesper::CameraComponent Struct Reference

```
#include <Components.h>
```

Public Member Functions

- CameraComponent ()=default
- CameraComponent (const CameraComponent &)=default

Public Attributes

- SceneCamera Camera
- bool Primary = true
- bool FixedAspectRatio = false

9.8.1 Constructor & Destructor Documentation

9.8.1.1 CameraComponent() [1/2]

```
Vesper::CameraComponent::CameraComponent ()  [default]
```

9.8.1.2 CameraComponent() [2/2]

```
Vesper::CameraComponent::CameraComponent (
            const CameraComponent & )  [default]
```

9.8.2 Member Data Documentation

9.8.2.1 Camera

```
SceneCamera Vesper::CameraComponent::Camera
```

9.8.2.2 FixedAspectRatio

```
bool Vesper::CameraComponent::FixedAspectRatio = false
```

9.8.2.3 Primary

```
bool Vesper::CameraComponent::Primary = true
```

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

- Vesper/src/Vesper/Scene/Components.h

9.9 YAML::convert< glm::vec2 > Struct Reference

Static Public Member Functions

- static Node encode (const glm::vec2 &rhs)
- static bool decode (const Node &node, glm::vec2 &rhs)
- static Node encode (const glm::vec2 &rhs)
- static bool decode (const Node &node, glm::vec2 &rhs)

9.9.1 Member Function Documentation

9.9.1.1 decode() [1/2]

```
bool YAML::convert< glm::vec2 >::decode (
            const Node & node,
            glm::vec2 & rhs)  [inline], [static]
```

```
00027         {
00028             if (!node.IsSequence() || node.size() != 2)
00029                 return false;
00030
00031             rhs.x = node[0].as<float>();
00032             rhs.y = node[1].as<float>();
00033             return true;
00034         }
```

9.9.1.2 decode() [2/2]

```
bool YAML::convert< glm::vec2 >::decode (
            const Node & node,
            glm::vec2 & rhs)  [inline], [static]
```

```
00027         {
00028             if (!node.IsSequence() || node.size() != 2)
00029                 return false;
00030
00031             rhs.x = node[0].as<float>();
00032             rhs.y = node[1].as<float>();
00033             return true;
00034         }
```

9.9.1.3 encode() [1/2]

```
Node YAML::convert< glm::vec2 >::encode (
            const glm::vec2 & rhs)  [inline], [static]
```

```
00018         {
00019             Node node;
00020             node.push_back(rhs.x);
00021             node.push_back(rhs.y);
00022             node.SetStyle(EmitterStyle::Flow);
00023             return node;
00024         }
```

9.9.1.4 encode() [2/2]

```
Node YAML::convert< glm::vec2 >::encode (
            const glm::vec2 & rhs)  [inline], [static]
```

```
00018         {
00019             Node node;
00020             node.push_back(rhs.x);
00021             node.push_back(rhs.y);
00022             node.SetStyle(EmitterStyle::Flow);
00023             return node;
00024         }
```

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

- Vesper/src/Vesper/Scene/SceneSerializer.cpp

9.10 YAML::convert< glm::vec3 > Struct Reference

Static Public Member Functions

- static Node encode (const glm::vec3 &rhs)
- static bool decode (const Node &node, glm::vec3 &rhs)
- static Node encode (const glm::vec3 &rhs)
- static bool decode (const Node &node, glm::vec3 &rhs)

9.10.1 Member Function Documentation

9.10.1.1 decode() [1/2]

```
bool YAML::convert< glm::vec3 >::decode (
            const Node & node,
            glm::vec3 & rhs)  [inline], [static]
```

```
00051         {
00052             if (!node.IsSequence() || node.size() != 3)
00053                 return false;
00054
00055             rhs.x = node[0].as<float>();
00056             rhs.y = node[1].as<float>();
00057             rhs.z = node[2].as<float>();
00058             return true;
00059         }
```

9.10.1.2 decode() [2/2]

```
bool YAML::convert< glm::vec3 >::decode (
            const Node & node,
            glm::vec3 & rhs)  [inline], [static]
```

```
00051         {
00052             if (!node.IsSequence() || node.size() != 3)
00053                 return false;
00054
00055             rhs.x = node[0].as<float>();
00056             rhs.y = node[1].as<float>();
00057             rhs.z = node[2].as<float>();
00058             return true;
00059         }
```

9.10.1.3 encode() [1/2]

```
Node YAML::convert< glm::vec3 >::encode (
            const glm::vec3 & rhs)  [inline], [static]
```

```
00041         {
00042             Node node;
00043             node.push_back(rhs.x);
00044             node.push_back(rhs.y);
00045             node.push_back(rhs.z);
00046             node.SetStyle(EmitterStyle::Flow);
00047             return node;
00048         }
```

9.10.1.4 encode() [2/2]

```
Node YAML::convert< glm::vec3 >::encode (
            const glm::vec3 & rhs)  [inline], [static]
00041         {
00042             Node node;
00043             node.push_back(rhs.x);
00044             node.push_back(rhs.y);
00045             node.push_back(rhs.z);
00046             node.SetStyle(EmitterStyle::Flow);
00047             return node;
00048         }
```

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

- Vesper/src/Vesper/Scene/SceneSerializer.cpp

9.11 YAML::convert< glm::vec4 > Struct Reference

Static Public Member Functions

- static Node encode (const glm::vec4 &rhs)
- static bool decode (const Node &node, glm::vec4 &rhs)
- static Node encode (const glm::vec4 &rhs)
- static bool decode (const Node &node, glm::vec4 &rhs)

9.11.1 Member Function Documentation

9.11.1.1 decode() [1/2]

```
bool YAML::convert< glm::vec4 >::decode (
            const Node & node,
            glm::vec4 & rhs)  [inline], [static]
00077         {
00078             if (!node.IsSequence() || node.size() != 4)
00079                 return false;
00080
00081             rhs.x = node[0].as<float>();
00082             rhs.y = node[1].as<float>();
00083             rhs.z = node[2].as<float>();
00084             rhs.w = node[3].as<float>();
00085             return true;
00086         }
```

9.11.1.2 decode() [2/2]

```
bool YAML::convert< glm::vec4 >::decode (
            const Node & node,
            glm::vec4 & rhs)  [inline], [static]
00077         {
00078             if (!node.IsSequence() || node.size() != 4)
00079                 return false;
00080
00081             rhs.x = node[0].as<float>();
00082             rhs.y = node[1].as<float>();
00083             rhs.z = node[2].as<float>();
00084             rhs.w = node[3].as<float>();
00085             return true;
00086         }
```

9.11.1.3 encode() [1/2]

```
Node YAML::convert< glm::vec4 >::encode (
             const glm::vec4 & rhs)  [inline], [static]
```

```
00066         {
00067             Node node;
00068             node.push_back(rhs.x);
00069             node.push_back(rhs.y);
00070             node.push_back(rhs.z);
00071             node.push_back(rhs.w);
00072             node.SetStyle(EmitterStyle::Flow);
00073             return node;
00074         }
```

9.11.1.4 encode() [2/2]

```
Node YAML::convert< glm::vec4 >::encode (
             const glm::vec4 & rhs)  [inline], [static]
```

```
00066         {
00067             Node node;
00068             node.push_back(rhs.x);
00069             node.push_back(rhs.y);
00070             node.push_back(rhs.z);
00071             node.push_back(rhs.w);
00072             node.SetStyle(EmitterStyle::Flow);
00073             return node;
00074         }
```

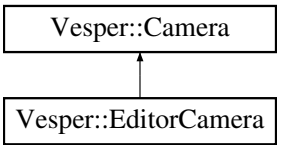
The documentation for this struct was generated from the following file:

- Vesper/src/Vesper/Scene/SceneSerializer.cpp

9.12 Vesper::EditorCamera Class Reference

```
#include <EditorCamera.h>
```

Inheritance diagram for Vesper::EditorCamera:



Public Member Functions

- EditorCamera ()
- EditorCamera (float fov, float aspectRation, float nearClip, float farClip)
- void OnUpdate (Timestep ts)
- void OnEvent (Event &e)
- float GetDistance () const
- void SetDistance (float distance)
- void SetViewportSize (float width, float height)
- const glm::mat4 & GetViewMatrix () const
- const glm::mat4 & GetViewProjection () const
- glm::vec3 GetUpDirection () const

- glm::vec3 GetRightDirection () const
- glm::vec3 GetForwardDirection () const
- glm::quat GetOrientation () const
- const glm::vec3 & GetPosition () const
- void SetPosition (const glm::vec3 &position)
- float GetPitch () const
- void SetPitch (float pitch)
- float GetYaw () const
- void SetYaw (float yaw)

Public Member Functions inherited from Vesper::Camera

- Camera ()=default
- Camera (const glm::mat4 &projection)
- ~Camera ()=default
- const glm::mat4 & GetProjection () const

Private Member Functions

- void UpdateProjection ()
- void UpdateView ()
- bool OnMouseScroll (MouseScrolledEvent &e)
- void MousePan (const glm::vec2 &delta)
- void MouseRotate (const glm::vec2 &delta)
- void MouseZoom (float delta)
- glm::vec3 CalculatePosition () const
- std::pair< float, float > PanSpeed () const
- float RotationSpeed () const
- float ZoomSpeed () const

Private Attributes

- float m_FOV = 45.0f
- float m_AspectRatio = 1.778f
- float m_NearClip = 0.1f
- float m_FarClip = 1000.0f
- glm::mat4 m_ViewMatrix
- glm::vec3 m_Position = { 0.0f, 0.0f, 0.0f }
- glm::vec3 m_FocalPoint = glm::vec3(1.0f)
- glm::vec2 m_InitialMousePosition = { 0.0f, 0.0f }
- float m_Distance = 10.0f
- float m_Pitch = 0.0f
- float m_Yaw = 0.0f
- float m_ViewportWidth = 1280
- float m_ViewportHeight = 720

Additional Inherited Members

Protected Attributes inherited from Vesper::Camera

- glm::mat4 m_Projection = glm::mat4(1.0f)

9.12.1 Constructor & Destructor Documentation

9.12.1.1 EditorCamera() [1/2]

```
Vesper::EditorCamera::EditorCamera ()
00013     {
00014     }
```

9.12.1.2 EditorCamera() [2/2]

```
Vesper::EditorCamera::EditorCamera (
             float fov,
             float aspectRation,
             float nearClip,
             float farClip)
00016         : m_FOV(fov), m_AspectRatio(aspectRatio), m_NearClip(nearClip), m_FarClip(farClip),
      Camera(glm::perspective(glm::radians(fov), aspectRatio, nearClip, farClip))
00017     {
00018         UpdateView();
00019     }
```

References EditorCamera(), m_AspectRatio, m_FarClip, m_FOV, m_NearClip, and UpdateView().

Referenced by EditorCamera().

9.12.2 Member Function Documentation

9.12.2.1 CalculatePosition()

```
glm::vec3 Vesper::EditorCamera::CalculatePosition () const  [private]
00120     {
00121         return m_FocalPoint - GetForwardDirection() * m_Distance;
00122     }
```

9.12.2.2 GetDistance()

```
float Vesper::EditorCamera::GetDistance () const  [inline]
00020 { return m_Distance; }
```

References m_Distance.

9.12.2.3 GetForwardDirection()

```
glm::vec3 Vesper::EditorCamera::GetForwardDirection () const
00133                                                     {
00134         return glm::rotate(GetOrientation(), glm::vec3(0.0f, 0.0f, -1.0f));
00135     }
```

9.12.2.4 GetOrientation()

```
glm::quat Vesper::EditorCamera::GetOrientation () const
00137                                                 {
00138         return glm::quat(glm::vec3(-m_Pitch, -m_Yaw, 0.0f));
00139     }
```

9.12.2.5 GetPitch()

```
float Vesper::EditorCamera::GetPitch () const  [inline]
00036 { return m_Pitch; }
```

References m_Pitch.

9.12.2.6 GetPosition()

```
const glm::vec3 & Vesper::EditorCamera::GetPosition () const  [inline]
00033 { return m_Position; }
```

9.12.2.7 GetRightDirection()

```
glm::vec3 Vesper::EditorCamera::GetRightDirection () const
00129                                                 {
00130         return glm::rotate(GetOrientation(), glm::vec3(1.0f, 0.0f, 0.0f));
00131     }
```

9.12.2.8 GetUpDirection()

```
glm::vec3 Vesper::EditorCamera::GetUpDirection () const
00124                                               {
00125
00126         return glm::rotate(GetOrientation(), glm::vec3(0.0f, 1.0f, 0.0f));
00127     }
```

9.12.2.9 GetViewMatrix()

```
const glm::mat4 & Vesper::EditorCamera::GetViewMatrix () const  [inline]
00025 { return m_ViewMatrix; }
```

9.12.2.10 GetViewProjection()

```
const glm::mat4 & Vesper::EditorCamera::GetViewProjection () const  [inline]
00026 { return m_Projection * m_ViewMatrix; }
```

9.12.2.11 GetYaw()

```
float Vesper::EditorCamera::GetYaw () const  [inline]
00039 { return m_Yaw; }
```

References m_Yaw.

9.12.2.12 MousePan()

```
void Vesper::EditorCamera::MousePan (
            const glm::vec2 & delta)  [private]
```

```
00096     {
00097         auto [xSpeed, ySpeed] = PanSpeed();
00098         m_FocalPoint += -GetRightDirection() * delta.x * xSpeed * m_Distance;
00099         m_FocalPoint += GetUpDirection() * delta.y * ySpeed * m_Distance;
00100     }
```

9.12.2.13 MouseRotate()

```
void Vesper::EditorCamera::MouseRotate (
            const glm::vec2 & delta)  [private]
```

```
00103     {
00104         float yawSign = GetUpDirection().y < 0 ? -1.0f : 1.0f;
00105         m_Yaw += yawSign * delta.x * RotationSpeed();
00106         m_Pitch += delta.y * RotationSpeed();
00107     }
```

References m_Pitch, m_Yaw, and RotationSpeed().

9.12.2.14 MouseZoom()

```
void Vesper::EditorCamera::MouseZoom (
            float delta)  [private]
```

```
00110     {
00111         m_Distance -= delta * ZoomSpeed();
00112         if (m_Distance < 1.0f)
00113         {
00114             m_FocalPoint += GetForwardDirection();
00115             m_Distance = 1.0f;
00116         }
00117     }
```

References m_Distance, and ZoomSpeed().

Referenced by OnMouseScroll(), and OnUpdate().

9.12.2.15 OnEvent()

```
void Vesper::EditorCamera::OnEvent (
            Event & e)
```

```
00066     {
00067         EventDispatcher dispatcher(e);
00068         dispatcher.Dispatch<MouseScrolledEvent>(VZ_BIND_EVENT_FN(EditorCamera::OnMouseScroll));
00069     }
```

References Vesper::EventDispatcher::EventDispatcher(), and OnMouseScroll().

9.12.2.16 OnMouseScroll()

```
bool Vesper::EditorCamera::OnMouseScroll (
            MouseScrolledEvent & e)  [private]
```

```
00088     {
00089         float delta = e.GetYOffset() * 0.1f;
00090         MouseZoom(delta);
00091         UpdateView();
00092         return false;
00093     }
```

References Vesper::MouseScrolledEvent::GetYOffset(), MouseZoom(), and UpdateView().

Referenced by OnEvent().

9.12.2.17 OnUpdate()

```
void Vesper::EditorCamera::OnUpdate (
            Timestep ts)
```

```
00047     {
00048         if (Input::IsKeyPressed(VZ_KEY_LEFT_ALT))
00049         {
00050             const glm::vec2& mouse{ Input::GetMouseX(), Input::GetMouseY() };
00051             glm::vec2 delta = (mouse - m_InitialMousePosition) * 0.003f;
00052             m_InitialMousePosition = mouse;
00053 
00054             if (Input::IsMouseButtonPressed(VZ_MOUSE_BUTTON_MIDDLE))
00055                 MousePan(delta);
00056             else if (Input::IsMouseButtonPressed(VZ_MOUSE_BUTTON_LEFT))
00057                 MouseRotate(delta);
00058             else if (Input::IsMouseButtonPressed(VZ_MOUSE_BUTTON_RIGHT))
00059                 MouseZoom(-delta.y);
00060         }
00061 
00062         UpdateView();
00063     }
```

References Vesper::Input::IsKeyPressed(), Vesper::Input::IsMouseButtonPressed(), MouseZoom(), and UpdateView().

9.12.2.18 PanSpeed()

```
std::pair< float, float > Vesper::EditorCamera::PanSpeed () const  [private]
```

```
00022     {
00023         float x = std::min(m_ViewportWidth / 1000.0f, 2.4f); // max = 2.4f
00024         float xFactor = 0.0366f * (x * x) - 0.1778f * x + 0.3021f;
00025 
00026         float y = std::min(m_ViewportHeight / 1000.0f, 2.4f); // max = 2.4f
00027         float yFactor = 0.0366f * (y * y) - 0.1778f * y + 0.3021f;
00028 
00029         return { xFactor, yFactor };
00030     }
```

9.12.2.19 RotationSpeed()

```
float Vesper::EditorCamera::RotationSpeed () const  [private]
```

```
00033     {
00034         return 0.8f;
00035     }
```

Referenced by MouseRotate().

9.12.2.20 SetDistance()

```
void Vesper::EditorCamera::SetDistance (
            float distance)  [inline]
```

```
00021 { m_Distance = distance; }
```

References m_Distance.

9.12.2.21 SetPitch()

```
void Vesper::EditorCamera::SetPitch (
            float pitch)  [inline]
```

```
00037 { m_Pitch = pitch; }
```

References m_Pitch.

9.12.2.22 SetPosition()

```
void Vesper::EditorCamera::SetPosition (
            const glm::vec3 & position)
```

9.12.2.23 SetViewportSize()

```
void Vesper::EditorCamera::SetViewportSize (
            float width,
            float height)  [inline]
00023 { m_ViewportWidth = width, m_ViewportHeight = height; UpdateProjection(); }
```

References m_ViewportHeight, m_ViewportWidth, and UpdateProjection().

9.12.2.24 SetYaw()

```
void Vesper::EditorCamera::SetYaw (
            float yaw)  [inline]
00040 { m_Yaw = yaw; }
```

References m_Yaw.

9.12.2.25 UpdateProjection()

```
void Vesper::EditorCamera::UpdateProjection ()  [private]
00072     {
00073         m_AspectRatio = m_ViewportWidth / m_ViewportHeight;
00074         m_Projection = glm::perspective(glm::radians(m_FOV), m_AspectRatio, m_NearClip, m_FarClip);
00075     }
```

References m_AspectRatio, m_ViewportHeight, and m_ViewportWidth.

Referenced by SetViewportSize().

9.12.2.26 UpdateView()

```
void Vesper::EditorCamera::UpdateView ()  [private]
00078     {
00079         //m_Yaw = m_Pitch = 0.0f; // Lock the camera's rotation
00080         m_Position = CalculatePosition();
00081
00082         glm::quat orientation = GetOrientation();
00083         m_ViewMatrix = glm::translate(glm::mat4(1.0f), m_Position) * glm::toMat4(orientation);
00084         m_ViewMatrix = glm::inverse(m_ViewMatrix);
00085     }
```

Referenced by EditorCamera(), OnMouseScroll(), and OnUpdate().

9.12.2.27 ZoomSpeed()

```
float Vesper::EditorCamera::ZoomSpeed () const  [private]
```

```
00038     {
00039         float distance = m_Distance * 0.2f;
00040         distance = std::max(distance, 0.0f);
00041         float speed = distance * distance;
00042         speed = std::min(speed, 100.0f); // max speed = 100
00043         return speed;
00044     }
```

References m_Distance.

Referenced by MouseZoom().

9.12.3 Member Data Documentation

9.12.3.1 m_AspectRatio

```
float Vesper::EditorCamera::m_AspectRatio = 1.778f  [private]
```

Referenced by EditorCamera(), and UpdateProjection().

9.12.3.2 m_Distance

```
float Vesper::EditorCamera::m_Distance = 10.0f  [private]
```

Referenced by GetDistance(), MouseZoom(), SetDistance(), and ZoomSpeed().

9.12.3.3 m_FarClip

```
float Vesper::EditorCamera::m_FarClip = 1000.0f  [private]
```

Referenced by EditorCamera().

9.12.3.4 m_FocalPoint

```
glm::vec3 Vesper::EditorCamera::m_FocalPoint = glm::vec3(1.0f)  [private]
```

9.12.3.5 m_FOV

```
float Vesper::EditorCamera::m_FOV = 45.0f  [private]
```

Referenced by EditorCamera().

9.12.3.6 m_InitialMousePosition

```
glm::vec2 Vesper::EditorCamera::m_InitialMousePosition = { 0.0f, 0.0f }  [private]
```

```
00065 { 0.0f, 0.0f };
```

9.12.3.7 m_NearClip

```
float Vesper::EditorCamera::m_NearClip = 0.1f  [private]
```

Referenced by EditorCamera().

9.12.3.8 m_Pitch

```
float Vesper::EditorCamera::m_Pitch = 0.0f  [private]
```

Referenced by GetPitch(), MouseRotate(), and SetPitch().

9.12.3.9 m_Position

```
glm::vec3 Vesper::EditorCamera::m_Position = { 0.0f, 0.0f, 0.0f }  [private]
00062 { 0.0f, 0.0f, 0.0f };
```

9.12.3.10 m_ViewMatrix

```
glm::mat4 Vesper::EditorCamera::m_ViewMatrix  [private]
```

9.12.3.11 m_ViewportHeight

```
float Vesper::EditorCamera::m_ViewportHeight = 720  [private]
```

Referenced by SetViewportSize(), and UpdateProjection().

9.12.3.12 m_ViewportWidth

```
float Vesper::EditorCamera::m_ViewportWidth = 1280  [private]
```

Referenced by SetViewportSize(), and UpdateProjection().

9.12.3.13 m_Yaw

```
float Vesper::EditorCamera::m_Yaw = 0.0f  [private]
```

Referenced by GetYaw(), MouseRotate(), and SetYaw().

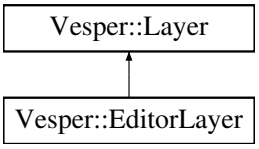
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/Renderer/EditorCamera.h
- Vesper/src/Vesper/Renderer/EditorCamera.cpp

9.13 Vesper::EditorLayer Class Reference

`#include <EditorLayer.h>`

Inheritance diagram for Vesper::EditorLayer:



Public Member Functions

- EditorLayer ()
- virtual ~EditorLayer ()=default
- virtual void OnAttach () override
- virtual void OnDetach () override
- virtual void OnUpdate (Timestep ts) override
- virtual void OnImGuiRender () override
- virtual void OnEvent (Event &e) override

Public Member Functions inherited from Vesper::Layer

- Layer (const std::string &name="Layer")
- virtual ~Layer ()
- virtual void OnRender ()
- const std::string & GetName () const

Private Types

- enum class SceneState { Edit = 0 , Play = 1 , Simulate = 2 }

Private Member Functions

- bool OnKeyPressed (KeyPressedEvent &e)
- void NewScene ()
- void OpenScene ()
- void SaveSceneAs ()
- void ResetScene ()

Private Attributes

- SceneHierarchyPanel m_SceneHierarchyPanel
- Ref< Scene > m_ActiveScene
- Ref< Scene > m_EditorScene
- SceneState m_SceneState = SceneState::Edit
- bool m_ViewportFocused = false
- bool m_ViewportHovered = false
- glm::vec2 m_ViewportSize = {0,0}
- glm::vec2 m_ViewportBounds [2] = { {0,0}, {0,0} }
- bool m_PrimaryCamera = true
- Entity m_CameraEntity
- int m_GizmoType = -1
- float m_TranslationSnap = 0.5f
- float m_RotationSnap = 45.0f
- float m_ScaleSnap = 0.5f
- OrthographicCameraController m_CameraController
- float lastFrameTime = 0.0f
- Entity m_FireEntity
- Entity m_SmokeEntity
- Ref< VertexArray > m_SquareVA
- Ref< Shader > m_FlatColorShader
- Ref< Texture2D > m_CheckerboardTexture
- Ref< Texture2D > m_SpriteSheetFire
- Ref< Texture2D > m_SpriteSheetSmoke
- Ref< Texture2D > m_SpriteSheetTown
- Ref< Texture2D > m_SpriteSheetCrystals
- Ref< Texture2D > m_SpriteSheetRocks
- Ref< Texture2D > m_SpriteSheetCursedLands
- Ref< SubTexture2D > m_SubTextureFire
- Ref< SubTexture2D > m_SubTextureSmoke
- Ref< SubTexture2D > m_SubTextureTown
- Ref< Framebuffer > m_Framebuffer
- EditorCamera m_EditorCamera
- float m_textureScale = 1.0f
- float m_squareRotation = 25.0f
- float m_specialQuadRotation = 0.5f
- int ParticleEmitCount = 100
- ParticleSystem m_ParticleSystem
- ParticleProps m_ParticleProps
- bool scene1 = false
- bool scene2 = false
- bool scene3 = false
- bool scene4 = false
- bool useEntityScene = true
- glm::vec4 m_SquareColor = { 0.2f, 0.3f, 0.8f, 1.0f }
- glm::vec4 m_TextureTintColor1 = { 1.0f, 1.0f, 1.0f, 1.0f }
- glm::vec4 m_TextureTintColor2 = { 1.0f, 1.0f, 1.0f, 1.0f }
- glm::vec4 m_BackgroundColor = { 0.1f, 0.1f, 0.1f, 1.0f }
- glm::vec4 m_ClearColor = { 0.1f, 0.3f, 0.3f, 1.0f }
- glm::vec4 m_SpecialQuadColor = { 0.9f, 0.2f, 0.8f, 1.0f }
- bool m_UseSpecialQuadColor = false
- std::unordered_map< char, Ref< SubTexture2D > > s_TextureMap

Additional Inherited Members

Protected Attributes inherited from Vesper::Layer

- std::string m_DebugName

9.13.1 Member Enumeration Documentation

9.13.1.1 SceneState

```
enum class Vesper::EditorLayer::SceneState  [strong], [private]
```

Enumerator

Edit	
Play	
Simulate	

```
00039         {
00040             Edit = 0, Play = 1, Simulate = 2
00041         };
```

9.13.2 Constructor & Destructor Documentation

9.13.2.1 EditorLayer()

```
Vesper::EditorLayer::EditorLayer ()
00038         : Layer("Sandbox2D"), m_CameraController(1280.0f / 720.0f, true)
00039     {
00040         VZ_PROFILE_FUNCTION();
00041     }
```

References EditorLayer().

Referenced by EditorLayer(), and Vesper::VesperEditor::VesperEditor().

9.13.2.2 ~EditorLayer()

```
virtual Vesper::EditorLayer::~EditorLayer ()  [virtual], [default]
```

9.13.3 Member Function Documentation

9.13.3.1 NewScene()

```
void Vesper::EditorLayer::NewScene ()  [private]
00722     {
00723         m_ActiveScene = CreateRef<Scene>();
00724         m_ActiveScene->OnViewportResize((uint32_t)m_ViewportSize.x, (uint32_t)m_ViewportSize.y);
00725         m_SceneHierarchyPanel.SetContext(m_ActiveScene);
00726     }
```

Referenced by OnImGuiRender(), and OnKeyPressed().

9.13.3.2 OnAttach()

```
void Vesper::EditorLayer::OnAttach ()  [override], [virtual]
```

TODO: move to resource manager TODO: fix pathing

TODO: Get an automatic path to resource that is NOT hardcoded

Reimplemented from Vesper::Layer.

```
00046     {
00047         VZ_PROFILE_FUNCTION();
00048 
00049         // Texture / SubTexture setup
00050         {
00053             m_CheckerboardTexture =
      Texture2D::Create("../../Vesper-Editor/assets/textures/Checkerboard.png");
00054             m_SpriteSheetFire =
      Texture2D::Create("../../Vesper-Editor/assets/textures/sheets/fire_01.png");
00055             m_SpriteSheetSmoke =
      Texture2D::Create("../../Vesper-Editor/assets/textures/sheets/fire_02.png");
00056             m_SpriteSheetTown =
      Texture2D::Create("../../Vesper-Editor/assets/textures/sheets/town_tilesheet.png");
00057             m_SpriteSheetCrystals =
      Texture2D::Create("../../Vesper-Editor/assets/textures/sheets/craftpix/Crystals/Crystals.png");
00058             m_SpriteSheetRocks =
      Texture2D::Create("../../Vesper-Editor/assets/textures/sheets/craftpix/Rocks/Rocks_source.png");
00059             m_SpriteSheetCursedLands =
      Texture2D::Create("../../Vesper-Editor/assets/textures/sheets/craftpix/CursedLand/Tiled_files/Objects.png");
00060 
00061             m_SubTextureFire = SubTexture2D::CreateFromCoords(m_SpriteSheetFire, { 1, 0 }, { 128, 127
      });
00062             m_SubTextureSmoke = SubTexture2D::CreateFromCoords(m_SpriteSheetSmoke, { 1, 0 }, { 128,
      127 });
00063             m_SubTextureTown = SubTexture2D::CreateFromCoords(m_SpriteSheetTown, { 4.25, 0.75 }, { 64,
      64 }, { 1, 1 });
00064             s_TextureMap['F'] = SubTexture2D::CreateFromCoords(m_SpriteSheetFire, { 1, 0 }, { 128, 127
      });
00065             s_TextureMap['G'] = SubTexture2D::CreateFromCoords(m_SpriteSheetTown, { 4.25, 0.75 }, {
      64, 64 }, { 1, 1 });
00066             s_TextureMap['C'] = SubTexture2D::CreateFromCoords(m_SpriteSheetCrystals, { 0, 1.25 }, {
      64, 64 }, { 1, 1 });
00067             s_TextureMap['R'] = SubTexture2D::CreateFromCoords(m_SpriteSheetRocks, { 0, 3.75 }, { 64,
      64 }, { 1, 1 });
00068             s_TextureMap['P'] = SubTexture2D::CreateFromCoords(m_SpriteSheetCursedLands, { 0, 1.875 },
      { 128, 128 }, { 1, 1 });
00069         }
00070 
00071 
00072         // Particle setup
00073         {
00074             m_ParticleProps.Position = { 0.0f, 0.0f, 0.0f };
00075             m_ParticleProps.Velocity = { 0.0f, 0.0f, 0.0f };
00076             m_ParticleProps.VelocityVariation = { 1.0f, 1.0f, 0.0f };
00077             m_ParticleProps.ColorBegin = { 1.0f, 0.5f, 0.2f, 1.0f };
00078             m_ParticleProps.ColorEnd = { 0.2f, 0.3f, 0.8f, 1.0f };
00079             m_ParticleProps.SizeBegin = 0.5f;
00080             m_ParticleProps.SizeEnd = 0.0f;
00081             m_ParticleProps.LifeTime = 3.0f;
00082             m_ParticleProps.Rotation = 0.0f;
00083             m_ParticleProps.RotationVariation = 27.0f;
00084             m_ParticleSystem = ParticleSystem(10000);
00085             m_ParticleSystem.SetParticleProps(m_ParticleProps);
00086         }
00087 
00088         // Framebuffer/Viewport setup
00089         {
00090             FramebufferSpecification fbSpec;
00091             fbSpec.Width = 1280;
00092             fbSpec.Height = 720;
00093             m_Framebuffer = Framebuffer::Create(fbSpec);
00094         }
00095 
00096         // Scene setup
00097         {
00098             //m_CameraController.SetZoomLevel(5.5f);
00099             m_ActiveScene = CreateRef<Scene>();
00100 
00101 #if 0
00102             m_CameraEntity = m_ActiveScene->CreateEntity("Primary Camera Entity");
00103 
00104             auto& pCam = m_CameraEntity.AddComponent<CameraComponent>();
00105             pCam.Primary = true;
```

```
00106             pCam.Camera.SetPerspective(glm::radians(45.0f), 0.1f, 1000.0f);
00107             auto& pos = m_CameraEntity.GetComponent<TransformComponent>().Translation;
00108             pos.x += 1.25f;
00109             pos.z += 5.0f;
00110 
00111             auto fbSpec = m_Framebuffer->GetSpecification();
00112             m_ActiveScene->OnViewportResize(fbSpec.Width, fbSpec.Height);
00113 
00114             // Animation 1
00115             {
00116                 auto square = m_ActiveScene->CreateEntity("Fire Animation");
00117                 auto& transform = square.GetComponent<TransformComponent>();
00118                 transform.Translation = (glm::vec3(-0.5f, 0.0f, -1.5f));
00119 
00120                 square.AddComponent<SpriteRendererComponent>(glm::vec4{ 0.8f, 0.8f, 0.2f, 1.0f });
00121                 std::vector<Ref<SubTexture2D» fireFrames;
00122                 for (int x = 0; x < 63; x++)
00123                 {
00124                     for (int y = 0; y < 2; y++)
00125                     {
00126                         fireFrames.push_back(SubTexture2D::CreateFromCoords(m_SpriteSheetFire, {
      (float)y, (float)x }, { 128, 128 }));
00127                     }
00128                 }
00129                 TextureAnimationComponent texAnim(fireFrames, 0.05f);
00130                 square.AddComponent<TextureAnimationComponent>(texAnim);
00131                 m_FireEntity = square;
00132             }
00133 
00134             // Animation 2
00135             {
00136                 auto square = m_ActiveScene->CreateEntity("Smoke Animation");
00137                 auto& transform = square.GetComponent<TransformComponent>();
00138                 // adjust the position of the square entity
00139                 transform.Translation = (glm::vec3(0.5f, 0.0f, 1.5f));
00140 
00141                 square.AddComponent<SpriteRendererComponent>(glm::vec4{ 0.8f, 0.8f, 0.2f, 1.0f });
00142                 std::vector<Ref<SubTexture2D» smokeFrames;
00143                 for (int x = 0; x < 63; x++)
00144                 {
00145                     for (int y = 0; y < 2; y++)
00146                     {
00147                         smokeFrames.push_back(SubTexture2D::CreateFromCoords(m_SpriteSheetSmoke, {
      (float)y, (float)x }, { 128, 128 }));
00148                     }
00149 
00150                 }
00151                 TextureAnimationComponent texAnim(smokeFrames, 0.05f);
00152                 square.AddComponent<TextureAnimationComponent>(texAnim);
00153                 m_SmokeEntity = square;
00154             }
00155 
00156             auto quadEntity = m_ActiveScene->CreateEntity("Quad Entity");
00157             quadEntity.AddComponent<SpriteRendererComponent>(glm::vec4{ 0.2f, 0.3f, 0.8f, 1.0f });
00158             quadEntity.GetComponent<TransformComponent>().Scale = { 0.5f, 0.5f, 1.0f };
00159             quadEntity.GetComponent<TransformComponent>().Translation = { 1.5f, 0.0f, 0.0f };
00160 
00161 
00162             class CameraController : public ScriptableEntity {
00163             public:
00164                 void OnCreate()
00165                 {
00166                     GetComponent<TransformComponent>().Translation = (glm::vec3(Random::RangeF1(-3.0f,
      3.0f), Random::RangeF1(-3.0f, 3.0f), 0.0f));
00167 
00168                 }
00169 
00170                 void OnDestroy()
00171                 {
00172 
00173                 }
00174 
00175                 void OnUpdate(Timestep ts)
00176                 {
00177                     auto& transform = GetComponent<TransformComponent>().GetTransform();
00178                     float speed = 5.0f;
00179 
00180 
00181                     if (Input::IsKeyPressed(VZ_KEY_A))
00182                         transform[3][0] -= speed * ts;
00183 
00184                     if (Input::IsKeyPressed(VZ_KEY_D))
00185                         transform[3][0] += speed * ts;
00186 
00187                     if (Input::IsKeyPressed(VZ_KEY_W))
00188                         transform[3][1] += speed * ts;
00189 
```

```
00190                     if (Input::IsKeyPressed(VZ_KEY_S))
00191                         transform[3][1] -= speed * ts;
00192                 }
00193 
00194             };
00195 
00196             m_CameraEntity.AddComponent<NativeScriptComponent>().Bind<CameraController>();
00197             m_SecondaryCameraEntity.AddComponent<NativeScriptComponent>().Bind<CameraController>();
00198 
00199 #endif
00200 
00201             m_SceneHierarchyPanel.SetContext(m_ActiveScene);
00202 
00203             SceneSerializer serializer(m_ActiveScene);
00204 
00205             FileSystem::Initialize();
00206 
00207             if (VZ_EDITOR_USE_DEFAULT_SCENE) {
00209                 std::string loadedScene = FileSystem::GetAbsolutePath("../../" +
      std::string(VZ_EDITOR_DEFAULT_SCENE));
00210 
00211                 bool valid = serializer.Deserialize(loadedScene);
00212                 if (!valid) {
00213                     VZ_CORE_ERROR("Failed to load default scene: " + loadedScene);
00214                     VZ_CORE_ERROR("Attempted Scene: " + std::string(VZ_EDITOR_DEFAULT_SCENE));
00215                     VZ_CORE_ERROR("Current Working Directory: " +
      FileSystem::GetCurrentWorkingDirectory());
00216                     VZ_CORE_ERROR("Absolute Path Attempted: " +
      FileSystem::GetAbsolutePath(loadedScene));
00217                     VZ_CORE_ERROR("Error loading the scene, please check the paths and file
      availability.");
00218                 }
00219                 else {
00220                     VZ_CORE_INFO("Successfully loaded the default scene: " + loadedScene);
00221                 }
00222             }
00223             m_EditorCamera = EditorCamera(30.0f, 1.778f, 0.1f, 1000.0f);
00224         }
00225 
00226 
00227     }
```

References Vesper::SceneSerializer::Deserialize(), Vesper::FramebufferSpecification::Height, Vesper::FileSystem::Initialize(), and Vesper::FramebufferSpecification::Width.

9.13.3.3 OnDetach()

```
void Vesper::EditorLayer::OnDetach ()  [override], [virtual]
```

Reimplemented from Vesper::Layer.

```
00230     {
00231         VZ_PROFILE_FUNCTION();
00232     }
```

9.13.3.4 OnEvent()

```
void Vesper::EditorLayer::OnEvent (
            Event & e)  [override], [virtual]
```

Reimplemented from Vesper::Layer.

```
00658     {
00659         m_CameraController.OnEvent(e);
00660         if (m_SceneState == SceneState::Edit) {
00661             m_EditorCamera.OnEvent(e);
00662         }
00663 
00664         EventDispatcher dispatcher(e);
00665         dispatcher.Dispatch<KeyPressedEvent>(VZ_BIND_EVENT_FN(EditorLayer::OnKeyPressed));
00666 
00667     }
```

References Edit, Vesper::EventDispatcher::EventDispatcher(), m_SceneState, and OnKeyPressed().

9.13.3.5 OnImGuiRender()

```
void Vesper::EditorLayer::OnImGuiRender ()  [override], [virtual]
```

Reimplemented from Vesper::Layer.

```
00426     {
00427         VZ_PROFILE_FUNCTION();
00428 
00429         static bool dockspaceOpen = true;
00430         static bool opt_fullscreen = true;
00431         static bool opt_padding = false;
00432         static ImGuiDockNodeFlags dockspace_flags = ImGuiDockNodeFlags_None;
00433 
00434         // We are using the ImGuiWindowFlags_NoDocking flag to make the parent window not dockable
      into,
00435         // because it would be confusing to have two docking targets within each others.
00436         ImGuiWindowFlags window_flags = ImGuiWindowFlags_MenuBar | ImGuiWindowFlags_NoDocking;
00437         if (opt_fullscreen)
00438         {
00439             const ImGuiViewport* viewport = ImGui::GetMainViewport();
00440             ImGui::SetNextWindowPos(viewport->WorkPos);
00441             ImGui::SetNextWindowSize(viewport->WorkSize);
00442             ImGui::SetNextWindowViewport(viewport->ID);
00443             ImGui::PushStyleVar(ImGuiStyleVar_WindowRounding, 0.0f);
00444             ImGui::PushStyleVar(ImGuiStyleVar_WindowBorderSize, 0.0f);
00445             window_flags |= ImGuiWindowFlags_NoTitleBar | ImGuiWindowFlags_NoCollapse |
      ImGuiWindowFlags_NoResize | ImGuiWindowFlags_NoMove;
00446             window_flags |= ImGuiWindowFlags_NoBringToFrontOnFocus | ImGuiWindowFlags_NoNavFocus;
00447         }
00448 
00449         // When using ImGuiDockNodeFlags_PassthruCentralNode, DockSpace() will render our background
00450         // and handle the pass-thru hole, so we ask Begin() to not render a background.
00451         if (dockspace_flags & ImGuiDockNodeFlags_PassthruCentralNode)
00452             window_flags |= ImGuiWindowFlags_NoBackground;
00453 
00454         // Important: note that we proceed even if Begin() returns false (aka window is collapsed).
00455         // This is because we want to keep our DockSpace() active. If a DockSpace() is inactive,
00456         // all active windows docked into it will lose their parent and become undocked.
00457         // We cannot preserve the docking relationship between an active window and an inactive
      docking, otherwise
00458         // any change of dockspace/settings would lead to windows being stuck in limbo and never being
      visible.
00459         if (!opt_padding)
00460             ImGui::PushStyleVar(ImGuiStyleVar_WindowPadding, ImVec2(0.0f, 0.0f));
00461         ImGui::Begin("DockSpace Demo", &dockspaceOpen, window_flags);
00462         if (!opt_padding)
00463             ImGui::PopStyleVar();
00464 
00465         if (opt_fullscreen)
00466             ImGui::PopStyleVar(2);
00467 
00468         // Submit the DockSpace
00469         ImGuiIO& io = ImGui::GetIO();
00470         ImGuiStyle& style = ImGui::GetStyle();
00471         float minWinSize = style.WindowMinSize.x = 370.0f;
00472 
00473         if (io.ConfigFlags & ImGuiConfigFlags_DockingEnable)
00474         {
00475             ImGuiID dockspace_id = ImGui::GetID("MyDockSpace");
00476             ImGui::DockSpace(dockspace_id, ImVec2(0.0f, 0.0f), dockspace_flags);
00477         }
00478 
00479         style.WindowMinSize.x = minWinSize;
00480 
00481         if (ImGui::BeginMenuBar())
00482         {
00483             if (ImGui::BeginMenu("File"))
00484             {
00485                 // Disabling fullscreen would allow the window to be moved to the front of other
      windows,
00486                 // which we can’t undo at the moment without finer window depth/z control.
00487 
00488                 if (ImGui::MenuItem("New", "Ctrl+N"))
00489                     NewScene();
00490 
00491 
00492                 if (ImGui::MenuItem("Open..", "Ctrl+O"))
00493                     OpenScene();
00494 
00495 
00496                 if (ImGui::MenuItem("Save As..", "Ctrl+Shift+S"))
00497                     SaveSceneAs();
00498 
00499 
```

```
00500                 if (ImGui::MenuItem("Reset Scene"))
00501                     ResetScene();
00502 
00503 
00504                 if (ImGui::MenuItem("Exit"))
00505                     Vesper::Application::Get().Close();
00506 
00507                 ImGui::EndMenu();
00508             }
00509 
00510             ImGui::EndMenuBar();
00511         }
00512 
00513         {
00514             if (ImGui::Begin("Scenes"))
00515             {
00516                 if (ImGui::Checkbox("Entity Scene", &useEntityScene)) {
00517                     if (useEntityScene) {
00518                         scene1 = false;
00519                         scene2 = false;
00520                         scene3 = false;
00521                         scene4 = false;
00522                     }
00523                 }
00524                 if (ImGui::Checkbox("Scene 1 - Basic Shapes", &scene1)) {
00525                     if (scene1) {
00526                         scene2 = false;
00527                         scene3 = false;
00528                         scene4 = false;
00529                     }
00530                 }
00531                 if (ImGui::Checkbox("Scene 2 - Sprite Sheets", &scene2)) {
00532                     if (scene2) {
00533                         scene1 = false;
00534                         scene3 = false;
00535                         scene4 = false;
00536                     }
00537                 }
00538                 if (ImGui::Checkbox("Scene 3 - Tile Map", &scene3)) {
00539                     if (scene3) {
00540                         scene1 = false;
00541                         scene2 = false;
00542                         scene4 = false;
00543                     }
00544                 }
00545                 if (ImGui::Checkbox("Scene 4 - Particle System", &scene4)) {
00546                     if (scene4) {
00547                         scene1 = false;
00548                         scene2 = false;
00549                         scene3 = false;
00550                     }
00551                 }
00552             }
00553             ImGui::End();
00554         }
00555 
00556         m_SceneHierarchyPanel.OnImGuiRender();
00557 
00558         {
00559             ImGui::Begin("Settings");
00560             ImGui::Text("Renderer2D Stats:");
00561             auto stats = Renderer2D::GetStats();
00562             ImGui::Text("\tDraw Calls: %d", stats.DrawCalls);
00563             ImGui::Text("\tQuad Count: %d", stats.QuadCount);
00564             ImGui::Text("\tVertex Count: %d", stats.GetTotalVertexCount());
00565             ImGui::Text("\tIndex Count: %d", stats.GetTotalIndexCount());
00566             ImGui::Text("Application Settings:");
00567             ImGui::Text("\tFPS: %.1f", ImGui::GetIO().Framerate);
00568             if (ImGui::ColorEdit4("Background Color", glm::value_ptr(m_ClearColor)))
00569             {
00570                 RenderCommand::SetClearColor(m_ClearColor);
00571             }
00572             ImGui::End();
00573         }
00574 
00575         DisplayVesperInfo_ImGui();
00576 
00577         {
00578             ImGui::PushStyleVar(ImGuiStyleVar_WindowPadding, ImVec2{ 0,0 });
00579             ImGui::Begin("Viewport");
00580             m_ViewportFocused = ImGui::IsWindowFocused();
00581             m_ViewportHovered = ImGui::IsWindowHovered();
00582             Application::Get().GetImGuiLayer()->SetBlockEvents(!m_ViewportFocused &&
    !m_ViewportHovered);
00583 
00584             ImVec2 viewportPanelSize = ImGui::GetContentRegionAvail();
00585             if (m_ViewportSize != *((glm::vec2*)&viewportPanelSize) && viewportPanelSize.x > 0.0f &&
```

```
    viewportPanelSize.y > 0)
00586             {
00587                 m_Framebuffer->Resize((uint32_t)viewportPanelSize.x, (uint32_t)viewportPanelSize.y);
00588                 m_ViewportSize = { viewportPanelSize.x, viewportPanelSize.y };
00589 
00590                 m_CameraController.OnResize(viewportPanelSize.x, viewportPanelSize.y);
00591             }
00592 
00593             ImVec2 viewportBoundsMin = ImGui::GetCursorScreenPos();
00594             ImVec2 viewportBoundsMax = { viewportBoundsMin.x + m_ViewportSize.x, viewportBoundsMin.y +
    m_ViewportSize.y };
00595             m_ViewportBounds[0] = { viewportBoundsMin.x, viewportBoundsMin.y };
00596             m_ViewportBounds[1] = { viewportBoundsMax.x, viewportBoundsMax.y };
00597 
00598             uint32_t textureID = m_Framebuffer->GetColorAttachmentRendererID();
00599             ImGui::Image(textureID, ImVec2(m_ViewportSize.x, m_ViewportSize.y), ImVec2(0, 1),
    ImVec2(1, 0));
00600 
00601             // Gizmos
00602             Entity selectedEntity = m_SceneHierarchyPanel.GetSelectedEntity();
00603             if (selectedEntity && m_GizmoType != -1)
00604             {
00605                 ImGuizmo::SetOrthographic(false);
00606                 ImGuizmo::SetDrawlist();
00607                 float windowWidth = (float)ImGui::GetWindowWidth();
00608                 float windowHeight = (float)ImGui::GetWindowHeight();
00609                 ImGuizmo::SetRect(ImGui::GetWindowPos().x, ImGui::GetWindowPos().y, windowWidth,
    windowHeight);
00610 
00611                 // Camera
00612                 // Runtime camera from entity
00613                 //auto cameraEntity = m_ActiveScene->GetPrimaryCameraEntity();
00614                 //const auto& camera = cameraEntity.GetComponent<CameraComponent>().Camera;
00615                 //const glm::mat4& cameraProjection = camera.GetProjection();
00616                 //glm::mat4 cameraView =
    glm::inverse(cameraEntity.GetComponent<TransformComponent>().GetTransform());
00617 
00618                 // Editor camera
00619                 const glm::mat4& cameraProjection = m_EditorCamera.GetProjection();
00620                 glm::mat4 cameraView = m_EditorCamera.GetViewMatrix();
00621 
00622                 // Entity Transform
00623                 auto& tc = selectedEntity.GetComponent<TransformComponent>();
00624                 glm::mat4 transform = tc.GetTransform();
00625 
00626                 // Snapping
00627                 bool snap = Input::IsKeyPressed(VZ_KEY_LEFT_CONTROL);
00628                 // use the editor layer snap values
00629                 float snapValue = m_TranslationSnap;
00630                 if (m_GizmoType == ImGuizmo::OPERATION::ROTATE)
00631                     snapValue = m_RotationSnap;
00632                 else if (m_GizmoType == ImGuizmo::OPERATION::SCALE)
00633                     snapValue = m_ScaleSnap;
00634 
00635                 float snapValues[3] = { snapValue, snapValue, snapValue };
00636 
00637                 ImGuizmo::Manipulate(glm::value_ptr(cameraView), glm::value_ptr(cameraProjection),
00638                     (ImGuizmo::OPERATION)m_GizmoType, ImGuizmo::LOCAL, glm::value_ptr(transform),
00639                     nullptr, snap ? snapValues : nullptr);
00640 
00641                 if (ImGuizmo::IsUsing())
00642                 {
00643                     glm::vec3 translation, rotation, scale;
00644                     Vesper::Math::DecomposeTransform(transform, translation, rotation, scale);
00645                     tc.Translation = translation;
00646                     tc.Rotation = rotation;
00647                     tc.Scale = scale;
00648                 }
00649             }
00650 
00651             ImGui::End();
00652             ImGui::PopStyleVar();
00653         }
00654         ImGui::End();
00655     }
```

References Vesper::Application::Close(), Vesper::DisplayVesperInfo_ImGui(), Vesper::Application::Get(), Vesper::Application::GetImG
Vesper::Renderer2D::GetStats(), Vesper::Input::IsKeyPressed(), m_GizmoType, m_RotationSnap, m_ScaleSnap,
m_TranslationSnap, m_ViewportFocused, m_ViewportHovered, NewScene(), OpenScene(), ResetScene(),
SaveSceneAs(), scene1, scene2, scene3, scene4, Vesper::ImGuiLayer::SetBlockEvents(), and useEntityScene.

9.13.3.6 OnKeyPressed()

```
bool Vesper::EditorLayer::OnKeyPressed (
            KeyPressedEvent & e)  [private]
```

```
00670     {
00671         // Shortcuts
00672         if (e.GetRepeatCount() > 0)
00673             return false;
00674 
00675         bool control = Input::IsKeyPressed(VZ_KEY_LEFT_CONTROL) ||
      Input::IsKeyPressed(VZ_KEY_RIGHT_CONTROL);
00676         bool shift = Input::IsKeyPressed(VZ_KEY_LEFT_SHIFT) ||
      Input::IsKeyPressed(VZ_KEY_RIGHT_SHIFT);
00677         switch (e.GetKeyCode())
00678         {
00679             // Scene Shortcuts
00680         case VZ_KEY_N:
00681 
00682             if (control)
00683             {
00684                 NewScene();
00685             }
00686             break;
00687 
00688         case VZ_KEY_O:
00689             if (control)
00690             {
00691                 OpenScene();
00692             }
00693             break;
00694 
00695         case VZ_KEY_S:
00696             if (control && shift)
00697             {
00698                 SaveSceneAs();
00699             }
00700             break;
00701 
00702 
00703             // Gizmo Shortcuts
00704         case VZ_KEY_Q:
00705             m_GizmoType = -1;
00706             break;
00707         case VZ_KEY_W:
00708             m_GizmoType = ImGuizmo::OPERATION::TRANSLATE;
00709             break;
00710         case VZ_KEY_E:
00711             m_GizmoType = ImGuizmo::OPERATION::ROTATE;
00712             break;
00713         case VZ_KEY_R:
00714             m_GizmoType = ImGuizmo::OPERATION::SCALE;
00715             break;
00716         }
00717         return false;
00718 
00719     }
```

References Vesper::KeyEvent::GetKeyCode(), Vesper::KeyPressedEvent::GetRepeatCount(), Vesper::Input::IsKeyPressed(), m_GizmoType, NewScene(), OpenScene(), and SaveSceneAs().

Referenced by OnEvent().

9.13.3.7 OnUpdate()

```
void Vesper::EditorLayer::OnUpdate (
            Timestep ts)  [override], [virtual]
```

C++ test code scenes

TODO: get it to animate through texture sheet sub texture indices

Reimplemented from Vesper::Layer.

```
00235     {
00236         VZ_PROFILE_FUNCTION();
```

```
00237
00238         // Resize
00239         if (Vesper::FramebufferSpecification spec = m_Framebuffer->GetSpecification();
00240             m_ViewportSize.x > 0.0f && m_ViewportSize.y > 0.0f &&
00241             (spec.Width != (uint32_t)m_ViewportSize.x || spec.Height != (uint32_t)m_ViewportSize.y))
00242         {
00243             m_Framebuffer->Resize((uint32_t)m_ViewportSize.x, (uint32_t)m_ViewportSize.y);
00244             m_CameraController.OnResize(m_ViewportSize.x, m_ViewportSize.y);
00245             m_EditorCamera.SetViewportSize(m_ViewportSize.x, m_ViewportSize.y);
00246             m_ActiveScene->OnViewportResize((uint32_t)m_ViewportSize.x, (uint32_t)m_ViewportSize.y);
00247         }
00248
00249         // Update
00250         if (m_ViewportFocused)
00251         {
00252             m_CameraController.OnUpdate(ts);
00253         }
00254             m_EditorCamera.OnUpdate(ts);
00255
00256
00257         // Render
00258         Renderer2D::ResetStats();
00259         {
00260             VZ_PROFILE_SCOPE("Renderer Prep");
00261             m_Framebuffer->Bind();
00262             RenderCommand::SetClearColor(m_ClearColor);
00263             RenderCommand::Clear();
00264         }
00265
00266
00267         // Draw
00268         {
00269             static float rotation = 0.0f;
00270             rotation += ts * 50.0f;
00271             VZ_PROFILE_SCOPE("Renderer2D Draw");
00272
00274             {
00275                 // Basic scene
00276                 if (scene1)
00277                 {
00278                     VZ_PROFILE_SCOPE("Scene 1");
00279                     Renderer2D::BeginScene(m_CameraController.GetCamera());
00280
00281                     // Checkerboard background
00282                     Renderer2D::DrawQuadWithTexture({ 0.0f, 0.0f, -0.25f }, { 25.0f, 25.0f },
      m_CheckerboardTexture, 10.0f, m_BackgroundColor);
00283
00284                     // Squares
00285                     Renderer2D::DrawQuadRotated({ 0.0f, 1.25f, -0.165f }, { 1.0f, 1.0f },
      glm::radians(45.0f + m_squareRotation + rotation), m_SquareColor);
00286
00287                     // Rotated Squares
00288                     Renderer2D::DrawQuadRotatedWithTexture({ 0.0f, 1.25f, -0.15f }, { 0.75f, 0.75f },
      m_CheckerboardTexture, glm::radians(m_squareRotation * m_specialQuadRotation * rotation),
      m_textureScale, m_SpecialQuadColor);
00289
00290                     Renderer2D::DrawQuadRotatedWithTexture({ 2.0f, -0.25f, -0.15f }, { 1.0f, 1.0f },
      m_CheckerboardTexture, glm::radians(m_squareRotation + rotation), m_textureScale,
      m_TextureTintColor1);
00291                     Renderer2D::DrawQuadRotatedWithTexture({ -2.0f, -0.25f, -0.15f }, { 1.0f, 1.0f },
      m_CheckerboardTexture, glm::radians(m_squareRotation + rotation), m_textureScale,
      m_TextureTintColor2);
00292
00293                     glm::vec3 startPos = { 0.0f, 0.0f, -0.175f };
00294                     //Renderer2D::DrawQuad(pos, { 0.8f, 0.8f }, { 0.8f, 0.2f, 0.3f, 1.0f });
00295                     glm::vec3 finalPos = startPos;
00296                     float offset = 0.9f;
00297                     for (int y = -10; y <= 10; y++)
00298                     {
00299                         for (int x = -10; x <= 10; x++)
00300                         {
00301                             glm::vec3 newPos = { startPos.x - x * offset, startPos.y - y * offset,
      startPos.z };
00302                             Renderer2D::DrawQuad(newPos, { 0.8f, 0.8f }, { (x + 5) / 10.0f, 0.4f, (y +
      5) / 10.0f, 1.0f });
00303                             finalPos = newPos;
00304                         }
00305                     }
00306                     Renderer2D::DrawQuad(finalPos, { 0.8f, 0.8f }, { 0.8f, 0.2f, 0.3f, 1.0f });
00307
00308                     Renderer2D::EndScene();
00309
00310                 }
00311
00312                 // Sprite sheet scene
00313                 if (scene2)
00314                 {
```

```
00315                     VZ_PROFILE_SCOPE("Scene 2");
00316
00317                     Renderer2D::BeginScene(m_CameraController.GetCamera());
00318
00319                     // Sprite sheet drawn as full texture
00320                     Renderer2D::DrawQuadWithTexture({ -1.0f, 1.5f, 0.5f }, { 1, 1 },
    m_SpriteSheetFire, 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00321                     Renderer2D::DrawQuadRotatedWithTexture({ 1.5f, 0.0f, 0.0f }, { 1.78f, 1.0f },
    m_SpriteSheetTown, 0, 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00322
00323
00324                     // Sprite sheet drawn as full texture rotated
00325                     Renderer2D::DrawQuadRotatedWithTexture({ -1.5f, 0.0f, 0.0f }, { 1.78f, 1.0f },
    m_SpriteSheetTown, glm::radians(-rotation), 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00326
00327                     // Sub texture from tilesheet
00328                     Renderer2D::DrawQuadWithTexture({ 2.0f, -1.5f, 0.0f }, { 1.0f, 1.0f },
    m_SubTextureTown, 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00329
00330                     // Grid of sub textures from tilesheet
00331                     for (int y = -5; y < 5; y++)
00332                     {
00333                         for (int x = -5; x < 5; x++)
00334                         {
00335                             glm::vec3 pos = glm::vec3(x * 0.09f, y * 0.09f, -0.09f);
00336                             Renderer2D::DrawQuadWithTexture(pos, { 0.1f, 0.1f }, m_SubTextureTown,
    1.0f, glm::vec4(1.0f));
00337                         }
00338                     }
00339
00341                     Renderer2D::DrawQuadRotatedWithTexture({ 0.0f, -1.5f, 0.0f }, { 1.0f, 1.0f },
    m_SubTextureFire, 0, 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00342
00343                     Renderer2D::EndScene();
00344                 }
00345
00346                 // Tile map scene
00347                 if (scene3)
00348                 {
00349                     VZ_PROFILE_SCOPE("Scene 3");
00350                     Renderer2D::BeginScene(m_CameraController.GetCamera());
00351                     for (uint32_t y = 0; y < s_MapHeight; y++)
00352                     {
00353                         for (uint32_t x = 0; x < s_MapWidth; x++)
00354                         {
00355                             char tileChar = s_MapTiles[x + y * s_MapWidth];
00356                             Ref<SubTexture2D> texture;
00357                             if (s_TextureMap.find(tileChar) != s_TextureMap.end())
00358                                 texture = s_TextureMap[tileChar];
00359                             else
00360                                 texture = s_TextureMap['G']; // Default to grass
00361
00362                             Renderer2D::DrawQuadWithTexture({ x - s_MapWidth / 2.0f, s_MapHeight - y -
    s_MapHeight / 2.0f, 0.1f }, { 1.0f, 1.0f }, texture, 1.0f, glm::vec4(1.0f));
00363
00364                         }
00365                     }
00366                     Renderer2D::EndScene();
00367                 }
00368
00369                 // Particle scene
00370                 if (scene4)
00371                 {
00372                     VZ_PROFILE_SCOPE("Scene 4");
00373                     Renderer2D::BeginScene(m_CameraController.GetCamera());
00374                     for (float y = -5.0f; y < 5.0f; y += 0.5f)
00375                     {
00376                         for (float x = -5.0f; x < 5.0f; x += 0.5f)
00377                         {
00378                             glm::vec4 color = { (x + 5.0f) / 10.0f, 0.4f,(y + 5.0f) / 10.0f, 0.35f };
00379                             Renderer2D::DrawQuad({ x, y }, { 0.45f, 0.45f }, color);
00380                         }
00381                     }
00382
00383                     if (Input::IsMouseButtonPressed(VZ_MOUSE_BUTTON_LEFT) && m_ViewportHovered)
00384                     {
00385                         ImVec2 mousePos = ImGui::GetMousePos();
00386
00387                         mousePos.x -= m_ViewportBounds[0].x;
00388                         mousePos.y -= m_ViewportBounds[0].y;
00389
00390                         if (mousePos.x < 0 || mousePos.y < 0 || mousePos.x > m_ViewportSize.x ||
    mousePos.y > m_ViewportSize.y)
00391                             return;
00392
00393                         auto bounds = m_CameraController.GetBounds();
00394                         auto camPos = m_CameraController.GetPosition();
```

```
00395
00396                           float width = m_ViewportSize.x;
00397                           float height = m_ViewportSize.y;
00398
00399                           m_ParticleProps.Position.x = (mousePos.x / width) * bounds.GetWidth() -
      bounds.GetWidth() * 0.5f + camPos.x;
00400                           m_ParticleProps.Position.y = bounds.GetHeight() * 0.5f - (mousePos.y / height)
      * bounds.GetHeight() + camPos.y;
00401
00402                           for (int i = 0; i < ParticleEmitCount; i++) {
00403                               m_ParticleSystem.Emit(m_ParticleProps);
00404                           }
00405                       }
00406                       m_ParticleSystem.OnUpdate(ts);
00407                       m_ParticleSystem.OnRender(m_CameraController.GetCamera());
00408                       Renderer2D::EndScene();
00409
00410                   }
00411               }
00412
00413               if (useEntityScene)
00414               {
00415                   VZ_PROFILE_SCOPE("Entity Scene Update");
00416                   // Update scene
00417                       m_ActiveScene->OnUpdateRuntime(ts);
00418                       m_ActiveScene->OnUpdateEditor(ts, m_EditorCamera);
00419               }
00420           }
00421
00422           m_Framebuffer->Unbind();
00423       }
```

References Vesper::RenderCommand::Clear(), Vesper::Renderer2D::EndScene(), Vesper::Input::IsMouseButtonPressed(), m_ViewportFocused, m_ViewportHovered, ParticleEmitCount, Vesper::Renderer2D::ResetStats(), s_MapHeight, s_MapTiles, s_MapWidth, scene1, scene2, scene3, scene4, and useEntityScene.

9.13.3.8 OpenScene()

```
void Vesper::EditorLayer::OpenScene ()  [private]
```

```
00729     {
00730         std::string filePath = FileDialogs::OpenFile("Vesper Scene (*.vesper)\0*.vesper\0");
00731
00732         if (!filePath.empty())
00733         {
00734             m_ActiveScene = CreateRef<Scene>();
00735             m_ActiveScene->OnViewportResize((uint32_t)m_ViewportSize.x, (uint32_t)m_ViewportSize.y);
00736             m_SceneHierarchyPanel.SetContext(m_ActiveScene);
00737
00738             SceneSerializer serializer(m_ActiveScene);
00739             serializer.Deserialize(filePath);
00740             VZ_CORE_INFO("Scene deserialized from: " + filePath);
00741         }
00742     }
```

References Vesper::SceneSerializer::Deserialize().

Referenced by OnImGuiRender(), and OnKeyPressed().

9.13.3.9 ResetScene()

```
void Vesper::EditorLayer::ResetScene ()  [private]
```

```
00756     {
00757         VZ_CORE_ASSERT(false, "Not implemented yet!");
00758     }
```

Referenced by OnImGuiRender().

9.13.3.10 SaveSceneAs()

```
void Vesper::EditorLayer::SaveSceneAs ()  [private]
```

```
00745     {
00746         std::string filePath = FileDialogs::SaveFile("Vesper Scene (*.vesper)\0*.vesper\0");
00747         if (!filePath.empty())
00748         {
00749             SceneSerializer serializer(m_ActiveScene);
00750             serializer.Serialize(filePath);
00751             VZ_CORE_INFO("Scene serialized to: " + filePath);
00752         }
00753     }
```

References Vesper::SceneSerializer::Serialize().

Referenced by OnImGuiRender(), and OnKeyPressed().

9.13.4 Member Data Documentation

9.13.4.1 lastFrameTime

```
float Vesper::EditorLayer::lastFrameTime = 0.0f  [private]
```

9.13.4.2 m_ActiveScene

```
Ref<Scene> Vesper::EditorLayer::m_ActiveScene  [private]
```

9.13.4.3 m_BackgroundColor

```
glm::vec4 Vesper::EditorLayer::m_BackgroundColor = { 0.1f, 0.1f, 0.1f, 1.0f }  [private]
```

```
00096 { 0.1f, 0.1f, 0.1f, 1.0f };
```

9.13.4.4 m_CameraController

```
OrthographicCameraController Vesper::EditorLayer::m_CameraController  [private]
```

9.13.4.5 m_CameraEntity

```
Entity Vesper::EditorLayer::m_CameraEntity  [private]
```

9.13.4.6 m_CheckerboardTexture

```
Ref<Texture2D> Vesper::EditorLayer::m_CheckerboardTexture  [private]
```

9.13.4.7 m_ClearColor

```
glm::vec4 Vesper::EditorLayer::m_ClearColor = { 0.1f, 0.3f, 0.3f, 1.0f }  [private]
```

```
00097 { 0.1f, 0.3f, 0.3f, 1.0f };
```

9.13.4.8 m_EditorCamera

```
EditorCamera Vesper::EditorLayer::m_EditorCamera  [private]
```

9.13.4.9 m_EditorScene

```
Ref<Scene> Vesper::EditorLayer::m_EditorScene  [private]
```

9.13.4.10 m_FireEntity

```
Entity Vesper::EditorLayer::m_FireEntity  [private]
```

9.13.4.11 m_FlatColorShader

```
Ref<Shader> Vesper::EditorLayer::m_FlatColorShader  [private]
```

9.13.4.12 m_Framebuffer

```
Ref<Framebuffer> Vesper::EditorLayer::m_Framebuffer  [private]
```

9.13.4.13 m_GizmoType

```
int Vesper::EditorLayer::m_GizmoType = -1  [private]
```

Referenced by OnImGuiRender(), and OnKeyPressed().

9.13.4.14 m_ParticleProps

```
ParticleProps Vesper::EditorLayer::m_ParticleProps  [private]
```

9.13.4.15 m_ParticleSystem

```
ParticleSystem Vesper::EditorLayer::m_ParticleSystem  [private]
```

9.13.4.16 m_PrimaryCamera

```
bool Vesper::EditorLayer::m_PrimaryCamera = true  [private]
```

9.13.4.17 m_RotationSnap

```
float Vesper::EditorLayer::m_RotationSnap = 45.0f  [private]
```

Referenced by OnImGuiRender().

9.13.4.18 m_ScaleSnap

```
float Vesper::EditorLayer::m_ScaleSnap = 0.5f  [private]
```

Referenced by OnImGuiRender().

9.13.4.19 m_SceneHierarchyPanel

```
SceneHierarchyPanel Vesper::EditorLayer::m_SceneHierarchyPanel  [private]
```

9.13.4.20 m_SceneState

```
SceneState Vesper::EditorLayer::m_SceneState = SceneState::Edit  [private]
```

Referenced by OnEvent().

9.13.4.21 m_SmokeEntity

```
Entity Vesper::EditorLayer::m_SmokeEntity  [private]
```

9.13.4.22 m_SpecialQuadColor

```
glm::vec4 Vesper::EditorLayer::m_SpecialQuadColor = { 0.9f, 0.2f, 0.8f, 1.0f }  [private]
00098 { 0.9f, 0.2f, 0.8f, 1.0f };
```

9.13.4.23 m_specialQuadRotation

```
float Vesper::EditorLayer::m_specialQuadRotation = 0.5f  [private]
```

9.13.4.24 m_SpriteSheetCrystals

```
Ref<Texture2D> Vesper::EditorLayer::m_SpriteSheetCrystals  [private]
```

9.13.4.25 m_SpriteSheetCursedLands

```
Ref<Texture2D> Vesper::EditorLayer::m_SpriteSheetCursedLands  [private]
```

9.13.4.26 m_SpriteSheetFire

```
Ref<Texture2D> Vesper::EditorLayer::m_SpriteSheetFire  [private]
```

9.13.4.27 m_SpriteSheetRocks

```
Ref<Texture2D> Vesper::EditorLayer::m_SpriteSheetRocks  [private]
```

9.13.4.28 m_SpriteSheetSmoke

```
Ref<Texture2D> Vesper::EditorLayer::m_SpriteSheetSmoke  [private]
```

9.13.4.29 m_SpriteSheetTown

```
Ref<Texture2D> Vesper::EditorLayer::m_SpriteSheetTown  [private]
```

9.13.4.30 m_SquareColor

```
glm::vec4 Vesper::EditorLayer::m_SquareColor = { 0.2f, 0.3f, 0.8f, 1.0f }  [private]
00093 { 0.2f, 0.3f, 0.8f, 1.0f };
```

9.13.4.31 m_squareRotation

```
float Vesper::EditorLayer::m_squareRotation = 25.0f  [private]
```

9.13.4.32 m_SquareVA

```
Ref<VertexArray> Vesper::EditorLayer::m_SquareVA  [private]
```

9.13.4.33 m_SubTextureFire

```
Ref<SubTexture2D> Vesper::EditorLayer::m_SubTextureFire  [private]
```

9.13.4.34 m_SubTextureSmoke

```
Ref<SubTexture2D> Vesper::EditorLayer::m_SubTextureSmoke  [private]
```

9.13.4.35 m_SubTextureTown

```
Ref<SubTexture2D> Vesper::EditorLayer::m_SubTextureTown  [private]
```

9.13.4.36 m_textureScale

```
float Vesper::EditorLayer::m_textureScale = 1.0f  [private]
```

9.13.4.37 m_TextureTintColor1

```
glm::vec4 Vesper::EditorLayer::m_TextureTintColor1 = { 1.0f, 1.0f, 1.0f, 1.0f }  [private]
00094 { 1.0f, 1.0f, 1.0f, 1.0f };
```

9.13.4.38 m_TextureTintColor2

```
glm::vec4 Vesper::EditorLayer::m_TextureTintColor2 = { 1.0f, 1.0f, 1.0f, 1.0f }  [private]
00095 { 1.0f, 1.0f, 1.0f, 1.0f };
```

9.13.4.39 m_TranslationSnap

```
float Vesper::EditorLayer::m_TranslationSnap = 0.5f  [private]
```

Referenced by OnImGuiRender().

9.13.4.40 m_UseSpecialQuadColor

```
bool Vesper::EditorLayer::m_UseSpecialQuadColor = false  [private]
```

9.13.4.41 m_ViewportBounds

```
glm::vec2 Vesper::EditorLayer::m_ViewportBounds[2] = { {0,0}, {0,0} }  [private]
00046 { {0,0}, {0,0} };
```

9.13.4.42 m_ViewportFocused

```
bool Vesper::EditorLayer::m_ViewportFocused = false  [private]
```

Referenced by OnImGuiRender(), and OnUpdate().

9.13.4.43 m_ViewportHovered

```
bool Vesper::EditorLayer::m_ViewportHovered = false  [private]
```

Referenced by OnImGuiRender(), and OnUpdate().

9.13.4.44 m_ViewportSize

```
glm::vec2 Vesper::EditorLayer::m_ViewportSize = {0,0}  [private]
00045 {0,0};
```

9.13.4.45 ParticleEmitCount

```
int Vesper::EditorLayer::ParticleEmitCount = 100  [private]
```

Referenced by OnUpdate().

9.13.4.46 s_TextureMap

```
std::unordered_map<char, Ref<SubTexture2D> > Vesper::EditorLayer::s_TextureMap  [private]
```

9.13.4.47 scene1

```
bool Vesper::EditorLayer::scene1 = false  [private]
```

Referenced by OnImGuiRender(), and OnUpdate().

9.13.4.48 scene2

```
bool Vesper::EditorLayer::scene2 = false  [private]
```

Referenced by OnImGuiRender(), and OnUpdate().

9.13.4.49 scene3

```
bool Vesper::EditorLayer::scene3 = false  [private]
```

Referenced by OnImGuiRender(), and OnUpdate().

9.13.4.50 scene4

```
bool Vesper::EditorLayer::scene4 = false  [private]
```

Referenced by OnImGuiRender(), and OnUpdate().

9.13.4.51 useEntityScene

```
bool Vesper::EditorLayer::useEntityScene = true  [private]
```

Referenced by OnImGuiRender(), and OnUpdate().

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

- Vesper-Editor/src/EditorLayer.h
- Vesper-Editor/src/EditorLayer.cpp

9.14 Vesper::Entity Class Reference

`#include <Entity.h>`

Public Member Functions

- Entity ()=default
- Entity (entt::entity handle, Scene *scene)
- Entity (const Entity &other)=default
- template<typename T>
  bool HasComponent () const
- template<typename T, typename... Args>
  T & AddComponent (Args &&... args)
- template<typename T, typename... Args>
  T & AddOrReplaceComponent (Args &&... args)
- template<typename T>
  T & GetComponent ()
- template<typename T, typename... Args>
  T & GetOrAddComponent (Args &&... args)
- template<typename T>
  void RemoveComponent ()
- const UUID & GetID ()
- const std::string & GetName ()
- operator bool () const
- operator entt::entity () const
- operator uint32_t () const
- bool operator== (const Entity &other) const
- bool operator!= (const Entity &other) const

Private Attributes

- entt::entity m_EntityID {entt::null}
- Scene * m_Scene = nullptr

9.14.1 Constructor & Destructor Documentation

9.14.1.1 Entity() [1/3]

```
Vesper::Entity::Entity ()  [default]
```

9.14.1.2 Entity() [2/3]

```
Vesper::Entity::Entity (
            entt::entity handle,
            Scene * scene)
00007         : m_EntityID(handle), m_Scene(scene)
00008     {
00009     }
```

References Entity(), and m_Scene.

Referenced by Entity().

9.14.1.3 Entity() [3/3]

```
Vesper::Entity::Entity (
            const Entity & other)  [default]
```

9.14.2 Member Function Documentation

9.14.2.1 AddComponent()

```
template<typename T, typename...  Args>
T & Vesper::Entity::AddComponent (
            Args &&...  args)  [inline]
```

```
00024         {
00025             VZ_CORE_ASSERT(!HasComponent<T>(), "Entity already has component!");
00026             T& component = m_Scene->m_Registry.emplace<T>(m_EntityID, std::forward<Args>(args)...);
00027             m_Scene->OnComponentAdded<T>(*this, component);
00028             return component;
00029         }
```

9.14.2.2 AddOrReplaceComponent()

```
template<typename T, typename...  Args>
T & Vesper::Entity::AddOrReplaceComponent (
            Args &&...  args)  [inline]
```

```
00034         {
00035             return m_Scene->m_Registry.emplace_or_replace<T>(m_EntityID, std::forward<Args>(args)...);
00036         }
```

9.14.2.3 GetComponent()

```
template<typename T>
T & Vesper::Entity::GetComponent ()  [inline]
```

```
00040         {
00041             VZ_CORE_ASSERT(HasComponent<T>(), "Entity does not have component!");
00042             return m_Scene->m_Registry.get<T>(m_EntityID);
00043         }
```

9.14.2.4 GetID()

```
const UUID & Vesper::Entity::GetID ()  [inline]
```

```
00063         {
00064             return GetComponent<UUIDComponent>().ID;
00065         }
```

9.14.2.5 GetName()

```
const std::string & Vesper::Entity::GetName ()  [inline]
```

```
00068         {
00069             return GetComponent<NameComponent>().Name;
00070         }
```

9.14.2.6 GetOrAddComponent()

```
template<typename T, typename...  Args>
T & Vesper::Entity::GetOrAddComponent (
             Args &&...  args)  [inline]
00047         {
00048             if (HasComponent<T>())
00049                 return GetComponent<T>();
00050             else
00051                 return AddComponent<T>(std::forward<Args>(args)...);
00052         }
```

9.14.2.7 HasComponent()

```
template<typename T>
bool Vesper::Entity::HasComponent () const  [inline]
00018         {
00019             return m_Scene->m_Registry.all_of<T>(m_EntityID);
00020         }
```

9.14.2.8 operator bool()

```
Vesper::Entity::operator bool () const  [inline]
00073 { return m_EntityID != entt::null; }
```

9.14.2.9 operator entt::entity()

```
Vesper::Entity::operator entt::entity () const  [inline]
00074 { return m_EntityID; }
```

9.14.2.10 operator uint32_t()

```
Vesper::Entity::operator uint32_t () const  [inline]
00075 { return (uint32_t)m_EntityID; }
```

9.14.2.11 operator"!=()

```
bool Vesper::Entity::operator!= (
             const Entity & other) const  [inline]
00077 {   return !(*this == other);   }
```

9.14.2.12 operator==()

```
bool Vesper::Entity::operator== (
             const Entity & other) const  [inline]
00076 {   return m_EntityID == other.m_EntityID && m_Scene == other.m_Scene;  }
```

9.14.2.13 RemoveComponent()

```
template<typename T>
void Vesper::Entity::RemoveComponent ()  [inline]
```

```
00056         {
00057             VZ_CORE_ASSERT(HasComponent<T>(), "Entity does not have component!");
00058             m_Scene->m_Registry.remove<T>(m_EntityID);
00059         }
```

9.14.3 Member Data Documentation

9.14.3.1 m_EntityID

```
entt::entity Vesper::Entity::m_EntityID {entt::null}  [private]
```

```
00080 {entt::null};
```

9.14.3.2 m_Scene

```
Scene* Vesper::Entity::m_Scene = nullptr  [private]
```

Referenced by Entity().

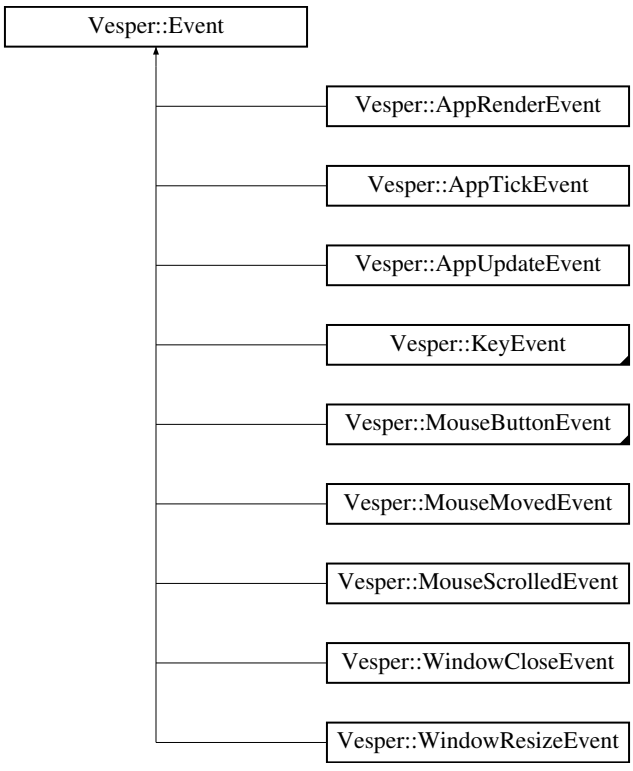
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/Scene/Entity.h
- Vesper/src/Vesper/Scene/Entity.cpp

9.15 Vesper::Event Class Reference

```
#include <Event.h>
```

Inheritance diagram for Vesper::Event:



Public Member Functions

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- virtual std::string ToString () const
- bool IsInCategory (EventCategory category)

Public Attributes

- bool Handled = false

Friends

- class EventDispatcher

9.15.1 Constructor & Destructor Documentation

9.15.1.1 ~Event()

```
virtual Vesper::Event::~Event ()  [virtual], [default]
```

9.15.2 Member Function Documentation

9.15.2.1 GetCategoryFlags()

```
virtual int Vesper::Event::GetCategoryFlags () const  [pure virtual]
```

Referenced by IsInCategory().

9.15.2.2 GetEventType()

```
virtual EventType Vesper::Event::GetEventType () const  [pure virtual]
```

9.15.2.3 GetName()

```
virtual const char * Vesper::Event::GetName () const  [pure virtual]
```

Referenced by ToString().

9.15.2.4 IsInCategory()

```
bool Vesper::Event::IsInCategory (
            EventCategory category)  [inline]
00044         {
00045             return GetCategoryFlags() & category;
00046         }
```

References GetCategoryFlags().

Referenced by Vesper::ImGuiLayer::OnEvent().

9.15.2.5 ToString()

```
virtual std::string Vesper::Event::ToString () const  [inline], [virtual]
```

Reimplemented in Vesper::KeyPressedEvent, Vesper::KeyReleasedEvent, Vesper::KeyTypedEvent, Vesper::MouseButtonPressedEve
Vesper::MouseButtonReleasedEvent, Vesper::MouseMovedEvent, Vesper::MouseScrolledEvent, and Vesper::WindowResizeEvent.

```
00041 { return GetName(); }
```

References GetName().

9.15.3 Friends And Related Symbol Documentation

9.15.3.1 EventDispatcher

```
friend class EventDispatcher  [friend]
```

9.15.4 Member Data Documentation

9.15.4.1 Handled

```
bool Vesper::Event::Handled = false
```

Referenced by Vesper::ImGuiLayer::OnEvent().

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

- Vesper/src/Vesper/Events/Event.h

9.16 Vesper::EventDispatcher Class Reference

```
#include <Event.h>
```

Public Member Functions

- EventDispatcher (Event &event)
- template<typename T>
  bool Dispatch (EventFn< T > func)

Private Types

- template<typename T>
  using EventFn = std::function<bool(T&)>

Private Attributes

- Event & m_Event

9.16.1 Member Typedef Documentation

9.16.1.1 EventFn

```
template<typename T>
using Vesper::EventDispatcher::EventFn = std::function<bool(T&)>  [private]
```

9.16.2 Constructor & Destructor Documentation

9.16.2.1 EventDispatcher()

```
Vesper::EventDispatcher::EventDispatcher (
            Event & event)  [inline]
00056           : m_Event(event)
00057       {
00058       }
```

References m_Event.

Referenced by Vesper::Application::OnEvent(), Vesper::EditorCamera::OnEvent(), Vesper::EditorLayer::OnEvent(), and Vesper::OrthographicCameraController::OnEvent().

9.16.3 Member Function Documentation

9.16.3.1 Dispatch()

```
template<typename T>
bool Vesper::EventDispatcher::Dispatch (
            EventFn< T > func)  [inline]
00062       {
00063           if (m_Event.GetEventType() == T::GetStaticType())
00064           {
00065               m_Event.Handled = func(*(T*)&m_Event);
00066               return true;
00067           }
00068           return false;
00069       }
```

9.16.4 Member Data Documentation

9.16.4.1 m_Event

```
Event& Vesper::EventDispatcher::m_Event  [private]
```

Referenced by EventDispatcher().

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

- Vesper/src/Vesper/Events/Event.h

9.17 Vesper::FileDialogs Class Reference

```
#include <PlatformUtils.h>
```

Static Public Member Functions

- static std::string OpenFile (const char *filter)
- static std::string SaveFile (const char *filter)

9.17.1 Member Function Documentation

9.17.1.1 OpenFile()

```
std::string Vesper::FileDialogs::OpenFile (
             const char * filter)  [static]
```

```
00012                                                    {
00013
00014         OPENFILENAMEA ofn;
00015         CHAR szFile[260] = { 0 };
00016         ZeroMemory(&ofn, sizeof(ofn));
00017         ofn.lStructSize = sizeof(ofn);
00018         ofn.hwndOwner =
      glfwGetWin32Window((GLFWwindow*)Vesper::Application::Get().GetWindow().GetNativeWindow());
00019         ofn.lpstrFile = szFile;
00020         ofn.nMaxFile = sizeof(szFile);
00021         ofn.lpstrFilter = filter;
00022         ofn.nFilterIndex = 1;
00023         ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST | OFN_NOCHANGEDIR;
00024         if (GetOpenFileNameA(&ofn) == TRUE)
00025             return std::string(ofn.lpstrFile);
00026         return std::string();
00027     }
```

9.17.1.2 SaveFile()

```
std::string Vesper::FileDialogs::SaveFile (
             const char * filter)  [static]
00029                                                    {
00030
00031         OPENFILENAMEA ofn;
00032         CHAR szFile[260] = { 0 };
00033         CHAR currentDir[256] = { 0 };
00034         ZeroMemory(&ofn, sizeof(OPENFILENAME));
00035         ofn.lStructSize = sizeof(OPENFILENAME);
00036         ofn.hwndOwner =
      glfwGetWin32Window((GLFWwindow*)Application::Get().GetWindow().GetNativeWindow());
00037         ofn.lpstrFile = szFile;
00038         ofn.nMaxFile = sizeof(szFile);
00039         if (GetCurrentDirectoryA(256, currentDir))
00040             ofn.lpstrInitialDir = currentDir;
00041         ofn.lpstrFilter = filter;
00042         ofn.nFilterIndex = 1;
00043         ofn.Flags = OFN_PATHMUSTEXIST | OFN_OVERWRITEPROMPT | OFN_NOCHANGEDIR;
00044
00045         // Sets the default extension by extracting it from the filter
00046         ofn.lpstrDefExt = strchr(filter, '\0') + 1;
00047
00048         if (GetSaveFileNameA(&ofn) == TRUE)
00049             return ofn.lpstrFile;
00050
00051         return std::string();
00052
00053     }
```

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

- Vesper/src/Vesper/Utils/PlatformUtils.h
- Vesper/src/Platform/Windows/WindowsPlatformUtils.cpp

9.18 Vesper::FileSystem Class Reference

```
#include <PlatformUtils.h>
```

Static Public Member Functions

- static void Initialize ()
- static std::string GetCurrentWorkingDirectory ()
- static std::string GetAbsolutePath (const std::string &relativePath)
- static std::string GetTravelingUpPath (const std::string &path)
- static bool IsInitialized ()

Static Public Attributes

- static bool m_Initialized = false
- static std::string m_RootEngineDirectory = ""
- static std::string m_RootEditorDirectory = ""
- static std::string m_ResourcesDirectory = ""
- static std::string m_AssetsDirectory = ""
- static std::string m_ProjectsDirectory = ""
- static std::string m_CurrentProjectDirectory = ""

9.18.1 Member Function Documentation

9.18.1.1 GetAbsolutePath()

```
std::string Vesper::FileSystem::GetAbsolutePath (
            const std::string & relativePath)  [static]
```

```
00104                                                                    {
00105         char fullPath[MAX_PATH];
00106         if (_fullpath(fullPath, relativePath.c_str(), MAX_PATH) != nullptr) {
00107             return std::string(fullPath);
00108         }
00109         return std::string();
00110     }
```

Referenced by Initialize().

9.18.1.2 GetCurrentWorkingDirectory()

```
std::string Vesper::FileSystem::GetCurrentWorkingDirectory ()  [static]
```

```
00097                                                 {
00098         CHAR currentDir[256] = { 0 };
00099         if (GetCurrentDirectoryA(256, currentDir))
00100             return std::string(currentDir);
00101         return std::string();
00102     }
```

9.18.1.3 GetTravelingUpPath()

```
std::string Vesper::FileSystem::GetTravelingUpPath (
            const std::string & path)  [static]
```

```
00112                                                          {
00113         size_t pos = path.find_last_of("/\\");
00114         if (pos != std::string::npos) {
00115             return path.substr(0, pos);
00116         }
00117         return std::string();
00118     }
```

Referenced by Initialize().

9.18.1.4 Initialize()

```
void Vesper::FileSystem::Initialize ()  [static]
```

```
00066                                {
00067         if (m_Initialized)
00068             return;
00069         m_Initialized = true;
00070
00071         // Set Root Engine Directory
00072         m_RootEngineDirectory =
      GetTravelingUpPath(GetTravelingUpPath(GetTravelingUpPath(GetAbsolutePath("."))));
00073         // Set Root Editor Directory
00074         m_RootEditorDirectory = GetAbsolutePath("../../Vesper-Editor/");
00075         // Set Resources Directory
00076         m_ResourcesDirectory = GetAbsolutePath(m_RootEngineDirectory + "/Resources/");
00077         // Set Assets Directory
00078         m_AssetsDirectory = GetAbsolutePath(m_RootEditorDirectory + "/assets");
00079         // Set Projects Directory
00080         m_ProjectsDirectory = GetAbsolutePath(m_RootEngineDirectory + "/Projects");
00081         // Set Current Project Directory
00082         m_CurrentProjectDirectory = GetAbsolutePath(m_ProjectsDirectory + "/DefaultProject");
00083
00084
00085
00086         // Log Directories for Debugging
```

```
00087         VZ_CORE_TRACE("FileSystem initialized");
00088         VZ_CORE_TRACE("FileSystem Directories:");
00089         VZ_CORE_TRACE("Root Engine Directory : " + m_RootEngineDirectory);
00090         VZ_CORE_TRACE("Root Editor Directory : " + m_RootEditorDirectory);
00091         VZ_CORE_TRACE("Resources Directory   : " + m_ResourcesDirectory);
00092         VZ_CORE_TRACE("Assets Directory      : " + m_AssetsDirectory);
00093         VZ_CORE_TRACE("Projects Directory    : " + m_ProjectsDirectory);
00094         VZ_CORE_TRACE("Current Project Directory : " + m_CurrentProjectDirectory);
00095     }
```

References GetAbsolutePath(), GetTravelingUpPath(), and m_Initialized.

Referenced by Vesper::EditorLayer::OnAttach().

9.18.1.5 IsInitialized()

```
bool Vesper::FileSystem::IsInitialized ()  [inline], [static]
00022 { return m_Initialized; }
```

References m_Initialized.

9.18.2 Member Data Documentation

9.18.2.1 m_AssetsDirectory

```
std::string Vesper::FileSystem::m_AssetsDirectory = ""  [static]
```

9.18.2.2 m_CurrentProjectDirectory

```
std::string Vesper::FileSystem::m_CurrentProjectDirectory = ""  [static]
```

9.18.2.3 m_Initialized

```
bool Vesper::FileSystem::m_Initialized = false  [static]
```

Referenced by Initialize(), and IsInitialized().

9.18.2.4 m_ProjectsDirectory

```
std::string Vesper::FileSystem::m_ProjectsDirectory = ""  [static]
```

9.18.2.5 m_ResourcesDirectory

```
std::string Vesper::FileSystem::m_ResourcesDirectory = ""  [static]
```

9.18.2.6 m_RootEditorDirectory

```
std::string Vesper::FileSystem::m_RootEditorDirectory = ""  [static]
```

9.18.2.7 m_RootEngineDirectory

```
std::string Vesper::FileSystem::m_RootEngineDirectory = ""  [static]
```

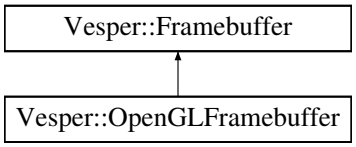
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/Utils/PlatformUtils.h
- Vesper/src/Platform/Windows/WindowsPlatformUtils.cpp

9.19 Vesper::Framebuffer Class Reference

```
#include <Framebuffer.h>
```

Inheritance diagram for Vesper::Framebuffer:



Public Member Functions

- ~Framebuffer ()=default
- virtual void Bind ()=0
- virtual void Unbind ()=0
- virtual void Resize (uint32_t width, uint32_t height)=0
- virtual uint32_t GetColorAttachmentRendererID () const =0
- virtual const FramebufferSpecification & GetSpecification () const =0

Static Public Member Functions

- static Ref< Framebuffer > Create (const FramebufferSpecification &spec)

9.19.1 Constructor & Destructor Documentation

9.19.1.1 ~Framebuffer()

```
Vesper::Framebuffer::~Framebuffer ()  [default]
```

9.19.2 Member Function Documentation

9.19.2.1 Bind()

```
virtual void Vesper::Framebuffer::Bind ()  [pure virtual]
```

Implemented in Vesper::OpenGLFramebuffer.

9.19.2.2 Create()

```
Ref< Framebuffer > Vesper::Framebuffer::Create (
            const FramebufferSpecification & spec)  [static]
```

```
00010     {
00011         switch (Renderer::GetAPI())
00012         {
00013             case RendererAPI::API::None:    VZ_CORE_ASSERT(false, "RendererAPI::None is currently not
      supported!"); return nullptr;
00014             case RendererAPI::API::OpenGL:  return CreateRef<OpenGLFramebuffer>(spec);
00015         }
00016
00017         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00018         return nullptr;
00019     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.19.2.3 GetColorAttachmentRendererID()

```
virtual uint32_t Vesper::Framebuffer::GetColorAttachmentRendererID () const  [pure virtual]
```

Implemented in Vesper::OpenGLFramebuffer.

9.19.2.4 GetSpecification()

```
virtual const FramebufferSpecification & Vesper::Framebuffer::GetSpecification () const  [pure
virtual]
```

Implemented in Vesper::OpenGLFramebuffer.

9.19.2.5 Resize()

```
virtual void Vesper::Framebuffer::Resize (
            uint32_t width,
            uint32_t height)  [pure virtual]
```

Implemented in Vesper::OpenGLFramebuffer.

9.19.2.6 Unbind()

```
virtual void Vesper::Framebuffer::Unbind ()  [pure virtual]
```

Implemented in Vesper::OpenGLFramebuffer.

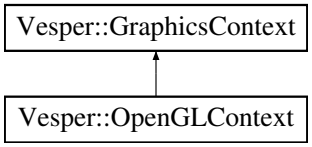
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/Renderer/Framebuffer.h
- Vesper/src/Vesper/Renderer/Framebuffer.cpp

9.20 Vesper::GraphicsContext Class Reference

`#include <GraphicsContext.h>`

Inheritance diagram for Vesper::GraphicsContext:



Public Member Functions

- virtual ~GraphicsContext ()
- virtual void Init ()=0
- virtual void SwapBuffers ()=0

9.20.1 Constructor & Destructor Documentation

9.20.1.1 ~GraphicsContext()

```
virtual Vesper::GraphicsContext::~GraphicsContext ()  [inline], [virtual]
00010 {}
```

9.20.2 Member Function Documentation

9.20.2.1 Init()

```
virtual void Vesper::GraphicsContext::Init ()  [pure virtual]
```

Implemented in Vesper::OpenGLContext.

Referenced by Vesper::WindowsWindow::Init().

9.20.2.2 SwapBuffers()

```
virtual void Vesper::GraphicsContext::SwapBuffers ()  [pure virtual]
```

Implemented in Vesper::OpenGLContext.

Referenced by Vesper::WindowsWindow::OnUpdate().

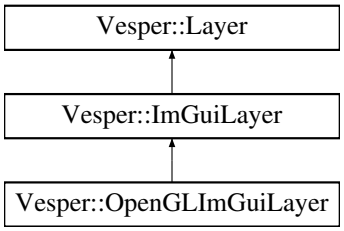
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Renderer/GraphicsContext.h

9.21 Vesper::ImGuiLayer Class Reference

`#include <ImGuiLayer.h>`

Inheritance diagram for Vesper::ImGuiLayer:



Public Member Functions

- ImGuiLayer ()
- ~ImGuiLayer ()
- virtual void OnAttach () override
- virtual void OnDetach () override
- virtual void OnImGuiRender () override
- virtual void OnEvent (Event &e) override
- virtual void Begin ()
- virtual void End ()
- virtual void SetBlockEvents (bool block)
- virtual void SetDarkThemeColors ()

Public Member Functions inherited from Vesper::Layer

- Layer (const std::string &name="Layer")
- virtual ~Layer ()
- virtual void OnUpdate (Timestep ts)
- virtual void OnRender ()
- const std::string & GetName () const

Protected Attributes

- bool m_BlockEvents = true
- float m_Time = 0.0f

Protected Attributes inherited from Vesper::Layer

- std::string m_DebugName

9.21.1 Constructor & Destructor Documentation

9.21.1.1 ImGuiLayer()

```
Vesper::ImGuiLayer::ImGuiLayer ()
```

```
00021                            : Layer("ImGuiLayer")
00022     {
00023     }
```

Referenced by Vesper::Application::Application(), and Vesper::OpenGLImGuiLayer::OpenGLImGuiLayer().

9.21.1.2 ~ImGuiLayer()

```
Vesper::ImGuiLayer::~ImGuiLayer ()
```

```
00026     {
00027     }
```

9.21.2 Member Function Documentation

9.21.2.1 Begin()

```
void Vesper::ImGuiLayer::Begin ()  [virtual]
```

Reimplemented in Vesper::OpenGLImGuiLayer.

```
00096     {
00097         VZ_PROFILE_FUNCTION();
00098         ImGui_ImplOpenGL3_NewFrame();
00099         ImGui_ImplGlfw_NewFrame();
00100         ImGui::NewFrame();
00101         ImGuizmo::BeginFrame();
00102     }
```

Referenced by Vesper::OpenGLImGuiLayer::Begin(), and Vesper::Application::Run().

9.21.2.2 End()

```
void Vesper::ImGuiLayer::End ()  [virtual]
```

Reimplemented in Vesper::OpenGLImGuiLayer.

```
00105     {
00106         VZ_PROFILE_FUNCTION();
00107         ImGuiIO& io = ImGui::GetIO();
00108         Application& app = Application::Get();
00109         io.DisplaySize = ImVec2((float)app.GetWindow().GetWidth(),
      (float)app.GetWindow().GetHeight());
00110         ImGui::Render();
00111         ImGui_ImplOpenGL3_RenderDrawData(ImGui::GetDrawData());
00112         if (io.ConfigFlags & ImGuiConfigFlags_ViewportsEnable)
00113         {
00114             GLFWwindow* backup_current_context = glfwGetCurrentContext();
00115             ImGui::UpdatePlatformWindows();
00116             ImGui::RenderPlatformWindowsDefault();
00117             glfwMakeContextCurrent(backup_current_context);
00118         }
00119     }
```

References Vesper::Application::Get(), Vesper::Window::GetHeight(), Vesper::Window::GetWidth(), and Vesper::Application::GetWinc

Referenced by Vesper::OpenGLImGuiLayer::End(), and Vesper::Application::Run().

9.21.2.3 OnAttach()

```
void Vesper::ImGuiLayer::OnAttach ()  [override], [virtual]
```

TODO: Remove to openGL specific

Reimplemented from Vesper::Layer.

Reimplemented in Vesper::OpenGLImGuiLayer.

```
00030     {
00031         VZ_PROFILE_FUNCTION();
00032         IMGUI_CHECKVERSION();
00033         ImGui::CreateContext();
00034         ImGuiIO& io = ImGui::GetIO(); (void)io;
00035         //io.ConfigFlags |= ImGuiConfigFlags_NavEnableKeyboard;     // Enable Keyboard Controls
00036         //io.ConfigFlags |= ImGuiConfigFlags_NavEnableGamepad;      // Enable Gamepad Controls
00037         io.ConfigFlags |= ImGuiConfigFlags_DockingEnable;         // Enable Docking
00038         io.ConfigFlags |= ImGuiConfigFlags_ViewportsEnable;       // Enable Multi-Viewport / Platform
      Windows
00039         //io.ConfigFlags |= ImGuiConfigFlags_ViewportsNoTaskBarIcons; // Disable Platform Windows task
      bar icons
00040         //io.ConfigFlags |= ImGuiConfigFlags_ViewportsNoMerge;      // Disable Platform Windows
      merging into host window
00041
00042
      io.Fonts->AddFontFromFileTTF("../../Vesper-Editor/assets/fonts/RedHatMono/static/RedHatMono-Bold.ttf",
      18.0f);
00043         io.FontDefault =
      io.Fonts->AddFontFromFileTTF("../../Vesper-Editor/assets/fonts/RedHatMono/static/RedHatMono-Light.ttf",
      18.0f);
00044
00045
00046         ImGui::StyleColorsDark();
00047
00048         // When viewports are enabled we tweak WindowRounding/WindowBg so platform windows can look
      identical to regular ones.
00049         ImGuiStyle& style = ImGui::GetStyle();
00050         if (io.ConfigFlags & ImGuiConfigFlags_ViewportsEnable)
00051         {
00052             style.WindowRounding = 0.0f;
00053             style.Colors[ImGuiCol_WindowBg].w = 1.0f;
00054         }
00055
00056         SetDarkThemeColors();
00057
00058
00060         {
00061             Application& app = Application::Get();
00062             GLFWwindow* window = static_cast<GLFWwindow*>(app.GetWindow().GetNativeWindow());
00063             ImGui_ImplGlfw_InitForOpenGL(window, true);
00064             ImGui_ImplOpenGL3_Init("#version 410");
00065         }
00066
00067     }
```

References Vesper::Application::Get(), and SetDarkThemeColors().

Referenced by Vesper::OpenGLImGuiLayer::OnAttach().

9.21.2.4 OnDetach()

```
void Vesper::ImGuiLayer::OnDetach ()  [override], [virtual]
```

TODO: Remove to openGL specific

Reimplemented from Vesper::Layer.

Reimplemented in Vesper::OpenGLImGuiLayer.

```
00070     {
00071         VZ_PROFILE_FUNCTION();
00072
00074         {
00075             ImGui_ImplOpenGL3_Shutdown();
00076             ImGui_ImplGlfw_Shutdown();
00077             ImGui::DestroyContext();
00078         }
00079     }
```

Referenced by Vesper::OpenGLImGuiLayer::OnDetach().

9.21.2.5 OnEvent()

```
void Vesper::ImGuiLayer::OnEvent (
            Event & e)  [override], [virtual]
```

Reimplemented from Vesper::Layer.

Reimplemented in Vesper::OpenGLImGuiLayer.

```
00087     {
00088         if (m_BlockEvents) {
00089             ImGuiIO& io = ImGui::GetIO();
00090             e.Handled |= e.IsInCategory(EventCategoryMouse) & io.WantCaptureMouse;
00091             e.Handled |= e.IsInCategory(EventCategoryKeyboard) & io.WantCaptureKeyboard;
00092         }
00093     }
```

References Vesper::EventCategoryKeyboard, Vesper::EventCategoryMouse, Vesper::Event::Handled, Vesper::Event::IsInCategory(), and m_BlockEvents.

Referenced by Vesper::OpenGLImGuiLayer::OnEvent().

9.21.2.6 OnImGuiRender()

```
void Vesper::ImGuiLayer::OnImGuiRender ()  [override], [virtual]
```

Reimplemented from Vesper::Layer.

Reimplemented in Vesper::OpenGLImGuiLayer.

```
00082     {
00083 
00084     }
```

Referenced by Vesper::OpenGLImGuiLayer::OnImGuiRender().

9.21.2.7 SetBlockEvents()

```
virtual void Vesper::ImGuiLayer::SetBlockEvents (
            bool block)  [inline], [virtual]
```

Reimplemented in Vesper::OpenGLImGuiLayer.

```
00026 { m_BlockEvents = block; }
```

References m_BlockEvents.

Referenced by Vesper::EditorLayer::OnImGuiRender().

9.21.2.8 SetDarkThemeColors()

```
void Vesper::ImGuiLayer::SetDarkThemeColors ()  [virtual]
```

Reimplemented in Vesper::OpenGLImGuiLayer.

```
00121     {
00122         auto& colors = ImGui::GetStyle().Colors;
00123         colors[ImGuiCol_WindowBg] = ImVec4{ 0.1f, 0.105f, 0.11f, 1.0f };
00124 
00125         // Header
00126         colors[ImGuiCol_Header] = ImVec4{ 0.2f, 0.205f, 0.21f, 1.0f };
00127         colors[ImGuiCol_HeaderHovered] = ImVec4{ 0.3f, 0.305f, 0.31f, 1.0f };
00128         colors[ImGuiCol_HeaderActive] = ImVec4{ 0.15f, 0.1505f, 0.151f, 1.0f };
00129 
00130         // Buttons
00131         colors[ImGuiCol_Button] = ImVec4{ 0.2f, 0.205f, 0.21f, 1.0f };
00132         colors[ImGuiCol_ButtonHovered] = ImVec4{ 0.3f, 0.305f, 0.31f, 1.0f };
00133         colors[ImGuiCol_ButtonActive] = ImVec4{ 0.15f, 0.1505f, 0.151f, 1.0f };
00134 
00135         // Frame BG
00136         colors[ImGuiCol_FrameBg] = ImVec4{ 0.2f, 0.205f, 0.21f, 1.0f };
00137         colors[ImGuiCol_FrameBgHovered] = ImVec4{ 0.3f, 0.305f, 0.31f, 1.0f };
00138         colors[ImGuiCol_FrameBgActive] = ImVec4{ 0.15f, 0.1505f, 0.151f, 1.0f };
00139 
00140         // Tabs
00141         colors[ImGuiCol_Tab] = ImVec4{ 0.15f, 0.1505f, 0.151f, 1.0f };
00142         colors[ImGuiCol_TabHovered] = ImVec4{ 0.38f, 0.3805f, 0.381f, 1.0f };
00143         colors[ImGuiCol_TabActive] = ImVec4{ 0.28f, 0.2805f, 0.281f, 1.0f };
00144         colors[ImGuiCol_TabUnfocused] = ImVec4{ 0.15f, 0.1505f, 0.151f, 1.0f };
00145         colors[ImGuiCol_TabUnfocusedActive] = ImVec4{ 0.2f, 0.205f, 0.21f, 1.0f };
00146 
00147         // Title
00148         colors[ImGuiCol_TitleBg] = ImVec4{ 0.1f, 0.105f, 0.11f, 1.0f };
00149         colors[ImGuiCol_TitleBgActive] = ImVec4{ 0.1f, 0.105f, 0.11f, 1.0f };
00150         colors[ImGuiCol_TitleBgCollapsed] = ImVec4{ 0.1f, 0.105f, 0.11f, 1.0f };
00151 
00152 
00153 
00154     }
```

Referenced by OnAttach(), and Vesper::OpenGLImGuiLayer::SetDarkThemeColors().

9.21.3 Member Data Documentation

9.21.3.1 m_BlockEvents

```
bool Vesper::ImGuiLayer::m_BlockEvents = true  [protected]
```

Referenced by OnEvent(), SetBlockEvents(), and Vesper::OpenGLImGuiLayer::SetBlockEvents().

9.21.3.2 m_Time

```
float Vesper::ImGuiLayer::m_Time = 0.0f  [protected]
```

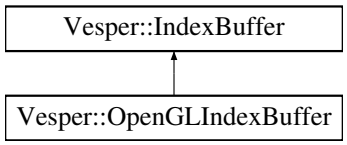
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/ImGui/ImGuiLayer.h
- Vesper/src/Vesper/ImGui/ImGuiLayer.cpp

9.22 Vesper::IndexBuffer Class Reference

`#include <Buffer.h>`

Inheritance diagram for Vesper::IndexBuffer:



Public Member Functions

- virtual ~IndexBuffer ()
- virtual void Bind () const =0
- virtual void Unbind () const =0
- virtual uint32_t GetCount () const =0

Static Public Member Functions

- static Ref< IndexBuffer > Create (uint32_t *indices, uint32_t count)

9.22.1 Constructor & Destructor Documentation

9.22.1.1 ~IndexBuffer()

```
virtual Vesper::IndexBuffer::~IndexBuffer ()  [inline], [virtual]
00124 {}
```

9.22.2 Member Function Documentation

9.22.2.1 Bind()

```
virtual void Vesper::IndexBuffer::Bind () const  [pure virtual]
```

Implemented in Vesper::OpenGLIndexBuffer.

9.22.2.2 Create()

```
Ref< IndexBuffer > Vesper::IndexBuffer::Create (
            uint32_t * indices,
            uint32_t count)  [static]
00034     {
00035
00036         switch (Renderer::GetAPI())
00037         {
00038             case RendererAPI::API::None:        VZ_CORE_ASSERT(false, "RendererAPI::None is currently
      not supported!"); return nullptr;
00039             case RendererAPI::API::OpenGL:      return CreateRef<OpenGLIndexBuffer>(indices, count);
00040         }
00041         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00042         return nullptr;
00043     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.22.2.3 GetCount()

```
virtual uint32_t Vesper::IndexBuffer::GetCount () const  [pure virtual]
```

Implemented in Vesper::OpenGLIndexBuffer.

9.22.2.4 Unbind()

```
virtual void Vesper::IndexBuffer::Unbind () const  [pure virtual]
```

Implemented in Vesper::OpenGLIndexBuffer.

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

- Vesper/src/Vesper/Renderer/Buffer.h
- Vesper/src/Vesper/Renderer/Buffer.cpp

9.23 Vesper::Input Class Reference

```
#include <Input.h>
```

Public Member Functions

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

Static Public Member Functions

- static bool IsKeyPressed (int keycode)
- static bool IsMouseButtonPressed (int button)
- static float GetMouseX ()
- static float GetMouseY ()
- static glm::vec2 GetMousePosition ()

Protected Member Functions

- Input ()=default

9.23.1 Constructor & Destructor Documentation

9.23.1.1 Input() [1/2]

```
Vesper::Input::Input ()  [protected], [default]
```

9.23.1.2 Input() [2/2]

```
Vesper::Input::Input (
            const Input & )  [delete]
```

9.23.2 Member Function Documentation

9.23.2.1 GetMousePosition()

```
glm::vec2 Vesper::Input::GetMousePosition ()  [static]
```

```
00024     {
00025         auto window = static_cast<GLFWwindow*>(Application::Get().GetWindow().GetNativeWindow());
00026         double xPos, yPos;
00027         glfwGetCursorPos(window, &xPos, &yPos);
00028         return { (float)xPos, (float)yPos };
00029
00030     }
```

References Vesper::Application::Get(), Vesper::Window::GetNativeWindow(), and Vesper::Application::GetWindow().

9.23.2.2 GetMouseX()

```
float Vesper::Input::GetMouseX ()  [static]
```

```
00033     {
00034         return GetMousePosition().x;
00035     }
```

9.23.2.3 GetMouseY()

```
float Vesper::Input::GetMouseY ()  [static]
```

```
00037     {
00038         return GetMousePosition().y;
00039     }
```

9.23.2.4 IsKeyPressed()

```
bool Vesper::Input::IsKeyPressed (
            int keycode)  [static]
```

```
00010     {
00011         auto window = static_cast<GLFWwindow*>(Application::Get().GetWindow().GetNativeWindow());
00012         auto state = glfwGetKey(window, keycode);
00013         return state == GLFW_PRESS || state == GLFW_REPEAT;
00014     }
```

References Vesper::Application::Get(), Vesper::Window::GetNativeWindow(), and Vesper::Application::GetWindow().

Referenced by Vesper::EditorLayer::OnImGuiRender(), Vesper::EditorLayer::OnKeyPressed(), Vesper::EditorCamera::OnUpdate(), and Vesper::OrthographicCameraController::OnUpdate().

9.23.2.5 IsMouseButtonPressed()

```
bool Vesper::Input::IsMouseButtonPressed (
            int button)  [static]
```

```
00017     {
00018         auto window = static_cast<GLFWwindow*>(Application::Get().GetWindow().GetNativeWindow());
00019         auto state = glfwGetMouseButton(window, button);
00020         return state == GLFW_PRESS;
00021     }
```

References Vesper::Application::Get(), Vesper::Window::GetNativeWindow(), and Vesper::Application::GetWindow().

Referenced by Vesper::EditorCamera::OnUpdate(), and Vesper::EditorLayer::OnUpdate().

9.23.2.6 operator=()

```
Input & Vesper::Input::operator= (
            const Input & )  [delete]
```

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

- Vesper/src/Vesper/Input/Input.h
- Vesper/src/Platform/Windows/WindowsInput.cpp

9.24 Vesper::InstrumentationTimer Class Reference

```
#include <Instrumentor.h>
```

Public Member Functions

- InstrumentationTimer (const char *name)
- ~InstrumentationTimer ()
- void Stop ()

Private Attributes

- const char * m_Name
- std::chrono::time_point< std::chrono::high_resolution_clock > m_StartTimepoint
- bool m_Stopped

9.24.1 Constructor & Destructor Documentation

9.24.1.1 InstrumentationTimer()

```
Vesper::InstrumentationTimer::InstrumentationTimer (
            const char * name)  [inline]
```

```
00143             : m_Name(name), m_Stopped(false)
00144         {
00145             m_StartTimepoint = std::chrono::high_resolution_clock::now();
00146         }
```

9.24.1.2 ~InstrumentationTimer()

```
Vesper::InstrumentationTimer::~InstrumentationTimer ()  [inline]
```

```
00149         {
00150             if (!m_Stopped)
00151                 Stop();
00152         }
```

9.24.2 Member Function Documentation

9.24.2.1 Stop()

```
void Vesper::InstrumentationTimer::Stop ()  [inline]
```

```
00155         {
00156             auto endTimepoint = std::chrono::high_resolution_clock::now();
00157
00158             long long start =
      std::chrono::time_point_cast<std::chrono::microseconds>(m_StartTimepoint).time_since_epoch().count();
00159             long long end =
      std::chrono::time_point_cast<std::chrono::microseconds>(endTimepoint).time_since_epoch().count();
00160
00161             uint32_t threadID = std::hash<std::thread::id>{}(std::this_thread::get_id());
00162             Instrumentor::Get().WriteProfile({ m_Name, start, end, threadID });
00163
00164             m_Stopped = true;
00165         }
```

9.24.3 Member Data Documentation

9.24.3.1 m_Name

```
const char* Vesper::InstrumentationTimer::m_Name  [private]
```

9.24.3.2 m_StartTimepoint

```
std::chrono::time_point<std::chrono::high_resolution_clock> Vesper::InstrumentationTimer::m_↩
StartTimepoint  [private]
```

9.24.3.3 m_Stopped

```
bool Vesper::InstrumentationTimer::m_Stopped  [private]
```

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

- Vesper/src/Vesper/Debug/Instrumentor.h

9.25 Vesper::Instrumentor Class Reference

```
#include <Instrumentor.h>
```

Public Member Functions

- Instrumentor ()
- void BeginSession (const std::string &name, const std::string &filepath="results.json")
- void EndSession ()
- void WriteProfile (const ProfileResult &result)
- void WriteHeader ()
- void WriteFooter ()
- void InternalEndSession ()

Static Public Member Functions

- static Instrumentor & Get ()

Private Attributes

- InstrumentationSession * m_CurrentSession
- std::ofstream m_OutputStream
- std::mutex m_Mutex

9.25.1 Constructor & Destructor Documentation

9.25.1.1 Instrumentor()

```
Vesper::Instrumentor::Instrumentor ()  [inline]
00048             : m_CurrentSession(nullptr)
00049         {
00050         }
```

9.25.2 Member Function Documentation

9.25.2.1 BeginSession()

```
void Vesper::Instrumentor::BeginSession (
             const std::string & name,
             const std::string & filepath = "results.json")  [inline]
00053         {
00054             std::lock_guard lock(m_Mutex);
00055             if (m_CurrentSession) {
00056                 // If there is already a current session, end it and start new one
00057                 if (Log::GetCoreLogger())
00058                 {
00059                     VZ_CORE_ERROR("Instrumentor::BeginSession('{0}') when session '{1}' already
      open.", name, m_CurrentSession->Name);
00060                 }
00061                 InternalEndSession();
00062             }
00063             m_OutputStream.open(filepath);
00064             if (m_OutputStream.is_open()) {
00065                 m_CurrentSession = new InstrumentationSession{ name };
00066                 WriteHeader();
00067             }
00068             else {
00069                 if (Log::GetCoreLogger())
00070                 {
00071                     VZ_CORE_ERROR("Instrumentor could not open results file '{0}'.", filepath);
00072                 }
00073             }
00074             WriteHeader();
00075             m_CurrentSession = new InstrumentationSession{ name };
00076         }
```

9.25.2.2 EndSession()

```
void Vesper::Instrumentor::EndSession ()  [inline]
```

```
00079         {
00080             std::lock_guard lock(m_Mutex);
00081             InternalEndSession();
00082         }
```

9.25.2.3 Get()

```
Instrumentor & Vesper::Instrumentor::Get ()  [inline], [static]
```

```
00133         {
00134             static Instrumentor instance;
00135             return instance;
00136         }
```

9.25.2.4 InternalEndSession()

```
void Vesper::Instrumentor::InternalEndSession ()  [inline]
```

```
00122         {
00123             if (m_CurrentSession)
00124             {
00125                 WriteFooter();
00126                 m_OutputStream.close();
00127                 delete m_CurrentSession;
00128                 m_CurrentSession = nullptr;
00129             }
00130         }
```

9.25.2.5 WriteFooter()

```
void Vesper::Instrumentor::WriteFooter ()  [inline]
```

```
00116         {
00117             m_OutputStream << "]}";
00118             m_OutputStream.flush();
00119         }
```

9.25.2.6 WriteHeader()

```
void Vesper::Instrumentor::WriteHeader ()  [inline]
```

```
00110         {
00111             m_OutputStream << "{\"otherData\": {},\"traceEvents\":[";
00112             m_OutputStream.flush();
00113         }
```

9.25.2.7 WriteProfile()

```
void Vesper::Instrumentor::WriteProfile (
             const ProfileResult & result)  [inline]
```

```
00085         {
00086             m_OutputStream << ",";
00087
00088             std::string name = result.Name;
00089             std::replace(name.begin(), name.end(), '"', '\');
00090
00091             m_OutputStream << "{";
00092             m_OutputStream << "\"cat\":\"function\",";
00093             m_OutputStream << "\"dur\":" << (result.End - result.Start) << ',';
00094             m_OutputStream << "\"name\":\"" << name << "\",";
```

```
00095             m_OutputStream « "\"ph\":\"X\",";
00096             m_OutputStream « "\"pid\":0,";
00097             m_OutputStream « "\"tid\":" « result.ThreadID « ",";
00098             m_OutputStream « "\"ts\":" « result.Start;
00099             m_OutputStream « "}";
00100
00101             std::lock_guard lock(m_Mutex);
00102             if (m_CurrentSession)
00103             {
00104                 //m_OutputStream « json.str();
00105                 m_OutputStream.flush();
00106             }
00107         }
```

9.25.3 Member Data Documentation

9.25.3.1 m_CurrentSession

```
InstrumentationSession* Vesper::Instrumentor::m_CurrentSession  [private]
```

9.25.3.2 m_Mutex

```
std::mutex Vesper::Instrumentor::m_Mutex  [private]
```

9.25.3.3 m_OutputStream

```
std::ofstream Vesper::Instrumentor::m_OutputStream  [private]
```

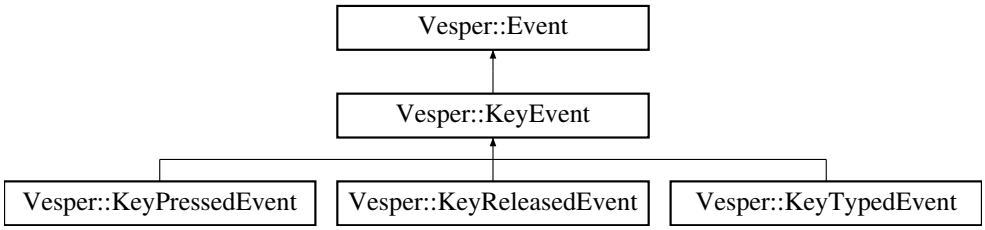
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Debug/Instrumentor.h

9.26 Vesper::KeyEvent Class Reference

```
#include <KeyEvent.h>
```

Inheritance diagram for Vesper::KeyEvent:



Public Member Functions

- int GetKeyCode () const

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- virtual std::string ToString () const
- bool IsInCategory (EventCategory category)

Protected Member Functions

- KeyEvent (int keycode)

Protected Attributes

- int m_KeyCode

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.26.1 Constructor & Destructor Documentation

9.26.1.1 KeyEvent()

```
Vesper::KeyEvent::KeyEvent (
            int keycode)  [inline], [protected]
00016           : m_KeyCode(keycode) {}
```

References m_KeyCode.

Referenced by Vesper::KeyPressedEvent::KeyPressedEvent(), Vesper::KeyReleasedEvent::KeyReleasedEvent(), and Vesper::KeyTypedEvent::KeyTypedEvent().

9.26.2 Member Function Documentation

9.26.2.1 GetKeyCode()

```
int Vesper::KeyEvent::GetKeyCode () const  [inline]
00012 { return m_KeyCode; }
```

References m_KeyCode.

Referenced by Vesper::EditorLayer::OnKeyPressed().

9.26.3 Member Data Documentation

9.26.3.1 m_KeyCode

```
int Vesper::KeyEvent::m_KeyCode  [protected]
```

Referenced by GetKeyCode(), KeyEvent(), Vesper::KeyPressedEvent::ToString(), Vesper::KeyReleasedEvent::ToString(), and Vesper::KeyTypedEvent::ToString().

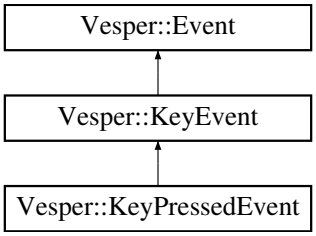
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/KeyEvent.h

9.27 Vesper::KeyPressedEvent Class Reference

```
#include <KeyEvent.h>
```

Inheritance diagram for Vesper::KeyPressedEvent:



Public Member Functions

- KeyPressedEvent (int keycode, int repeatCount)
- int GetRepeatCount () const
- std::string ToString () const override

Public Member Functions inherited from Vesper::KeyEvent

- int GetKeyCode () const

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- bool IsInCategory (EventCategory category)

Private Attributes

- int m_RepeatCount

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

Protected Member Functions inherited from Vesper::KeyEvent

- KeyEvent (int keycode)

Protected Attributes inherited from Vesper::KeyEvent

- int m_KeyCode

9.27.1 Constructor & Destructor Documentation

9.27.1.1 KeyPressedEvent()

```
Vesper::KeyPressedEvent::KeyPressedEvent (
            int keycode,
            int repeatCount)  [inline]
00024           : KeyEvent(keycode), m_RepeatCount(repeatCount) {}
```

References Vesper::KeyEvent::KeyEvent(), and m_RepeatCount.

9.27.2 Member Function Documentation

9.27.2.1 GetRepeatCount()

```
int Vesper::KeyPressedEvent::GetRepeatCount () const  [inline]
00026 { return m_RepeatCount; }
```

References m_RepeatCount.

Referenced by Vesper::EditorLayer::OnKeyPressed().

9.27.2.2 ToString()

```
std::string Vesper::KeyPressedEvent::ToString () const  [inline], [override], [virtual]
```

Reimplemented from Vesper::Event.

```
00029         {
00030             std::stringstream ss;
00031             ss << "KeyPressedEvent: " << m_KeyCode << " (" << m_RepeatCount << " repeats)";
00032             return ss.str();
00033         }
```

References Vesper::KeyEvent::m_KeyCode, and m_RepeatCount.

9.27.3 Member Data Documentation

9.27.3.1 m_RepeatCount

```
int Vesper::KeyPressedEvent::m_RepeatCount  [private]
```

Referenced by GetRepeatCount(), KeyPressedEvent(), and ToString().

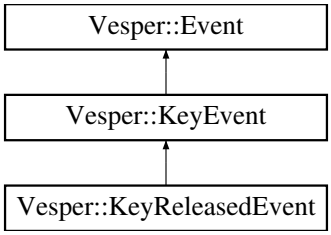
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/KeyEvent.h

9.28 Vesper::KeyReleasedEvent Class Reference

```
#include <KeyEvent.h>
```

Inheritance diagram for Vesper::KeyReleasedEvent:



Public Member Functions

- KeyReleasedEvent (int keycode)
- std::string ToString () const override

Public Member Functions inherited from Vesper::KeyEvent

- int GetKeyCode () const

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- bool IsInCategory (EventCategory category)

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

Protected Member Functions inherited from Vesper::KeyEvent

- KeyEvent (int keycode)

Protected Attributes inherited from Vesper::KeyEvent

- int m_KeyCode

9.28.1 Constructor & Destructor Documentation

9.28.1.1 KeyReleasedEvent()

```
Vesper::KeyReleasedEvent::KeyReleasedEvent (
            int keycode)  [inline]
00044           : KeyEvent(keycode) {}
```

References Vesper::KeyEvent::KeyEvent().

9.28.2 Member Function Documentation

9.28.2.1 ToString()

```
std::string Vesper::KeyReleasedEvent::ToString () const  [inline], [override], [virtual]
```

Reimplemented from Vesper::Event.

```
00047         {
00048             std::stringstream ss;
00049             ss « "KeyReleasedEvent: " « m_KeyCode;
00050             return ss.str();
00051         }
```

References Vesper::KeyEvent::m_KeyCode.

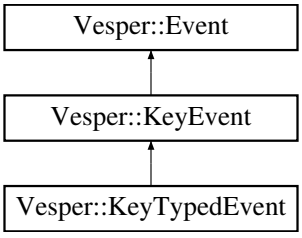
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/KeyEvent.h

9.29 Vesper::KeyTypedEvent Class Reference

```
#include <KeyEvent.h>
```

Inheritance diagram for Vesper::KeyTypedEvent:



Public Member Functions

- KeyTypedEvent (int keycode)
- std::string ToString () const override

Public Member Functions inherited from Vesper::KeyEvent

- int GetKeyCode () const

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- bool IsInCategory (EventCategory category)

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

Protected Member Functions inherited from Vesper::KeyEvent

- KeyEvent (int keycode)

Protected Attributes inherited from Vesper::KeyEvent

- int m_KeyCode

9.29.1 Constructor & Destructor Documentation

9.29.1.1 KeyTypedEvent()

```
Vesper::KeyTypedEvent::KeyTypedEvent (
            int keycode)  [inline]
00060             : KeyEvent(keycode) {
00061         }
```

References Vesper::KeyEvent::KeyEvent().

9.29.2 Member Function Documentation

9.29.2.1 ToString()

```
std::string Vesper::KeyTypedEvent::ToString () const  [inline], [override], [virtual]
```

Reimplemented from Vesper::Event.

```
00065         {
00066             std::stringstream ss;
00067             ss « "KeyTypedEvent: " « m_KeyCode;
00068             return ss.str();
00069         }
```

References Vesper::KeyEvent::m_KeyCode.

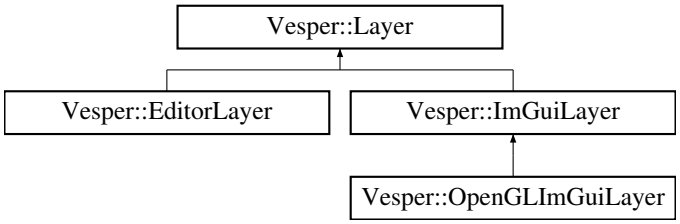
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/KeyEvent.h

9.30 Vesper::Layer Class Reference

```
#include <Layer.h>
```

Inheritance diagram for Vesper::Layer:



Public Member Functions

- Layer (const std::string &name="Layer")
- virtual ~Layer ()
- virtual void OnAttach ()
- virtual void OnDetach ()
- virtual void OnUpdate (Timestep ts)
- virtual void OnEvent (Event &event)
- virtual void OnRender ()
- virtual void OnImGuiRender ()
- const std::string & GetName () const

Protected Attributes

- std::string m_DebugName

9.30.1 Constructor & Destructor Documentation

9.30.1.1 Layer()

```
Vesper::Layer::Layer (
            const std::string & name = "Layer")
00007         : m_DebugName(name)
00008     {
00009     }
```

References Layer().

Referenced by Layer().

9.30.1.2 ~Layer()

```
Vesper::Layer::~Layer ()  [virtual]
00012     {
00013     }
```

9.30.2 Member Function Documentation

9.30.2.1 GetName()

```
const std::string & Vesper::Layer::GetName () const  [inline]
00021 { return m_DebugName; }
```

9.30.2.2 OnAttach()

```
virtual void Vesper::Layer::OnAttach ()  [inline], [virtual]
```

Reimplemented in Vesper::EditorLayer, Vesper::ImGuiLayer, and Vesper::OpenGLImGuiLayer.

```
00014 {};
```

Referenced by Vesper::Application::PushLayer(), and Vesper::Application::PushOverlay().

9.30.2.3 OnDetach()

```
virtual void Vesper::Layer::OnDetach ()  [inline], [virtual]
```

Reimplemented in Vesper::EditorLayer, Vesper::ImGuiLayer, and Vesper::OpenGLImGuiLayer.

```
00015 {};
```

9.30.2.4 OnEvent()

```
virtual void Vesper::Layer::OnEvent (
            Event & event)  [inline], [virtual]
```

Reimplemented in Vesper::EditorLayer, Vesper::ImGuiLayer, and Vesper::OpenGLImGuiLayer.

```
00017 {}
```

9.30.2.5 OnImGuiRender()

```
virtual void Vesper::Layer::OnImGuiRender ()  [inline], [virtual]
```

Reimplemented in Vesper::EditorLayer, Vesper::ImGuiLayer, and Vesper::OpenGLImGuiLayer.

```
00019 {};
```

9.30.2.6 OnRender()

```
virtual void Vesper::Layer::OnRender ()  [inline], [virtual]
00018 {};
```

9.30.2.7 OnUpdate()

```
virtual void Vesper::Layer::OnUpdate (
            Timestep ts)  [inline], [virtual]
```

Reimplemented in Vesper::EditorLayer.

```
00016 {};
```

9.30.3 Member Data Documentation

9.30.3.1 m_DebugName

```
std::string Vesper::Layer::m_DebugName  [protected]
```

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

- Vesper/src/Vesper/App/Layer.h
- Vesper/src/Vesper/App/Layer.cpp

9.31 Vesper::LayerStack Class Reference

```
#include <LayerStack.h>
```

Public Member Functions

- LayerStack ()
- ~LayerStack ()
- void PushLayer (Layer *layer)
- void PushOverlay (Layer *overlay)
- void PopLayer (Layer *layer)
- void PopOverlay (Layer *overlay)
- std::vector< Layer * >::iterator begin ()
- std::vector< Layer * >::iterator end ()
- std::vector< Layer * >::reverse_iterator rbegin ()
- std::vector< Layer * >::reverse_iterator rend ()

Private Attributes

- std::vector< Layer * > m_Layers
- unsigned int m_LayerInsertIndex = 0

9.31.1 Constructor & Destructor Documentation

9.31.1.1 LayerStack()

```
Vesper::LayerStack::LayerStack ()
00007     {
00008     }
```

9.31.1.2 ~LayerStack()

```
Vesper::LayerStack::~LayerStack ()
00011     {
00012         for (Layer* layer : m_Layers)
00013         {
00014             layer->OnDetach();
00015             delete layer;
00016         }
00017     }
```

9.31.2 Member Function Documentation

9.31.2.1 begin()

```
std::vector< Layer * >::iterator Vesper::LayerStack::begin ()  [inline]
00020 { return m_Layers.begin(); }
```

9.31.2.2 end()

```
std::vector< Layer * >::iterator Vesper::LayerStack::end ()  [inline]
00021 { return m_Layers.end(); }
```

9.31.2.3 PopLayer()

```
void Vesper::LayerStack::PopLayer (
             Layer * layer)
00033     {
00034         VZ_PROFILE_FUNCTION();
00035         auto it = std::find(m_Layers.begin(), m_Layers.end(), layer);
00036         if (it != m_Layers.end())
00037         {
00038             m_Layers.erase(it);
00039             m_LayerInsertIndex--;
00040         }
00041     }
```

References m_LayerInsertIndex.

9.31.2.4 PopOverlay()

```
void Vesper::LayerStack::PopOverlay (
            Layer * overlay)
```

```
00043     {
00044         VZ_PROFILE_FUNCTION();
00045         auto it = std::find(m_Layers.begin(), m_Layers.end(), overlay);
00046         if (it != m_Layers.end())
00047         {
00048             m_Layers.erase(it);
00049         }
00050     }
```

9.31.2.5 PushLayer()

```
void Vesper::LayerStack::PushLayer (
            Layer * layer)
```

```
00020     {
00021         VZ_PROFILE_FUNCTION();
00022         m_Layers.emplace(m_Layers.begin() + m_LayerInsertIndex, layer);
00023         m_LayerInsertIndex++;
00024     }
```

References m_LayerInsertIndex.

9.31.2.6 PushOverlay()

```
void Vesper::LayerStack::PushOverlay (
            Layer * overlay)
```

```
00027     {
00028         VZ_PROFILE_FUNCTION();
00029         m_Layers.emplace_back(overlay);
00030     }
```

9.31.2.7 rbegin()

```
std::vector< Layer * >::reverse_iterator Vesper::LayerStack::rbegin ()  [inline]
```

```
00022 { return m_Layers.rbegin(); }
```

9.31.2.8 rend()

```
std::vector< Layer * >::reverse_iterator Vesper::LayerStack::rend ()  [inline]
```

```
00023 { return m_Layers.rend(); }
```

9.31.3 Member Data Documentation

9.31.3.1 m_LayerInsertIndex

```
unsigned int Vesper::LayerStack::m_LayerInsertIndex = 0  [private]
```

Referenced by PopLayer(), and PushLayer().

9.31.3.2 m_Layers

```
std::vector<Layer*> Vesper::LayerStack::m_Layers  [private]
```

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

- Vesper/src/Vesper/App/LayerStack.h
- Vesper/src/Vesper/App/LayerStack.cpp

9.32 Vesper::Log Class Reference

```
#include <Log.h>
```

Static Public Member Functions

- static void Init ()
- static std::shared_ptr< spdlog::logger > & GetCoreLogger ()
- static std::shared_ptr< spdlog::logger > & GetClientLogger ()

Static Private Attributes

- static std::shared_ptr< spdlog::logger > s_CoreLogger
- static std::shared_ptr< spdlog::logger > s_ClientLogger

9.32.1 Member Function Documentation

9.32.1.1 GetClientLogger()

```
std::shared_ptr< spdlog::logger > & Vesper::Log::GetClientLogger ()  [inline], [static]
00014 { return s_ClientLogger; }
```

9.32.1.2 GetCoreLogger()

```
std::shared_ptr< spdlog::logger > & Vesper::Log::GetCoreLogger ()  [inline], [static]
00013 { return s_CoreLogger; }
```

9.32.1.3 Init()

```
void Vesper::Log::Init ()  [static]
00012     {
00013         spdlog::set_pattern("%^[%T] %n: %v%$");
00014         s_CoreLogger = spdlog::stdout_color_mt("VESPER");
00015         s_CoreLogger->set_level(spdlog::level::trace);
00016
00017         s_ClientLogger = spdlog::stdout_color_mt("APP");
00018         s_ClientLogger->set_level(spdlog::level::trace);
00019     }
```

9.32.2 Member Data Documentation

9.32.2.1 s_ClientLogger

```
std::shared_ptr< spdlog::logger > Vesper::Log::s_ClientLogger  [static], [private]
```

9.32.2.2 s_CoreLogger

```
std::shared_ptr< spdlog::logger > Vesper::Log::s_CoreLogger  [static], [private]
```

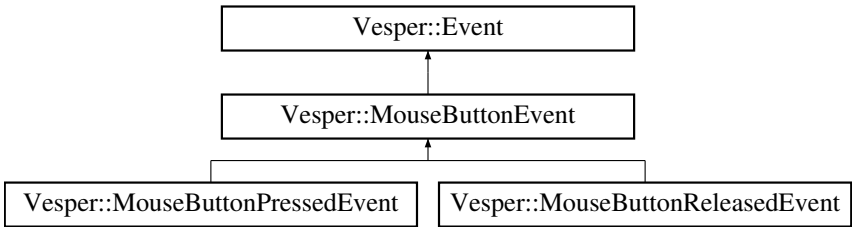
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/Core/Log.h
- Vesper/src/Vesper/Core/Log.cpp

9.33 Vesper::MouseButtonEvent Class Reference

```
#include <MouseEvent.h>
```

Inheritance diagram for Vesper::MouseButtonEvent:



Public Member Functions

- int GetMouseButton () const

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- virtual std::string ToString () const
- bool IsInCategory (EventCategory category)

Protected Member Functions

- MouseButtonEvent (int button)

Protected Attributes

- int m_Button

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.33.1 Constructor & Destructor Documentation

9.33.1.1 MouseButtonEvent()

```
Vesper::MouseButtonEvent::MouseButtonEvent (
            int button)  [inline], [protected]
00064            : m_Button(button) {
00065        }
```

References m_Button.

Referenced by Vesper::MouseButtonPressedEvent::MouseButtonPressedEvent(), and Vesper::MouseButtonReleasedEvent::MouseB

9.33.2 Member Function Documentation

9.33.2.1 GetMouseButton()

```
int Vesper::MouseButtonEvent::GetMouseButton () const  [inline]
00059 { return m_Button; }
```

References m_Button.

Referenced by Vesper::MouseButtonPressedEvent::ToString(), and Vesper::MouseButtonReleasedEvent::ToString().

9.33.3 Member Data Documentation

9.33.3.1 m_Button

```
int Vesper::MouseButtonEvent::m_Button  [protected]
```

Referenced by GetMouseButton(), and MouseButtonEvent().

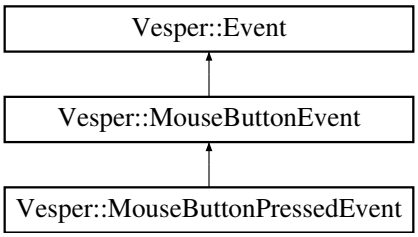
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/MouseEvent.h

9.34 Vesper::MouseButtonPressedEvent Class Reference

`#include <MouseEvent.h>`

Inheritance diagram for Vesper::MouseButtonPressedEvent:



Public Member Functions

- MouseButtonPressedEvent (int button)
- std::string ToString () const override

Public Member Functions inherited from Vesper::MouseButtonEvent

- int GetMouseButton () const

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- bool IsInCategory (EventCategory category)

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

Protected Member Functions inherited from Vesper::MouseButtonEvent

- MouseButtonEvent (int button)

Protected Attributes inherited from Vesper::MouseButtonEvent

- int m_Button

9.34.1 Constructor & Destructor Documentation

9.34.1.1 MouseButtonPressedEvent()

```
Vesper::MouseButtonPressedEvent::MouseButtonPressedEvent (
            int button)  [inline]
00073             : MouseButtonEvent(button) {
00074         }
```

References Vesper::MouseButtonEvent::MouseButtonEvent().

9.34.2 Member Function Documentation

9.34.2.1 ToString()

```
std::string Vesper::MouseButtonPressedEvent::ToString () const  [inline], [override], [virtual]
```

Reimplemented from Vesper::Event.

```
00077         {
00078             std::stringstream ss;
00079             ss « "MouseButtonPressedEvent: " « GetMouseButton();
00080             return ss.str();
00081         }
```

References Vesper::MouseButtonEvent::GetMouseButton().

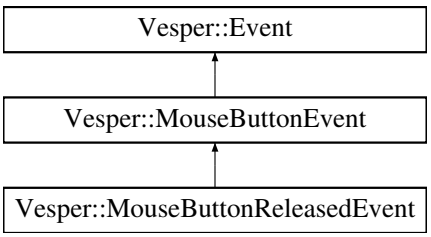
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/MouseEvent.h

9.35 Vesper::MouseButtonReleasedEvent Class Reference

```
#include <MouseEvent.h>
```

Inheritance diagram for Vesper::MouseButtonReleasedEvent:



Public Member Functions

- MouseButtonReleasedEvent (int button)
- std::string ToString () const override

Public Member Functions inherited from Vesper::MouseButtonEvent

- int GetMouseButton () const

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- bool IsInCategory (EventCategory category)

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

Protected Member Functions inherited from Vesper::MouseButtonEvent

- MouseButtonEvent (int button)

Protected Attributes inherited from Vesper::MouseButtonEvent

- int m_Button

9.35.1 Constructor & Destructor Documentation

9.35.1.1 MouseButtonReleasedEvent()

```
Vesper::MouseButtonReleasedEvent::MouseButtonReleasedEvent (
            int button)  [inline]
00091             : MouseButtonEvent(button) {
00092         }
```

References Vesper::MouseButtonEvent::MouseButtonEvent().

9.35.2 Member Function Documentation

9.35.2.1 ToString()

```
std::string Vesper::MouseButtonReleasedEvent::ToString () const  [inline], [override], [virtual]
```

Reimplemented from Vesper::Event.

```
00095         {
00096             std::stringstream ss;
00097             ss << "MouseButtonReleasedEvent: " << GetMouseButton();
00098             return ss.str();
00099         }
```

References Vesper::MouseButtonEvent::GetMouseButton().

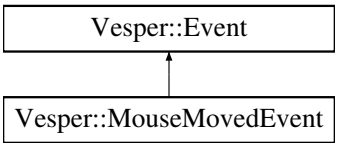
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/MouseEvent.h

9.36 Vesper::MouseMovedEvent Class Reference

`#include <MouseEvent.h>`

Inheritance diagram for Vesper::MouseMovedEvent:



Public Member Functions

- MouseMovedEvent (float x, float y)
- float GetX () const
- float GetY () const
- std::string ToString () const override

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- bool IsInCategory (EventCategory category)

Private Attributes

- float m_MouseX
- float m_MouseY

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.36.1 Constructor & Destructor Documentation

9.36.1.1 MouseMovedEvent()

```
Vesper::MouseMovedEvent::MouseMovedEvent (
            float x,
            float y)  [inline]
00014            : m_MouseX(x), m_MouseY(y) {
00015        }
```

References m_MouseX, and m_MouseY.

9.36.2 Member Function Documentation

9.36.2.1 GetX()

```
float Vesper::MouseMovedEvent::GetX () const  [inline]
00017 { return m_MouseX; }
```

References m_MouseX.

9.36.2.2 GetY()

```
float Vesper::MouseMovedEvent::GetY () const  [inline]
00018 { return m_MouseY; }
```

References m_MouseY.

9.36.2.3 ToString()

```
std::string Vesper::MouseMovedEvent::ToString () const  [inline], [override], [virtual]
```

Reimplemented from Vesper::Event.

```
00021         {
00022             std::stringstream ss;
00023             ss « "MouseMovedEvent: " « m_MouseX « ", " « m_MouseY;
00024             return ss.str();
00025         }
```

References m_MouseX, and m_MouseY.

9.36.3 Member Data Documentation

9.36.3.1 m_MouseX

```
float Vesper::MouseMovedEvent::m_MouseX  [private]
```

Referenced by GetX(), MouseMovedEvent(), and ToString().

9.36.3.2 m_MouseY

```
float Vesper::MouseMovedEvent::m_MouseY  [private]
```

Referenced by GetY(), MouseMovedEvent(), and ToString().

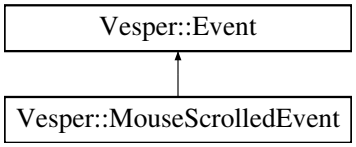
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/MouseEvent.h

9.37 Vesper::MouseScrolledEvent Class Reference

`#include <MouseEvent.h>`

Inheritance diagram for Vesper::MouseScrolledEvent:



Public Member Functions

- MouseScrolledEvent (float xOffset, float yOffset)
- float GetXOffset () const
- float GetYOffset () const
- std::string ToString () const override

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- bool IsInCategory (EventCategory category)

Private Attributes

- float m_XOffset
- float m_YOffset

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.37.1 Constructor & Destructor Documentation

9.37.1.1 MouseScrolledEvent()

```
Vesper::MouseScrolledEvent::MouseScrolledEvent (
            float xOffset,
            float yOffset)  [inline]
00037             : m_XOffset(xOffset), m_YOffset(yOffset) {
00038         }
```

References m_XOffset, and m_YOffset.

9.37.2 Member Function Documentation

9.37.2.1 GetXOffset()

```
float Vesper::MouseScrolledEvent::GetXOffset () const  [inline]
00040 { return m_XOffset; }
```

References m_XOffset.

Referenced by ToString().

9.37.2.2 GetYOffset()

```
float Vesper::MouseScrolledEvent::GetYOffset () const  [inline]
00041 { return m_YOffset; }
```

References m_YOffset.

Referenced by Vesper::EditorCamera::OnMouseScroll(), Vesper::OrthographicCameraController::OnMouseScrolled(), and ToString().

9.37.2.3 ToString()

```
std::string Vesper::MouseScrolledEvent::ToString () const  [inline], [override], [virtual]
```

Reimplemented from Vesper::Event.

```
00044         {
00045             std::stringstream ss;
00046             ss « "MouseScrolledEvent: " « GetXOffset() « ", " « GetYOffset();
00047             return ss.str();
00048         }
```

References GetXOffset(), and GetYOffset().

9.37.3 Member Data Documentation

9.37.3.1 m_XOffset

```
float Vesper::MouseScrolledEvent::m_XOffset  [private]
```

Referenced by GetXOffset(), and MouseScrolledEvent().

9.37.3.2 m_YOffset

```
float Vesper::MouseScrolledEvent::m_YOffset  [private]
```

Referenced by GetYOffset(), and MouseScrolledEvent().

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

- Vesper/src/Vesper/Events/MouseEvent.h

9.38 Vesper::NameComponent Struct Reference

`#include <Components.h>`

Public Member Functions

- NameComponent ()=default
- NameComponent (const NameComponent &)=default
- NameComponent (const std::string &name)
- operator std::string & ()
- operator const std::string & () const
- std::string & GetName ()

Public Attributes

- std::string Name

9.38.1 Constructor & Destructor Documentation

9.38.1.1 NameComponent() [1/3]

```
Vesper::NameComponent::NameComponent ()  [default]
```

9.38.1.2 NameComponent() [2/3]

```
Vesper::NameComponent::NameComponent (
            const NameComponent & )  [default]
```

9.38.1.3 NameComponent() [3/3]

```
Vesper::NameComponent::NameComponent (
            const std::string & name)  [inline]
00042            : Name(name) {
00043        }
```

9.38.2 Member Function Documentation

9.38.2.1 GetName()

```
std::string & Vesper::NameComponent::GetName ()  [inline]
00046 { return Name; }
```

9.38.2.2 operator const std::string &()

```
Vesper::NameComponent::operator const std::string & () const  [inline]
00045 { return Name; }
```

9.38.2.3 operator std::string &()

```
Vesper::NameComponent::operator std::string & ()  [inline]
00044 { return Name; }
```

9.38.3 Member Data Documentation

9.38.3.1 Name

```
std::string Vesper::NameComponent::Name
```

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

- Vesper/src/Vesper/Scene/Components.h

9.39 Vesper::NativeScriptComponent Struct Reference

```
#include <Components.h>
```

Public Member Functions

- template<typename T>
  void Bind ()

Public Attributes

- ScriptableEntity * Instance = nullptr
- ScriptableEntity *(* InstantiateScript)()
- void(* DestroyScript)(NativeScriptComponent *)

9.39.1 Member Function Documentation

9.39.1.1 Bind()

```
template<typename T>
void Vesper::NativeScriptComponent::Bind ()  [inline]
00178         {
00179             InstantiateScript = []() { return static_cast<ScriptableEntity*> (new T()); };
00180             DestroyScript = [](NativeScriptComponent* nsc) { delete nsc->Instance; nsc->Instance =
      nullptr; };
00181         }
```

9.39.2 Member Data Documentation

9.39.2.1 DestroyScript

```
void(* Vesper::NativeScriptComponent::DestroyScript) (NativeScriptComponent *)
```

9.39.2.2 Instance

```
ScriptableEntity* Vesper::NativeScriptComponent::Instance = nullptr
```

9.39.2.3 InstantiateScript

```
ScriptableEntity *(* Vesper::NativeScriptComponent::InstantiateScript) ()
```

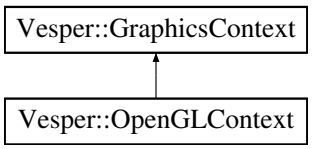
The documentation for this struct was generated from the following file:

- Vesper/src/Vesper/Scene/Components.h

9.40 Vesper::OpenGLContext Class Reference

```
#include <OpenGLContext.h>
```

Inheritance diagram for Vesper::OpenGLContext:



Public Member Functions

- OpenGLContext (GLFWwindow *windowHandle)
- virtual ~OpenGLContext ()
- void Init () override
- void SwapBuffers () override

Public Member Functions inherited from Vesper::GraphicsContext

- virtual ~GraphicsContext ()

Private Attributes

- GLFWwindow * m_WindowHandle

9.40.1 Constructor & Destructor Documentation

9.40.1.1 OpenGLContext()

```
Vesper::OpenGLContext::OpenGLContext (
            GLFWwindow * windowHandle)
```

```
00012         : m_WindowHandle(windowHandle)
00013     {
00014         VZ_CORE_ASSERT(windowHandle, "Window handle is null!");
00015
00016     }
```

References m_WindowHandle.

9.40.1.2 ~OpenGLContext()

```
Vesper::OpenGLContext::~OpenGLContext ()  [virtual]
```

```
00019     {
00020     }
```

9.40.2 Member Function Documentation

9.40.2.1 Init()

```
void Vesper::OpenGLContext::Init ()  [override], [virtual]
```

Implements Vesper::GraphicsContext.

```
00023     {
00024         VZ_PROFILE_FUNCTION();
00025
00026         glfwMakeContextCurrent(m_WindowHandle);
00027         int status = gladLoadGLLoader((GLADloadproc)glfwGetProcAddress);
00028         VZ_CORE_ASSERT(status, "Failed to initialize Glad!");
00029
00030         VZ_CORE_INFO("OpenGL Info:");
00031         VZ_CORE_INFO("  Vendor: {0}", (const char *)glGetString(GL_VENDOR));
00032         VZ_CORE_INFO("  Renderer: {0}", (const char *)glGetString(GL_RENDERER));
00033         VZ_CORE_INFO("  Version: {0}", (const char *)glGetString(GL_VERSION));
00034
00035 #ifdef VZ_ENABLE_ASSERTS
00036         int major = 0, minor = 0;
00037         glGetIntegerv(GL_MAJOR_VERSION, &major);
00038         glGetIntegerv(GL_MINOR_VERSION, &minor);
00039         VZ_CORE_ASSERT(major > 4 || (major == 4 && minor >= 5), "Vesper requires at least OpenGL
      version 4.5!");
00040 #endif
00041
00042     }
```

References m_WindowHandle.

9.40.2.2 SwapBuffers()

```
void Vesper::OpenGLContext::SwapBuffers ()  [override], [virtual]
```

Implements Vesper::GraphicsContext.

```
00045     {
00046         VZ_PROFILE_FUNCTION();
00047         glfwSwapBuffers(m_WindowHandle);
00048
00049     }
```

References m_WindowHandle.

9.40.3 Member Data Documentation

9.40.3.1 m_WindowHandle

```
GLFWwindow* Vesper::OpenGLContext::m_WindowHandle  [private]
```

Referenced by Init(), OpenGLContext(), and SwapBuffers().

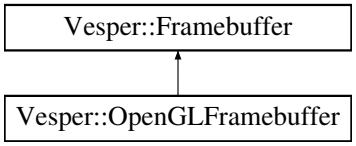
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLContext.h
- Vesper/src/RenderAPI/OpenGL/OpenGLContext.cpp

9.41 Vesper::OpenGLFramebuffer Class Reference

```
#include <OpenGLFramebuffer.h>
```

Inheritance diagram for Vesper::OpenGLFramebuffer:



Public Member Functions

- OpenGLFramebuffer (const FramebufferSpecification &spec)
- virtual ~OpenGLFramebuffer ()
- void Invalidate ()
- virtual void Bind () override
- virtual void Unbind () override
- virtual void Resize (uint32_t width, uint32_t height) override
- virtual uint32_t GetColorAttachmentRendererID () const override
- virtual const FramebufferSpecification & GetSpecification () const

Public Member Functions inherited from Vesper::Framebuffer

- ~Framebuffer ()=default

Private Attributes

- uint32_t m_RendererID
- uint32_t m_ColorAttachment
- uint32_t m_DepthAttachment
- FramebufferSpecification m_Specification

Additional Inherited Members

Static Public Member Functions inherited from Vesper::Framebuffer

- static Ref< Framebuffer > Create (const FramebufferSpecification &spec)

9.41.1 Constructor & Destructor Documentation

9.41.1.1 OpenGLFramebuffer()

```
Vesper::OpenGLFramebuffer::OpenGLFramebuffer (
            const FramebufferSpecification & spec)
00012         : m_Specification(spec)
00013     {
00014         Invalidate();
00015     }
```

References Invalidate(), and m_Specification.

9.41.1.2 ~OpenGLFramebuffer()

```
Vesper::OpenGLFramebuffer::~OpenGLFramebuffer ()  [virtual]
00018     {
00019         glDeleteFramebuffers(1, &m_RendererID);
00020         glDeleteTextures(1, &m_ColorAttachment);
00021         glDeleteTextures(1, &m_DepthAttachment);
00022     }
```

References m_ColorAttachment, m_DepthAttachment, and m_RendererID.

9.41.2 Member Function Documentation

9.41.2.1 Bind()

```
void Vesper::OpenGLFramebuffer::Bind ()  [override], [virtual]
```

Implements Vesper::Framebuffer.

```
00057     {
00058         glBindFramebuffer(GL_FRAMEBUFFER, m_RendererID);
00059         glViewport(0, 0, m_Specification.Width, m_Specification.Height);
00060     }
```

References Vesper::FramebufferSpecification::Height, m_Specification, and Vesper::FramebufferSpecification::Width.

9.41.2.2 GetColorAttachmentRendererID()

```
virtual uint32_t Vesper::OpenGLFramebuffer::GetColorAttachmentRendererID () const  [inline],
[override], [virtual]
```

Implements Vesper::Framebuffer.

```
00019 { return m_ColorAttachment; }
```

References m_ColorAttachment.

9.41.2.3 GetSpecification()

```
virtual const FramebufferSpecification & Vesper::OpenGLFramebuffer::GetSpecification () const
[inline], [virtual]
```

Implements Vesper::Framebuffer.

```
00020 { return m_Specification; }
```

References m_Specification.

9.41.2.4 Invalidate()

```
void Vesper::OpenGLFramebuffer::Invalidate ()
```

```
00025     {
00026         if (m_RendererID)
00027         {
00028             glDeleteFramebuffers(1, &m_RendererID);
00029             glDeleteTextures(1, &m_ColorAttachment);
00030             glDeleteTextures(1, &m_DepthAttachment);
00031         }
00032 
00033         glCreateFramebuffers(1, &m_RendererID);
00034         glBindFramebuffer(GL_FRAMEBUFFER, m_RendererID);
00035 
00036         glCreateTextures(GL_TEXTURE_2D, 1, &m_ColorAttachment);
00037         glBindTexture(GL_TEXTURE_2D, m_ColorAttachment);
00038         glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA8, m_Specification.Width, m_Specification.Height, 0,
      GL_RGBA, GL_UNSIGNED_BYTE, nullptr);
00039         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
00040         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
00041 
00042         glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, m_ColorAttachment,
      0);
00043 
00044         glCreateTextures(GL_TEXTURE_2D, 1, &m_DepthAttachment);
00045         glBindTexture(GL_TEXTURE_2D, m_DepthAttachment);
00046         glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH24_STENCIL8, m_Specification.Width,
      m_Specification.Height, 0, GL_DEPTH_STENCIL, GL_UNSIGNED_INT_24_8, nullptr);
00047 
00048         glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_STENCIL_ATTACHMENT, GL_TEXTURE_2D,
      m_DepthAttachment, 0);
00049 
00050         VZ_CORE_ASSERT(glCheckFramebufferStatus(GL_FRAMEBUFFER) == GL_FRAMEBUFFER_COMPLETE,
      "Framebuffer is complete!");
00051 
00052         glBindFramebuffer(GL_FRAMEBUFFER, 0);
00053 
00054     }
```

References m_ColorAttachment, m_DepthAttachment, and m_RendererID.

Referenced by OpenGLFramebuffer(), and Resize().

9.41.2.5 Resize()

```
void Vesper::OpenGLFramebuffer::Resize (
            uint32_t width,
            uint32_t height)  [override], [virtual]
```

Implements Vesper::Framebuffer.

```
00068     {
00069         if (width == 0 || height == 0 || width > 8192 || height > 8192)
00070         {
00071             VZ_CORE_WARN("Attempted to resize framebuffer to {0}, {1}", width, height);
00072             return;
00073         }
00074         m_Specification.Width = width;
00075         m_Specification.Height = height;
00076         Invalidate();
00077     }
```

References Vesper::FramebufferSpecification::Height, Invalidate(), m_Specification, and Vesper::FramebufferSpecification::Width.

9.41.2.6 Unbind()

```
void Vesper::OpenGLFramebuffer::Unbind ()  [override], [virtual]
```

Implements Vesper::Framebuffer.

```
00063     {
00064         glBindFramebuffer(GL_FRAMEBUFFER, 0);
00065     }
```

9.41.3 Member Data Documentation

9.41.3.1 m_ColorAttachment

```
uint32_t Vesper::OpenGLFramebuffer::m_ColorAttachment  [private]
```

Referenced by GetColorAttachmentRendererID(), Invalidate(), and ~OpenGLFramebuffer().

9.41.3.2 m_DepthAttachment

```
uint32_t Vesper::OpenGLFramebuffer::m_DepthAttachment  [private]
```

Referenced by Invalidate(), and ~OpenGLFramebuffer().

9.41.3.3 m_RendererID

```
uint32_t Vesper::OpenGLFramebuffer::m_RendererID  [private]
```

Referenced by Invalidate(), and ~OpenGLFramebuffer().

9.41.3.4 m_Specification

```
FramebufferSpecification Vesper::OpenGLFramebuffer::m_Specification  [private]
```

Referenced by Bind(), GetSpecification(), OpenGLFramebuffer(), and Resize().

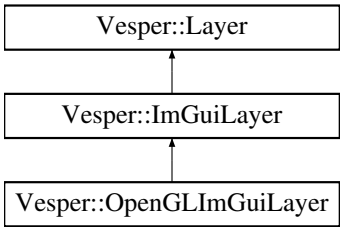
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLFramebuffer.h
- Vesper/src/RenderAPI/OpenGL/OpenGLFramebuffer.cpp

9.42 Vesper::OpenGLImGuiLayer Class Reference

```
#include <OpenGLImGuiLayer.h>
```

Inheritance diagram for Vesper::OpenGLImGuiLayer:



Public Member Functions

- OpenGLImGuiLayer ()
- ~OpenGLImGuiLayer ()
- virtual void OnAttach () override
- virtual void OnDetach () override
- virtual void OnImGuiRender () override
- virtual void OnEvent (Event &e) override
- virtual void Begin () override
- virtual void End () override
- virtual void SetBlockEvents (bool block)
- virtual void SetDarkThemeColors () override

Public Member Functions inherited from Vesper::ImGuiLayer

- ImGuiLayer ()
- ~ImGuiLayer ()

Public Member Functions inherited from Vesper::Layer

- Layer (const std::string &name="Layer")
- virtual ~Layer ()
- virtual void OnUpdate (Timestep ts)
- virtual void OnRender ()
- const std::string & GetName () const

Additional Inherited Members

Protected Attributes inherited from Vesper::ImGuiLayer

- bool m_BlockEvents = true
- float m_Time = 0.0f

Protected Attributes inherited from Vesper::Layer

- std::string m_DebugName

9.42.1 Constructor & Destructor Documentation

9.42.1.1 OpenGLImGuiLayer()

```
Vesper::OpenGLImGuiLayer::OpenGLImGuiLayer ()
00020                                       : ImGuiLayer()
00021     {
00022     }
```

References Vesper::ImGuiLayer::ImGuiLayer().

9.42.1.2 ~OpenGLImGuiLayer()

```
Vesper::OpenGLImGuiLayer::~OpenGLImGuiLayer ()
00025     {
00026     }
```

9.42.2 Member Function Documentation

9.42.2.1 Begin()

```
void Vesper::OpenGLImGuiLayer::Begin ()  [override], [virtual]
```

Reimplemented from Vesper::ImGuiLayer.

```
00059     {
00060         ImGuiLayer::Begin();
00061     }
```

References Vesper::ImGuiLayer::Begin().

9.42.2.2 End()

```
void Vesper::OpenGLImGuiLayer::End ()  [override], [virtual]
```

Reimplemented from Vesper::ImGuiLayer.

```
00064     {
00065         ImGuiLayer::End();
00066     }
```

References Vesper::ImGuiLayer::End().

9.42.2.3 OnAttach()

```
void Vesper::OpenGLImGuiLayer::OnAttach ()  [override], [virtual]
```

Reimplemented from Vesper::ImGuiLayer.

```
00029     {
00030         ImGuiLayer::OnAttach();
00031
00032
00033
00034     }
```

References Vesper::ImGuiLayer::OnAttach().

9.42.2.4 OnDetach()

```
void Vesper::OpenGLImGuiLayer::OnDetach ()  [override], [virtual]
```

Reimplemented from Vesper::ImGuiLayer.

```
00037     {
00038         ImGuiLayer::OnDetach();
00039
00040
00041
00042     }
```

References Vesper::ImGuiLayer::OnDetach().

9.42.2.5 OnEvent()

```
void Vesper::OpenGLImGuiLayer::OnEvent (
            Event & e)  [override], [virtual]
```

Reimplemented from Vesper::ImGuiLayer.

```
00052     {
00053         ImGuiLayer::OnEvent(e);
00054 
00055 
00056     }
```

References Vesper::ImGuiLayer::OnEvent().

9.42.2.6 OnImGuiRender()

```
void Vesper::OpenGLImGuiLayer::OnImGuiRender ()  [override], [virtual]
```

Reimplemented from Vesper::ImGuiLayer.

```
00045     {
00046         ImGuiLayer::OnImGuiRender();
00047 
00048 
00049     }
```

References Vesper::ImGuiLayer::OnImGuiRender().

9.42.2.7 SetBlockEvents()

```
virtual void Vesper::OpenGLImGuiLayer::SetBlockEvents (
            bool block)  [inline], [virtual]
```

Reimplemented from Vesper::ImGuiLayer.

```
00027 { m_BlockEvents = block; }
```

References Vesper::ImGuiLayer::m_BlockEvents.

9.42.2.8 SetDarkThemeColors()

```
void Vesper::OpenGLImGuiLayer::SetDarkThemeColors ()  [override], [virtual]
```

Reimplemented from Vesper::ImGuiLayer.

```
00069     {
00070         ImGuiLayer::SetDarkThemeColors();
00071     }
```

References Vesper::ImGuiLayer::SetDarkThemeColors().

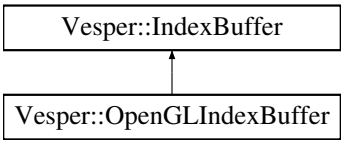
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLImGuiLayer.h
- Vesper/src/RenderAPI/OpenGL/OpenGLImGuiLayer.cpp

9.43 Vesper::OpenGLIndexBuffer Class Reference

`#include <OpenGLBuffer.h>`

Inheritance diagram for Vesper::OpenGLIndexBuffer:



Public Member Functions

- OpenGLIndexBuffer (uint32_t *indices, uint32_t count)
- virtual ~OpenGLIndexBuffer ()
- virtual void Bind () const override
- virtual void Unbind () const override
- virtual uint32_t GetCount () const override

Public Member Functions inherited from Vesper::IndexBuffer

- virtual ~IndexBuffer ()

Private Attributes

- uint32_t m_RendererID
- uint32_t m_Count

Additional Inherited Members

Static Public Member Functions inherited from Vesper::IndexBuffer

- static Ref< IndexBuffer > Create (uint32_t *indices, uint32_t count)

9.43.1 Constructor & Destructor Documentation

9.43.1.1 OpenGLIndexBuffer()

```
Vesper::OpenGLIndexBuffer::OpenGLIndexBuffer (
            uint32_t * indices,
            uint32_t count)
00063         : m_Count(count)
00064     {
00065         VZ_PROFILE_FUNCTION();
00066
00067         glCreateBuffers(1, &m_RendererID);
00068         glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_RendererID);
00069         glBufferData(GL_ELEMENT_ARRAY_BUFFER, count * sizeof(uint32_t), indices, GL_STATIC_DRAW);
00070     }
```

References m_Count, and m_RendererID.

9.43.1.2 ~OpenGLIndexBuffer()

```
Vesper::OpenGLIndexBuffer::~OpenGLIndexBuffer ()  [virtual]
```

```
00073     {
00074         VZ_PROFILE_FUNCTION();
00075
00076         glDeleteBuffers(1, &m_RendererID);
00077     }
```

References m_RendererID.

9.43.2 Member Function Documentation

9.43.2.1 Bind()

```
void Vesper::OpenGLIndexBuffer::Bind () const  [override], [virtual]
```

Implements Vesper::IndexBuffer.

```
00080     {
00081         VZ_PROFILE_FUNCTION();
00082
00083         glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, m_RendererID);
00084     }
```

9.43.2.2 GetCount()

```
virtual uint32_t Vesper::OpenGLIndexBuffer::GetCount () const  [inline], [override], [virtual]
```

Implements Vesper::IndexBuffer.

```
00034 { return m_Count; }
```

References m_Count.

9.43.2.3 Unbind()

```
void Vesper::OpenGLIndexBuffer::Unbind () const  [override], [virtual]
```

Implements Vesper::IndexBuffer.

```
00087     {
00088         VZ_PROFILE_FUNCTION();
00089
00090         glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
00091     }
```

9.43.3 Member Data Documentation

9.43.3.1 m_Count

```
uint32_t Vesper::OpenGLIndexBuffer::m_Count  [private]
```

Referenced by GetCount(), and OpenGLIndexBuffer().

9.43.3.2 m_RendererID

```
uint32_t Vesper::OpenGLIndexBuffer::m_RendererID  [private]
```

Referenced by OpenGLIndexBuffer(), and ~OpenGLIndexBuffer().

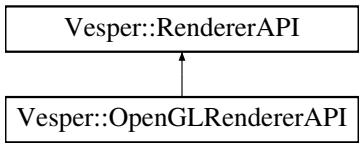
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.h
- Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.cpp

9.44 Vesper::OpenGLRendererAPI Class Reference

```
#include <OpenGLRendererAPI.h>
```

Inheritance diagram for Vesper::OpenGLRendererAPI:



Public Member Functions

- virtual void Init () override
- virtual void SetViewport (uint32_t x, uint32_t y, uint32_t width, uint32_t height) override
- virtual void SetClearColor (const glm::vec4 &color) override
- virtual void Clear () override
- virtual void DrawIndexed (const Ref< VertexArray > &vertexArray, uint32_t indexCount=0) override

Public Member Functions inherited from Vesper::RendererAPI

- virtual ~RendererAPI ()=default

Additional Inherited Members

Public Types inherited from Vesper::RendererAPI

- enum class API { None = 0 , OpenGL = 1 }

Static Public Member Functions inherited from Vesper::RendererAPI

- static API GetAPI ()

9.44.1 Member Function Documentation

9.44.1.1 Clear()

```
void Vesper::OpenGLRendererAPI::Clear ()  [override], [virtual]
```

Implements Vesper::RendererAPI.

```
00030     {
00031         VZ_PROFILE_FUNCTION();
00032 
00033         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
00034     }
```

9.44.1.2 DrawIndexed()

```
void Vesper::OpenGLRendererAPI::DrawIndexed (
            const Ref< VertexArray > & vertexArray,
            uint32_t indexCount = 0)  [override], [virtual]
```

Implements Vesper::RendererAPI.

```
00037     {
00038         VZ_PROFILE_FUNCTION();
00039 
00040         uint32_t count = indexCount ? indexCount : vertexArray->GetIndexBuffer()->GetCount();
00041         glDrawElements(GL_TRIANGLES, count, GL_UNSIGNED_INT, nullptr);
00042         glBindTexture(GL_TEXTURE_2D, 0);
00043     }
```

9.44.1.3 Init()

```
void Vesper::OpenGLRendererAPI::Init ()  [override], [virtual]
```

Implements Vesper::RendererAPI.

```
00009     {
00010         glEnable(GL_BLEND);
00011         glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
00012         glEnable(GL_DEPTH_TEST);
00013     }
```

9.44.1.4 SetClearColor()

```
void Vesper::OpenGLRendererAPI::SetClearColor (
            const glm::vec4 & color)  [override], [virtual]
```

Implements Vesper::RendererAPI.

```
00023     {
00024         VZ_PROFILE_FUNCTION();
00025 
00026         glClearColor(color.r, color.g, color.b, color.a);
00027     }
```

9.44.1.5 SetViewport()

```
void Vesper::OpenGLRendererAPI::SetViewport (
            uint32_t x,
            uint32_t y,
            uint32_t width,
            uint32_t height)  [override], [virtual]
```

Implements Vesper::RendererAPI.

```
00016     {
00017         VZ_PROFILE_FUNCTION();
00018 
00019         glViewport(x, y, width, height);
00020     }
```

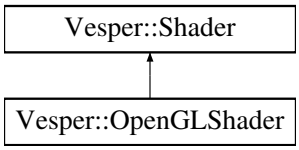
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.h
- Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.cpp

9.45 Vesper::OpenGLShader Class Reference

```
#include <OpenGLShader.h>
```

Inheritance diagram for Vesper::OpenGLShader:



Public Member Functions

- OpenGLShader (const std::string &filepath)
- OpenGLShader (const std::string &name, const std::string &vertexSrc, const std::string &fragmentSrc)
- ~OpenGLShader ()
- void Bind () const override
- void Unbind () const override
- virtual void SetMat4 (const std::string &name, const glm::mat4 &value) override
- virtual void SetFloat4 (const std::string &name, const glm::vec4 &value) override
- virtual void SetFloat3 (const std::string &name, const glm::vec3 &value) override
- virtual void SetFloat (const std::string &name, float value) override
- virtual void SetInt (const std::string &name, int value) override
- virtual void SetIntArray (const std::string &name, int *values, uint32_t count) override
- virtual const std::string & GetName () const override
- void UploadUniformMat4 (const std::string &name, const glm::mat4 &matrix)
- void UploadUniformMat3 (const std::string &name, const glm::mat3 &matrix)
- void UploadUniformFloat4 (const std::string &name, const glm::vec4 &values)
- void UploadUniformFloat3 (const std::string &name, const glm::vec3 &values)
- void UploadUniformFloat2 (const std::string &name, const glm::vec2 &values)
- void UploadUniformFloat (const std::string &name, float value)
- void UploadUniformInt (const std::string &name, int value)

Public Member Functions inherited from Vesper::Shader

- virtual ~Shader ()=default

Private Member Functions

- std::string ReadFile (const std::string &filepath)
- std::unordered_map< GLenum, std::string > PreProcess (const std::string &source)
- void Compile (std::unordered_map< GLenum, std::string > &shaderSources)

Private Attributes

- unsigned int m_RendererID
- std::string m_Name

Additional Inherited Members

Static Public Member Functions inherited from Vesper::Shader

- static Ref< Shader > Create (const std::string &name, const std::string &vertexSrc, const std::string &fragmentSrc)
- static Ref< Shader > Create (const std::string &filepath)

9.45.1 Constructor & Destructor Documentation

9.45.1.1 OpenGLShader() [1/2]

```
Vesper::OpenGLShader::OpenGLShader (
            const std::string & filepath)
```

```
00022     {
00023         VZ_PROFILE_FUNCTION();
00024         std::string shaderSrc = ReadFile(filepath);
00025         auto shaderSources = PreProcess(shaderSrc);
00026         Compile(shaderSources);
00027 
00028         // Extract name from filepath
00029         auto lastSlash = filepath.find_last_of("/\\");
00030         lastSlash = lastSlash == std::string::npos ? 0 : lastSlash + 1;
00031         auto lastDot = filepath.rfind('.');
00032         auto count = lastDot == std::string::npos ? filepath.size() - lastSlash : lastDot - lastSlash;
00033         m_Name = filepath.substr(lastSlash, count);
00034     }
```

9.45.1.2 OpenGLShader() [2/2]

```
Vesper::OpenGLShader::OpenGLShader (
            const std::string & name,
            const std::string & vertexSrc,
            const std::string & fragmentSrc)
```

```
00037         : m_Name(name)
00038     {
00039         VZ_PROFILE_FUNCTION();
00040         std::unordered_map<GLenum, std::string> sources;
00041         sources[GL_VERTEX_SHADER] = vertexSrc;
00042         sources[GL_FRAGMENT_SHADER] = fragmentSrc;
00043         Compile(sources);
00044     }
```

References OpenGLShader().

Referenced by OpenGLShader().

9.45.1.3 ~OpenGLShader()

```
Vesper::OpenGLShader::~OpenGLShader ()
```

```
00047     {
00048         VZ_PROFILE_FUNCTION();
00049         glDeleteProgram(m_RendererID);
00050     }
```

References m_RendererID.

9.45.2 Member Function Documentation

9.45.2.1 Bind()

```
void Vesper::OpenGLShader::Bind () const  [override], [virtual]
```

Implements Vesper::Shader.

```
00176     {
00177         VZ_PROFILE_FUNCTION();
00178         glUseProgram(m_RendererID);
00179     }
```

References m_RendererID.

9.45.2.2 Compile()

```
void Vesper::OpenGLShader::Compile (
             std::unordered_map< GLenum, std::string > & shaderSources)  [private]
```

```
00099     {
00100         VZ_PROFILE_FUNCTION();
00101         GLuint program = glCreateProgram();
00102
00103         std::array<GLenum, 2> shaderIDs;
00104         VZ_CORE_ASSERT(shaderSources.size() <= shaderIDs.size(), "We only support 2 shaders for
      now!");
00105
00106         int glShaderIDIndex = 0;
00107
00108         for (auto& kv : shaderSources)
00109         {
00110             GLenum type = kv.first;
00111             const std::string& source = kv.second;
00112
00113             GLuint shader = glCreateShader(type);
00114
00115             const GLchar* shaderSource = source.c_str();
00116             glShaderSource(shader, 1, &shaderSource, 0);
00117
00118             glCompileShader(shader);
00119
00120             GLint isCompiled = 0;
00121             glGetShaderiv(shader, GL_COMPILE_STATUS, &isCompiled);
00122             if (isCompiled == GL_FALSE)
00123             {
00124                 GLint maxLength = 0;
00125                 glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &maxLength);
00126
00127                 std::vector<GLchar> infoLog(maxLength);
00128                 glGetShaderInfoLog(shader, maxLength, &maxLength, &infoLog[0]);
00129
00130                 glDeleteShader(shader);
00131
00132                 VZ_CORE_ERROR("{0}", infoLog.data());
00133                 VZ_CORE_ASSERT(false, "Shader compilation failure!");
00134                 break;
00135             }
00136             glAttachShader(program, shader);
00137             shaderIDs[glShaderIDIndex++] = shader;
00138         }
```

```
00139
00140         // Link our program
00141         glLinkProgram(program);
00142
00143         // Note the different functions here: glGetProgram* instead of glGetShader*.
00144         GLint isLinked = 0;
00145         glGetProgramiv(program, GL_LINK_STATUS, (int*)&isLinked);
00146         if (isLinked == GL_FALSE)
00147         {
00148             GLint maxLength = 0;
00149             glGetProgramiv(program, GL_INFO_LOG_LENGTH, &maxLength);
00150
00151             // The maxLength includes the NULL character
00152             std::vector<GLchar> infoLog(maxLength);
00153             glGetProgramInfoLog(program, maxLength, &maxLength, &infoLog[0]);
00154
00155             // We don't need the program anymore.
00156             glDeleteProgram(program);
00157
00158             for (auto id : shaderIDs)
00159                 glDeleteShader(id);
00160
00161             VZ_CORE_ERROR("{0}", infoLog.data());
00162             VZ_CORE_ASSERT(false, "Shader link failure!");
00163             return;
00164         }
00165
00166         for (auto id : shaderIDs)
00167         {
00168             glDetachShader(program, id);
00169         }
00170
00171         m_RendererID = program;
00172
00173     }
```

References m_RendererID.

9.45.2.3 GetName()

```
virtual const std::string & Vesper::OpenGLShader::GetName () const  [inline], [override],
[virtual]
```

Implements Vesper::Shader.

```
00028 { return m_Name; }
```

9.45.2.4 PreProcess()

```
std::unordered_map< GLenum, std::string > Vesper::OpenGLShader::PreProcess (
            const std::string & source)  [private]
```

```
00073     {
00074         VZ_PROFILE_FUNCTION();
00075         std::unordered_map<GLenum, std::string> shaderSources;
00076
00077         const char* typeToken = "#type";
00078         size_t typeTokenLength = strlen(typeToken);
00079         size_t pos = source.find(typeToken, 0); // Start of shader type declaration line
00080         while (pos != std::string::npos)
00081         {
00082             size_t eol = source.find_first_of("\r\n", pos); // End of shader type declaration line
00083             VZ_CORE_ASSERT(eol != std::string::npos, "Syntax error");
00084
00085             size_t begin = pos + typeTokenLength + 1; // Start of shader type name (after "#type "
    keyword)
00086             std::string type = source.substr(begin, eol - begin);
00087             VZ_CORE_ASSERT(ShaderTypeFromString(type), "Invalid shader type specified");
00088
00089             size_t nextLinePos = source.find_first_not_of("\r\n", eol); // Start of shader code after
    shader type declaration line
00090             VZ_CORE_ASSERT(nextLinePos != std::string::npos, "Syntax error");
00091
00092             pos = source.find(typeToken, nextLinePos); // Start of next shader type declaration line
00093             shaderSources[ShaderTypeFromString(type)] = (pos == std::string::npos) ?
    source.substr(nextLinePos) : source.substr(nextLinePos, pos - nextLinePos);
00094         }
00095         return shaderSources;
00096     }
```

9.45.2.5 ReadFile()

```
std::string Vesper::OpenGLShader::ReadFile (
            const std::string & filepath)  [private]
```

```
00053     {
00054         VZ_PROFILE_FUNCTION();
00055         std::string result;
00056         std::ifstream in(filepath, std::ios::in | std::ios::binary);
00057         if (in)
00058         {
00059             in.seekg(0, std::ios::end);
00060             result.resize(in.tellg());
00061             in.seekg(0, std::ios::beg);
00062             in.read(&result[0], result.size());
00063         }
00064         else
00065         {
00066             VZ_CORE_ERROR("Could not open file '{0}'", filepath);
00067             VZ_CORE_ASSERT(false, "Failed to open file!");
00068         }
00069         return result;
00070     }
```

9.45.2.6 SetFloat()

```
void Vesper::OpenGLShader::SetFloat (
            const std::string & name,
            float value)  [override], [virtual]
```

Implements Vesper::Shader.

```
00201     {
00202         VZ_PROFILE_FUNCTION();
00203         UploadUniformFloat(name, value);
00204     }
```

References UploadUniformFloat().

9.45.2.7 SetFloat3()

```
void Vesper::OpenGLShader::SetFloat3 (
            const std::string & name,
            const glm::vec3 & value)  [override], [virtual]
```

Implements Vesper::Shader.

```
00195     {
00196         VZ_PROFILE_FUNCTION();
00197         UploadUniformFloat3(name, value);
00198     }
```

9.45.2.8 SetFloat4()

```
void Vesper::OpenGLShader::SetFloat4 (
            const std::string & name,
            const glm::vec4 & value)  [override], [virtual]
```

Implements Vesper::Shader.

```
00189     {
00190         VZ_PROFILE_FUNCTION();
00191         UploadUniformFloat4(name, value);
00192     }
```

9.45.2.9 SetInt()

```
void Vesper::OpenGLShader::SetInt (
            const std::string & name,
            int value)  [override], [virtual]
```

Implements Vesper::Shader.

```
00207     {
00208         VZ_PROFILE_FUNCTION();
00209         UploadUniformInt(name, value);
00210     }
```

References UploadUniformInt().

9.45.2.10 SetIntArray()

```
void Vesper::OpenGLShader::SetIntArray (
            const std::string & name,
            int * values,
            uint32_t count)  [override], [virtual]
```

Implements Vesper::Shader.

```
00213     {
00214         VZ_PROFILE_FUNCTION();
00215         GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00216         glUniform1iv(location, count, values);
00217     }
```

9.45.2.11 SetMat4()

```
void Vesper::OpenGLShader::SetMat4 (
            const std::string & name,
            const glm::mat4 & value)  [override], [virtual]
```

Implements Vesper::Shader.

```
00220     {
00221         VZ_PROFILE_FUNCTION();
00222         UploadUniformMat4(name, value);
00223     }
```

9.45.2.12 Unbind()

```
void Vesper::OpenGLShader::Unbind () const  [override], [virtual]
```

Implements Vesper::Shader.

```
00182     {
00183         VZ_PROFILE_FUNCTION();
00184         glUseProgram(0);
00185     }
```

9.45.2.13 UploadUniformFloat()

```
void Vesper::OpenGLShader::UploadUniformFloat (
            const std::string & name,
            float value)
```

```
00257     {
00258         GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00259         glUniform1f(location, value);
00260     }
```

Referenced by SetFloat().

9.45.2.14 UploadUniformFloat2()

```
void Vesper::OpenGLShader::UploadUniformFloat2 (
            const std::string & name,
            const glm::vec2 & values)
```

```
00251     {
00252         GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00253         glUniform2f(location, values.x, values.y);
00254     }
```

9.45.2.15 UploadUniformFloat3()

```
void Vesper::OpenGLShader::UploadUniformFloat3 (
            const std::string & name,
            const glm::vec3 & values)
```

```
00245     {
00246         GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00247         glUniform3f(location, values.x, values.y, values.z);
00248     }
```

9.45.2.16 UploadUniformFloat4()

```
void Vesper::OpenGLShader::UploadUniformFloat4 (
            const std::string & name,
            const glm::vec4 & values)
```

```
00239     {
00240         GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00241         glUniform4f(location, values.x, values.y, values.z, values.w);
00242     }
```

9.45.2.17 UploadUniformInt()

```
void Vesper::OpenGLShader::UploadUniformInt (
            const std::string & name,
            int value)
```

```
00263     {
00264         GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00265         glUniform1i(location, value);
00266     }
```

Referenced by SetInt().

9.45.2.18 UploadUniformMat3()

```
void Vesper::OpenGLShader::UploadUniformMat3 (
            const std::string & name,
            const glm::mat3 & matrix)
```

```
00233     {
00234         GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00235         glUniformMatrix3fv(location, 1, GL_FALSE, glm::value_ptr(matrix));
00236     }
```

9.45.2.19 UploadUniformMat4()

```
void Vesper::OpenGLShader::UploadUniformMat4 (
            const std::string & name,
            const glm::mat4 & matrix)
00227     {
00228         GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00229         glUniformMatrix4fv(location, 1, GL_FALSE, glm::value_ptr(matrix));
00230     }
```

9.45.3 Member Data Documentation

9.45.3.1 m_Name

```
std::string Vesper::OpenGLShader::m_Name  [private]
```

9.45.3.2 m_RendererID

```
unsigned int Vesper::OpenGLShader::m_RendererID  [private]
```

Referenced by Bind(), Compile(), and ~OpenGLShader().

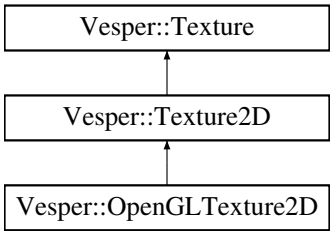
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLShader.h
- Vesper/src/RenderAPI/OpenGL/OpenGLShader.cpp

9.46 Vesper::OpenGLTexture2D Class Reference

```
#include <OpenGLTexture.h>
```

Inheritance diagram for Vesper::OpenGLTexture2D:



Public Member Functions

- OpenGLTexture2D (uint32_t width, uint32_t height)
- OpenGLTexture2D (const std::string &path)
- virtual ~OpenGLTexture2D ()
- virtual uint32_t GetWidth () const override
- virtual uint32_t GetHeight () const override
- virtual uint32_t GetRendererID () const override
- virtual void Bind (uint32_t slot) const override
- virtual void SetData (void *data, uint32_t size) override
- virtual bool operator== (const Texture2D &other) const override
- virtual std::string GetName () const override

Public Member Functions inherited from Vesper::Texture

- virtual ~Texture ()=default

Private Attributes

- std::string m_Path
- uint32_t m_Width
- uint32_t m_Height
- uint32_t m_RendererID
- GLenum m_InternalFormat
- GLenum m_DataFormat

Additional Inherited Members

Static Public Member Functions inherited from Vesper::Texture2D

- static Ref< Texture2D > Create (uint32_t width, uint32_t height)
- static Ref< Texture2D > Create (const std::string &path)

9.46.1 Constructor & Destructor Documentation

9.46.1.1 OpenGLTexture2D() [1/2]

```
Vesper::OpenGLTexture2D::OpenGLTexture2D (
            uint32_t width,
            uint32_t height)
```

```
00010         : m_Width(width), m_Height(height)
00011     {
00012         VZ_PROFILE_FUNCTION();
00013         m_InternalFormat = GL_RGBA8;
00014         m_DataFormat = GL_RGBA;
00015 
00016         glCreateTextures(GL_TEXTURE_2D, 1, &m_RendererID);
00017         glTextureStorage2D(m_RendererID, 1, m_InternalFormat, m_Width, m_Height);
00018 
00019         glTextureParameteri(m_RendererID, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
00020         glTextureParameteri(m_RendererID, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
00021 
00022         glTextureParameteri(m_RendererID, GL_TEXTURE_WRAP_S, GL_REPEAT);
00023         glTextureParameteri(m_RendererID, GL_TEXTURE_WRAP_T, GL_REPEAT);
00024     }
```

References m_Height, and m_Width.

9.46.1.2 OpenGLTexture2D() [2/2]

```
Vesper::OpenGLTexture2D::OpenGLTexture2D (
            const std::string & path)
```

```
00027         : m_Path(path)
00028     {
00029         VZ_PROFILE_FUNCTION();
00030         int width, height, channels;
00031         stbi_set_flip_vertically_on_load(1);
00032         stbi_uc* data = nullptr;
00033         {
00034             VZ_PROFILE_SCOPE("stbi_load - OpenGLTexture2D::OpenGLTexture2D(const std::string&)");
00035             data = stbi_load(path.c_str(), (int*)&width, (int*)&height, &channels, 0);
00036         }
00037         VZ_CORE_ASSERT(data, "Failed to load image from: " + path);
00038 
00039         m_Width = width;
00040         m_Height = height;
00041 
00042         GLenum internalFormat = 0, dataFormat = 0;
00043         if (channels == 4)
00044         {
00045             internalFormat = GL_RGBA8;
00046             dataFormat = GL_RGBA;
00047         }
00048         else if (channels == 3)
00049         {
00050             internalFormat = GL_RGB8;
00051             dataFormat = GL_RGB;
00052         }
00053 
00054         m_InternalFormat = internalFormat;
00055         m_DataFormat = dataFormat;
00056 
00057 
00058         VZ_CORE_ASSERT(internalFormat & dataFormat, "Format not supported!");
00059 
00060         glCreateTextures(GL_TEXTURE_2D, 1, &m_RendererID);
00061         glTextureStorage2D(m_RendererID, 1, internalFormat, m_Width, m_Height);
00062 
00063         glTextureParameteri(m_RendererID, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
00064         glTextureParameteri(m_RendererID, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
00065 
00066         glTextureParameteri(m_RendererID, GL_TEXTURE_WRAP_S, GL_REPEAT);
00067         glTextureParameteri(m_RendererID, GL_TEXTURE_WRAP_T, GL_REPEAT);
00068 
00069         glTextureSubImage2D(m_RendererID, 0, 0, 0, m_Width, m_Height, dataFormat, GL_UNSIGNED_BYTE,
      data);
00070 
00071         stbi_image_free(data);
00072     }
```

References m_Height, m_RendererID, m_Width, and OpenGLTexture2D().

Referenced by OpenGLTexture2D().

9.46.1.3 ~OpenGLTexture2D()

```
Vesper::OpenGLTexture2D::~OpenGLTexture2D ()  [virtual]
```

```
00075     {
00076         VZ_PROFILE_FUNCTION();
00077         glDeleteTextures(1, &m_RendererID);
00078     }
```

References m_RendererID.

9.46.2 Member Function Documentation

9.46.2.1 Bind()

```
void Vesper::OpenGLTexture2D::Bind (
            uint32_t slot) const  [override], [virtual]
```

Implements Vesper::Texture.

```
00081     {
00082         VZ_PROFILE_FUNCTION();
00083         glBindTextureUnit(slot, m_RendererID);
00084     }
```

References m_RendererID.

9.46.2.2 GetHeight()

```
virtual uint32_t Vesper::OpenGLTexture2D::GetHeight () const  [inline], [override], [virtual]
```

Implements Vesper::Texture.

```
00016 { return m_Height; }
```

References m_Height.

9.46.2.3 GetName()

```
std::string Vesper::OpenGLTexture2D::GetName () const  [override], [virtual]
```

Implements Vesper::Texture.

```
00095     {
00096         // Extract filename from path
00097         size_t lastSlash = m_Path.find_last_of("/\\");
00098         if (lastSlash == std::string::npos)
00099             return m_Path; // No directory part
00100         else
00101             return m_Path.substr(lastSlash + 1);
00102 
00103     }
```

9.46.2.4 GetRendererID()

```
virtual uint32_t Vesper::OpenGLTexture2D::GetRendererID () const  [inline], [override], [virtual]
```

Implements Vesper::Texture.

```
00017 { return m_RendererID; }
```

References m_RendererID.

9.46.2.5 GetWidth()

```
virtual uint32_t Vesper::OpenGLTexture2D::GetWidth () const  [inline], [override], [virtual]
```

Implements Vesper::Texture.

```
00015 { return m_Width; }
```

References m_Width.

9.46.2.6 operator==()

```
virtual bool Vesper::OpenGLTexture2D::operator== (
            const Texture2D & other) const  [inline], [override], [virtual]
```

Implements Vesper::Texture.

```
00024         {
00025             return m_RendererID == ((OpenGLTexture2D&)other).m_RendererID;
00026         }
```

References m_RendererID.

9.46.2.7 SetData()

```
void Vesper::OpenGLTexture2D::SetData (
            void * data,
            uint32_t size)  [override], [virtual]
```

Implements Vesper::Texture.

```
00087     {
00088         VZ_PROFILE_FUNCTION();
00089         uint32_t bpp = (m_DataFormat == GL_RGBA ? 4 : 3);
00090         VZ_CORE_ASSERT(size == m_Width * m_Height * bpp, "Data must be entire texture!");
00091         glTextureSubImage2D(m_RendererID, 0, 0, 0, m_Width, m_Height, m_DataFormat, GL_UNSIGNED_BYTE,
      data);
00092     }
```

9.46.3 Member Data Documentation

9.46.3.1 m_DataFormat

```
GLenum Vesper::OpenGLTexture2D::m_DataFormat  [private]
```

9.46.3.2 m_Height

```
uint32_t Vesper::OpenGLTexture2D::m_Height  [private]
```

Referenced by GetHeight(), OpenGLTexture2D(), and OpenGLTexture2D().

9.46.3.3 m_InternalFormat

```
GLenum Vesper::OpenGLTexture2D::m_InternalFormat  [private]
```

9.46.3.4 m_Path

```
std::string Vesper::OpenGLTexture2D::m_Path  [private]
```

9.46.3.5 m_RendererID

```
uint32_t Vesper::OpenGLTexture2D::m_RendererID  [private]
```

Referenced by Bind(), GetRendererID(), OpenGLTexture2D(), operator==(), and ~OpenGLTexture2D().

9.46.3.6 m_Width

```
uint32_t Vesper::OpenGLTexture2D::m_Width  [private]
```

Referenced by GetWidth(), OpenGLTexture2D(), and OpenGLTexture2D().

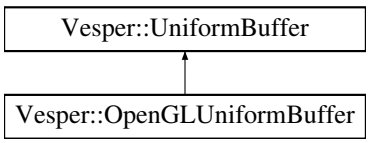
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLTexture.h
- Vesper/src/RenderAPI/OpenGL/OpenGLTexture.cpp

9.47 Vesper::OpenGLUniformBuffer Class Reference

```
#include <OpenGLUniformBuffer.h>
```

Inheritance diagram for Vesper::OpenGLUniformBuffer:



Public Member Functions

- OpenGLUniformBuffer (uint32_t size, uint32_t binding)
- virtual ~OpenGLUniformBuffer ()
- virtual void SetData (const void *data, uint32_t size, uint32_t offset=0) override

Public Member Functions inherited from Vesper::UniformBuffer

- virtual ~UniformBuffer ()

Private Attributes

- uint32_t m_RendererID = 0

Additional Inherited Members

Static Public Member Functions inherited from Vesper::UniformBuffer

- static Ref< UniformBuffer > Create (uint32_t size, uint32_t binding)

9.47.1 Constructor & Destructor Documentation

9.47.1.1 OpenGLUniformBuffer()

```
Vesper::OpenGLUniformBuffer::OpenGLUniformBuffer (
            uint32_t size,
            uint32_t binding)
```

```
00009     {
00010         glCreateBuffers(1, &m_RendererID);
00011         glNamedBufferData(m_RendererID, size, nullptr, GL_DYNAMIC_DRAW); // TODO: investigate usage
      hint
00012         glBindBufferBase(GL_UNIFORM_BUFFER, binding, m_RendererID);
00013     }
```

References m_RendererID.

9.47.1.2 ~OpenGLUniformBuffer()

```
Vesper::OpenGLUniformBuffer::~OpenGLUniformBuffer ()  [virtual]
```

```
00016     {
00017         glDeleteBuffers(1, &m_RendererID);
00018     }
```

References m_RendererID.

9.47.2 Member Function Documentation

9.47.2.1 SetData()

```
void Vesper::OpenGLUniformBuffer::SetData (
            const void * data,
            uint32_t size,
            uint32_t offset = 0)  [override], [virtual]
```

Implements Vesper::UniformBuffer.

```
00022     {
00023         glNamedBufferSubData(m_RendererID, offset, size, data);
00024     }
```

References m_RendererID.

9.47.3 Member Data Documentation

9.47.3.1 m_RendererID

```
uint32_t Vesper::OpenGLUniformBuffer::m_RendererID = 0  [private]
```

Referenced by OpenGLUniformBuffer(), SetData(), and ~OpenGLUniformBuffer().

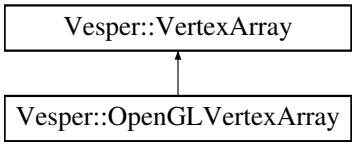
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLUniformBuffer.h
- Vesper/src/RenderAPI/OpenGL/OpenGLUniformBuffer.cpp

9.48 Vesper::OpenGLVertexArray Class Reference

`#include <OpenGLVertexArray.h>`

Inheritance diagram for Vesper::OpenGLVertexArray:



Public Member Functions

- OpenGLVertexArray ()
- ~OpenGLVertexArray ()
- void Bind () const override
- void Unbind () const override
- void AddVertexBuffer (const Ref< VertexBuffer > &vertexBuffer) override
- void SetIndexBuffer (const Ref< IndexBuffer > &indexBuffer) override
- const std::vector< Ref< VertexBuffer > > & GetVertexBuffers () override
- const Ref< IndexBuffer > & GetIndexBuffer () const override

Public Member Functions inherited from Vesper::VertexArray

- virtual ~VertexArray ()

Private Attributes

- uint32_t m_RendererID
- uint32_t m_VertexBufferIndex = 0
- std::vector< Ref< VertexBuffer > > m_VertexBuffers
- Ref< IndexBuffer > m_IndexBuffer

Additional Inherited Members

Static Public Member Functions inherited from Vesper::VertexArray

- static Ref< VertexArray > Create ()

9.48.1 Constructor & Destructor Documentation

9.48.1.1 OpenGLVertexArray()

```
Vesper::OpenGLVertexArray::OpenGLVertexArray ()
00031     {
00032         VZ_PROFILE_FUNCTION();
00033
00034         glCreateVertexArrays(1, &m_RendererID);
00035     }
```

References m_RendererID.

9.48.1.2 ~OpenGLVertexArray()

```
Vesper::OpenGLVertexArray::~OpenGLVertexArray ()
```

```
00038     {
00039         VZ_PROFILE_FUNCTION();
00040 
00041         glDeleteVertexArrays(1, &m_RendererID);
00042     }
```

References m_RendererID.

9.48.2 Member Function Documentation

9.48.2.1 AddVertexBuffer()

```
void Vesper::OpenGLVertexArray::AddVertexBuffer (
             const Ref< VertexBuffer > & vertexBuffer)  [override], [virtual]
```

Implements Vesper::VertexArray.

```
00059     {
00060         VZ_PROFILE_FUNCTION();
00061 
00062         VZ_CORE_ASSERT(vertexBuffer->GetLayout().GetElements().size(), "Vertex Buffer has no
      layout!");
00063         glBindVertexArray(m_RendererID);
00064         vertexBuffer->Bind();
00065 
00066 
00067         uint32_t index = 0;
00068         const auto& layout = vertexBuffer->GetLayout();
00069         for (const auto& element : layout)
00070         {
00071             VZ_PROFILE_SCOPE("VertexBufferElement");
00072             glEnableVertexAttribArray(index);
00073             glVertexAttribPointer(index, element.GetComponentCount(),
00074                 ShaderDataTypeToOpenGLBaseType(element.Type),
00075                 element.Normalized ? GL_TRUE : GL_FALSE,
00076                 layout.GetStride(), (const void*)element.Offset);
00077             index++;
00078         }
00079         m_VertexBuffers.push_back(vertexBuffer);
00080     }
```

References m_RendererID.

9.48.2.2 Bind()

```
void Vesper::OpenGLVertexArray::Bind () const  [override], [virtual]
```

Implements Vesper::VertexArray.

```
00045     {
00046         VZ_PROFILE_FUNCTION();
00047 
00048         glBindVertexArray(m_RendererID);
00049     }
```

References m_RendererID.

9.48.2.3 GetIndexBuffer()

```
const Ref< IndexBuffer > & Vesper::OpenGLVertexArray::GetIndexBuffer () const  [inline], [override],
[virtual]
```

Implements Vesper::VertexArray.

```
00018 { return m_IndexBuffer; }
```

9.48.2.4 GetVertexBuffers()

```
const std::vector< Ref< VertexBuffer > > & Vesper::OpenGLVertexArray::GetVertexBuffers ()
[inline], [override], [virtual]
```

Implements Vesper::VertexArray.

```
00017 { return m_VertexBuffers; }
```

9.48.2.5 SetIndexBuffer()

```
void Vesper::OpenGLVertexArray::SetIndexBuffer (
            const Ref< IndexBuffer > & indexBuffer)  [override], [virtual]
```

Implements Vesper::VertexArray.

```
00083     {
00084         VZ_PROFILE_FUNCTION();
00085
00086         glBindVertexArray(m_RendererID);
00087         indexBuffer->Bind();
00088
00089         m_IndexBuffer = indexBuffer;
00090     }
```

References m_RendererID.

9.48.2.6 Unbind()

```
void Vesper::OpenGLVertexArray::Unbind () const  [override], [virtual]
```

Implements Vesper::VertexArray.

```
00052     {
00053         VZ_PROFILE_FUNCTION();
00054
00055         glBindVertexArray(0);
00056     }
```

9.48.3 Member Data Documentation

9.48.3.1 m_IndexBuffer

```
Ref<IndexBuffer> Vesper::OpenGLVertexArray::m_IndexBuffer  [private]
```

9.48.3.2 m_RendererID

```
uint32_t Vesper::OpenGLVertexArray::m_RendererID  [private]
```

Referenced by AddVertexBuffer(), Bind(), OpenGLVertexArray(), SetIndexBuffer(), and ~OpenGLVertexArray().

9.48.3.3 m_VertexBufferIndex

```
uint32_t Vesper::OpenGLVertexArray::m_VertexBufferIndex = 0  [private]
```

9.48.3.4 m_VertexBuffers

```
std::vector<Ref<VertexBuffer> > Vesper::OpenGLVertexArray::m_VertexBuffers  [private]
```

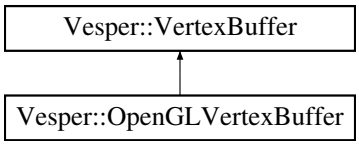
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/OpenGLVertexArray.h
- Vesper/src/RenderAPI/OpenGL/OpenGLVertexArray.cpp

9.49 Vesper::OpenGLVertexBuffer Class Reference

```
#include <OpenGLBuffer.h>
```

Inheritance diagram for Vesper::OpenGLVertexBuffer:



Public Member Functions

- OpenGLVertexBuffer (uint32_t size)
- OpenGLVertexBuffer (float *vertices, uint32_t size)
- virtual ~OpenGLVertexBuffer ()
- virtual void Bind () const override
- virtual void Unbind () const override
- virtual void SetLayout (const BufferLayout &layout) override
- virtual const BufferLayout & GetLayout () const override
- virtual void SetData (const void *data, uint32_t size) override

Public Member Functions inherited from Vesper::VertexBuffer

- virtual ~VertexBuffer ()

Private Attributes

- uint32_t m_RendererID
- BufferLayout m_Layout

Additional Inherited Members

Static Public Member Functions inherited from Vesper::VertexBuffer

- static Ref< VertexBuffer > Create (uint32_t size)
- static Ref< VertexBuffer > Create (float *vertices, uint32_t size)

9.49.1 Constructor & Destructor Documentation

9.49.1.1 OpenGLVertexBuffer() [1/2]

```
Vesper::OpenGLVertexBuffer::OpenGLVertexBuffer (
            uint32_t size)
```

```
00013     {
00014         VZ_PROFILE_FUNCTION();
00015
00016         glCreateBuffers(1, &m_RendererID);
00017         glBindBuffer(GL_ARRAY_BUFFER, m_RendererID);
00018         glBufferData(GL_ARRAY_BUFFER, size, nullptr, GL_DYNAMIC_DRAW);
00019     }
```

References m_RendererID.

9.49.1.2 OpenGLVertexBuffer() [2/2]

```
Vesper::OpenGLVertexBuffer::OpenGLVertexBuffer (
            float * vertices,
            uint32_t size)
```

```
00022     {
00023         VZ_PROFILE_FUNCTION();
00024
00025         glCreateBuffers(1, &m_RendererID);
00026         glBindBuffer(GL_ARRAY_BUFFER, m_RendererID);
00027         glBufferData(GL_ARRAY_BUFFER, size, vertices, GL_STATIC_DRAW);
00028     }
```

References m_RendererID.

9.49.1.3 ~OpenGLVertexBuffer()

```
Vesper::OpenGLVertexBuffer::~OpenGLVertexBuffer ()  [virtual]
```

```
00031     {
00032         VZ_PROFILE_FUNCTION();
00033
00034         glDeleteBuffers(1, &m_RendererID);
00035     }
```

References m_RendererID.

9.49.2 Member Function Documentation

9.49.2.1 Bind()

```
void Vesper::OpenGLVertexBuffer::Bind () const  [override], [virtual]
```

Implements Vesper::VertexBuffer.

```
00038     {
00039         VZ_PROFILE_FUNCTION();
00040
00041         glBindBuffer(GL_ARRAY_BUFFER, m_RendererID);
00042     }
```

9.49.2.2 GetLayout()

```
virtual const BufferLayout & Vesper::OpenGLVertexBuffer::GetLayout () const  [inline], [override],
[virtual]
```

Implements Vesper::VertexBuffer.

```
00018 { return m_Layout; }
```

9.49.2.3 SetData()

```
void Vesper::OpenGLVertexBuffer::SetData (
            const void * data,
            uint32_t size)  [override], [virtual]
```

Implements Vesper::VertexBuffer.

```
00052     {
00053         VZ_PROFILE_FUNCTION();
00054         glBindBuffer(GL_ARRAY_BUFFER, m_RendererID);
00055         glBufferSubData(GL_ARRAY_BUFFER, 0, size, data);
00056     }
```

9.49.2.4 SetLayout()

```
virtual void Vesper::OpenGLVertexBuffer::SetLayout (
            const BufferLayout & layout)  [inline], [override], [virtual]
```

Implements Vesper::VertexBuffer.

```
00017 { m_Layout = layout; }
```

9.49.2.5 Unbind()

```
void Vesper::OpenGLVertexBuffer::Unbind () const  [override], [virtual]
```

Implements Vesper::VertexBuffer.

```
00045     {
00046         VZ_PROFILE_FUNCTION();
00047
00048         glBindBuffer(GL_ARRAY_BUFFER, 0);
00049     }
```

9.49.3 Member Data Documentation

9.49.3.1 m_Layout

```
BufferLayout Vesper::OpenGLVertexBuffer::m_Layout  [private]
```

9.49.3.2 m_RendererID

```
uint32_t Vesper::OpenGLVertexBuffer::m_RendererID  [private]
```

Referenced by OpenGLVertexBuffer(), OpenGLVertexBuffer(), and ~OpenGLVertexBuffer().

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

- Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.h
- Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.cpp

9.50 Vesper::OrthographicCamera Class Reference

`#include <OrthographicCamera.h>`

Public Member Functions

- OrthographicCamera (float left, float right, float bottom, float top)
- void SetProjection (float left, float right, float bottom, float top)
- void SetPosition (const glm::vec3 &position)
- const glm::vec3 & GetPosition () const
- void SetRotation (float rotation)
- const float GetRotation () const
- const glm::mat4 & GetProjectionMatrix () const
- const glm::mat4 & GetViewMatrix () const
- const glm::mat4 & GetViewProjectionMatrix () const

Private Member Functions

- void RecalculateViewMatrix ()

Private Attributes

- glm::mat4 m_ProjectionMatrix
- glm::mat4 m_ViewMatrix
- glm::mat4 m_ViewProjectionMatrix
- glm::vec3 m_Position = { 0.0f, 0.0f, 0.0f }
- float m_Rotation = 0.0f

9.50.1 Constructor & Destructor Documentation

9.50.1.1 OrthographicCamera()

```
Vesper::OrthographicCamera::OrthographicCamera (
            float left,
            float right,
            float bottom,
            float top)
00009         : m_ProjectionMatrix(glm::ortho(left, right, bottom, top, -1.0f, 1.0f)), m_ViewMatrix(1.0f)
00010     {
00011         VZ_PROFILE_FUNCTION();
00012         m_ViewProjectionMatrix = m_ProjectionMatrix * m_ViewMatrix;
00013     }
```

References OrthographicCamera().

Referenced by OrthographicCamera().

9.50.2 Member Function Documentation

9.50.2.1 GetPosition()

```
const glm::vec3 & Vesper::OrthographicCamera::GetPosition () const  [inline]
00014 { return m_Position; }
```

9.50.2.2 GetProjectionMatrix()

```
const glm::mat4 & Vesper::OrthographicCamera::GetProjectionMatrix () const  [inline]
00020 { return m_ProjectionMatrix; }
```

9.50.2.3 GetRotation()

```
const float Vesper::OrthographicCamera::GetRotation () const  [inline]
00017 { return m_Rotation; }
```

References m_Rotation.

9.50.2.4 GetViewMatrix()

```
const glm::mat4 & Vesper::OrthographicCamera::GetViewMatrix () const  [inline]
00021 { return m_ViewMatrix; }
```

9.50.2.5 GetViewProjectionMatrix()

```
const glm::mat4 & Vesper::OrthographicCamera::GetViewProjectionMatrix () const  [inline]
00022 { return m_ViewProjectionMatrix; }
```

9.50.2.6 RecalculateViewMatrix()

```
void Vesper::OrthographicCamera::RecalculateViewMatrix ()  [private]
00023     {
00024         VZ_PROFILE_FUNCTION();
00025         glm::mat4 transform = glm::translate(glm::mat4(1.0f), m_Position) *
00026             glm::rotate(glm::mat4(1.0f), glm::radians(m_Rotation), glm::vec3(0, 0, 1));
00027         m_ViewMatrix = glm::inverse(transform);
00028         m_ViewProjectionMatrix = m_ProjectionMatrix * m_ViewMatrix;
00029     }
```

Referenced by SetPosition(), and SetRotation().

9.50.2.7 SetPosition()

```
void Vesper::OrthographicCamera::SetPosition (
             const glm::vec3 & position)  [inline]
00013 { m_Position = position; RecalculateViewMatrix(); }
```

References RecalculateViewMatrix().

9.50.2.8 SetProjection()

```
void Vesper::OrthographicCamera::SetProjection (
            float left,
            float right,
            float bottom,
            float top)
00016     {
00017         VZ_PROFILE_FUNCTION();
00018         m_ProjectionMatrix = glm::ortho(left, right, bottom, top, -1.0f, 1.0f);
00019         m_ViewProjectionMatrix = m_ProjectionMatrix * m_ViewMatrix;
00020     }
```

9.50.2.9 SetRotation()

```
void Vesper::OrthographicCamera::SetRotation (
            float rotation)  [inline]
00016 { m_Rotation = rotation; RecalculateViewMatrix(); }
```

References m_Rotation, and RecalculateViewMatrix().

9.50.3 Member Data Documentation

9.50.3.1 m_Position

```
glm::vec3 Vesper::OrthographicCamera::m_Position = { 0.0f, 0.0f, 0.0f }  [private]
00030 { 0.0f, 0.0f, 0.0f };
```

9.50.3.2 m_ProjectionMatrix

```
glm::mat4 Vesper::OrthographicCamera::m_ProjectionMatrix  [private]
```

9.50.3.3 m_Rotation

```
float Vesper::OrthographicCamera::m_Rotation = 0.0f  [private]
```

Referenced by GetRotation(), and SetRotation().

9.50.3.4 m_ViewMatrix

```
glm::mat4 Vesper::OrthographicCamera::m_ViewMatrix  [private]
```

9.50.3.5 m_ViewProjectionMatrix

```
glm::mat4 Vesper::OrthographicCamera::m_ViewProjectionMatrix  [private]
```

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

- Vesper/src/Vesper/Renderer/OrthographicCamera.h
- Vesper/src/Vesper/Renderer/OrthographicCamera.cpp

9.51 Vesper::OrthographicCameraBounds Struct Reference

`#include <OrthographicCameraController.h>`

Public Member Functions

- float GetWidth () const
- float GetHeight () const

Public Attributes

- float Left
- float Right
- float Bottom
- float Top

9.51.1 Member Function Documentation

9.51.1.1 GetHeight()

```
float Vesper::OrthographicCameraBounds::GetHeight () const  [inline]
00020 { return Top - Bottom; }
```

References Bottom, and Top.

9.51.1.2 GetWidth()

```
float Vesper::OrthographicCameraBounds::GetWidth () const  [inline]
00019 { return Right - Left; }
```

References Left, and Right.

9.51.2 Member Data Documentation

9.51.2.1 Bottom

```
float Vesper::OrthographicCameraBounds::Bottom
```

Referenced by GetHeight(), and Vesper::OrthographicCameraController::UpdateCameraBounds().

9.51.2.2 Left

```
float Vesper::OrthographicCameraBounds::Left
```

Referenced by GetWidth(), and Vesper::OrthographicCameraController::UpdateCameraBounds().

9.51.2.3 Right

```
float Vesper::OrthographicCameraBounds::Right
```

Referenced by GetWidth(), and Vesper::OrthographicCameraController::UpdateCameraBounds().

9.51.2.4 Top

```
float Vesper::OrthographicCameraBounds::Top
```

Referenced by GetHeight(), and Vesper::OrthographicCameraController::UpdateCameraBounds().

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

- Vesper/src/Vesper/Renderer/OrthographicCameraController.h

9.52 Vesper::OrthographicCameraController Class Reference

```
#include <OrthographicCameraController.h>
```

Public Member Functions

- OrthographicCameraController (float aspectRatio, bool rotation=false)
- void OnUpdate (Timestep ts)
- void OnEvent (Event &e)
- void OnResize (float width, float height)
- OrthographicCamera & GetCamera ()
- const OrthographicCamera & GetCamera () const
- OrthographicCameraBounds GetBounds () const
- glm::vec3 GetPosition () const
- void SetPosition (float x, float y)
- void SetMoveSpeed (float speed)
- float GetMoveSpeed () const
- void SetRotation (float rotation)
- float GetRotation () const
- void SetRotationSpeed (float speed)
- float GetRotationSpeed () const
- float GetAspectRatio () const
- void SetAspectRatio (float aspectRatio)
- bool CanRotate () const
- void SetCanRotate (bool canRotate)
- void SetZoomLevel (float level)
- float GetZoomLevel () const
- void OnImGuiRender ()

  TODO: move to an editor component that can be attached to any system (EditorAbstractionComp).

Private Member Functions

- bool OnMouseScrolled (MouseScrolledEvent &e)
- bool OnWindowResized (WindowResizeEvent &e)
- void UpdateCameraBounds ()
- void OnUpdateBounds ()
- void CalculateView ()

Private Attributes

- float m_AspectRatio
- float m_ZoomLevel = 1.0f
- OrthographicCamera camera
- OrthographicCameraBounds m_Bounds
- bool m_Rotation = true
- glm::vec3 m_CameraPosition = { 0.0f, 0.0f, 0.0f }
- float m_CameraRotation = 0.0f
- float m_CameraMoveSpeed = 5.0f
- float m_CameraRotationSpeed = 180.0f

9.52.1 Constructor & Destructor Documentation

9.52.1.1 OrthographicCameraController()

```
Vesper::OrthographicCameraController::OrthographicCameraController (
            float aspectRatio,
            bool rotation = false)
00014         : m_AspectRatio(aspectRatio), m_Rotation(rotation),
00015         camera(-m_AspectRatio * m_ZoomLevel, m_AspectRatio* m_ZoomLevel, -m_ZoomLevel, m_ZoomLevel),
00016         m_Bounds{ -m_AspectRatio * m_ZoomLevel, m_AspectRatio * m_ZoomLevel, -m_ZoomLevel, m_ZoomLevel
    }
00017     {
00018
00019     }
```

References m_AspectRatio, m_Bounds, m_Rotation, m_ZoomLevel, and OrthographicCameraController().

Referenced by OrthographicCameraController().

9.52.2 Member Function Documentation

9.52.2.1 CalculateView()

```
void Vesper::OrthographicCameraController::CalculateView ()  [private]
00096     {
00097         UpdateCameraBounds();
00098         camera.SetProjection(m_Bounds.Left, m_Bounds.Right, m_Bounds.Bottom, m_Bounds.Top);
00099     }
```

References UpdateCameraBounds().

Referenced by OnMouseScrolled(), OnResize(), SetAspectRatio(), and SetZoomLevel().

9.52.2.2 CanRotate()

```
bool Vesper::OrthographicCameraController::CanRotate () const  [inline]
00053 { return m_Rotation; }
```

References m_Rotation.

Referenced by OnImGuiRender().

9.52.2.3 GetAspectRatio()

```
float Vesper::OrthographicCameraController::GetAspectRatio () const
00124     {
00125         return m_AspectRatio;
00126     }
```

References m_AspectRatio.

9.52.2.4 GetBounds()

```
OrthographicCameraBounds Vesper::OrthographicCameraController::GetBounds () const  [inline]
00036 { return m_Bounds; }
```

References m_Bounds.

9.52.2.5 GetCamera() [1/2]

```
OrthographicCamera & Vesper::OrthographicCameraController::GetCamera ()  [inline]
00034 { return camera; }
```

9.52.2.6 GetCamera() [2/2]

```
const OrthographicCamera & Vesper::OrthographicCameraController::GetCamera () const  [inline]
00035 { return camera; }
```

9.52.2.7 GetMoveSpeed()

```
float Vesper::OrthographicCameraController::GetMoveSpeed () const  [inline]
00042 { return m_CameraMoveSpeed; }
```

References m_CameraMoveSpeed.

Referenced by OnImGuiRender().

9.52.2.8 GetPosition()

```
glm::vec3 Vesper::OrthographicCameraController::GetPosition () const  [inline]
00038 { return m_CameraPosition; }
```

9.52.2.9 GetRotation()

```
float Vesper::OrthographicCameraController::GetRotation () const  [inline]
00045 { return m_CameraRotation; }
```

References m_CameraRotation.

Referenced by OnImGuiRender().

9.52.2.10 GetRotationSpeed()

```
float Vesper::OrthographicCameraController::GetRotationSpeed () const  [inline]
00048 { return m_CameraRotationSpeed; }
```

References m_CameraRotationSpeed.

Referenced by OnImGuiRender().

9.52.2.11 GetZoomLevel()

```
float Vesper::OrthographicCameraController::GetZoomLevel () const  [inline]
00057 { return m_ZoomLevel; }
```

References m_ZoomLevel.

9.52.2.12 OnEvent()

```
void Vesper::OrthographicCameraController::OnEvent (
              Event & e)
00048     {
00049         VZ_PROFILE_FUNCTION();
00050         EventDispatcher dispatcher(e);
00051
      dispatcher.Dispatch<MouseScrolledEvent>(VZ_BIND_EVENT_FN(OrthographicCameraController::OnMouseScrolled));
00052
      dispatcher.Dispatch<WindowResizeEvent>(VZ_BIND_EVENT_FN(OrthographicCameraController::OnWindowResized));
00053
00054     }
```

References Vesper::EventDispatcher::EventDispatcher(), OnMouseScrolled(), and OnWindowResized().

9.52.2.13 OnImGuiRender()

```
void Vesper::OrthographicCameraController::OnImGuiRender ()
```

TODO: move to an editor component that can be attached to any system (EditorAbstractionComp).

```
00137     {
00138         //ImGui::Begin("Settings");
00139 
00140         static float camPos[3] = { GetPosition().x, GetPosition().y, GetPosition().z };
00141         if (ImGui::DragFloat3("Cam Pos", camPos, 0.1f)) {
00142             SetPosition(camPos[0], camPos[1]);
00143         }
00144         static float camMoveSpeed = GetMoveSpeed();
00145         if (ImGui::DragFloat("Cam Speed", &camMoveSpeed, 0.1f)) {
00146             SetMoveSpeed(camMoveSpeed);
00147         }
00148         ImGui::Separator();
00149         static bool rotate = CanRotate();
00150         if (ImGui::Checkbox("Rotate?", &rotate)) {
00151             SetCanRotate(rotate);
00152         }
00153         ImGui::Separator();
00154         static float camRot = GetRotation();
00155         if (ImGui::DragFloat("Cam Rotation", &camRot, 0.1f)) {
00156             SetRotation(camRot);
00157         }
00158         static float camRotSpeed = GetRotationSpeed();
00159         if (ImGui::DragFloat("Cam Rot Speed", &camRotSpeed, 1.0f)) {
00160             SetRotationSpeed(camRotSpeed);
00161         }
00162         //ImGui::End();
00163 
00164     }
```

References CanRotate(), GetMoveSpeed(), GetRotation(), GetRotationSpeed(), SetCanRotate(), SetMoveSpeed(), SetPosition(), SetRotation(), and SetRotationSpeed().

9.52.2.14 OnMouseScrolled()

```
bool Vesper::OrthographicCameraController::OnMouseScrolled (
            MouseScrolledEvent & e)  [private]
```

```
00066     {
00067         VZ_PROFILE_FUNCTION();
00068         m_ZoomLevel -= e.GetYOffset() * 0.25f;
00069         m_ZoomLevel = std::max(m_ZoomLevel, 0.25f);
00070         CalculateView();
00071         return false;
00072     }
```

References CalculateView(), Vesper::MouseScrolledEvent::GetYOffset(), and m_ZoomLevel.

Referenced by OnEvent().

9.52.2.15 OnResize()

```
void Vesper::OrthographicCameraController::OnResize (
            float width,
            float height)
```

```
00057     {
00058         VZ_PROFILE_FUNCTION();
00059         m_AspectRatio = width / height;
00060         CalculateView();
00061 
00062     }
```

References CalculateView(), and m_AspectRatio.

Referenced by OnWindowResized().

9.52.2.16 OnUpdate()

```
void Vesper::OrthographicCameraController::OnUpdate (
            Timestep ts)
```

```
00022     {
00023         VZ_PROFILE_FUNCTION();
00024 
00025         if (Input::IsKeyPressed(VZ_KEY_A))
00026             m_CameraPosition.x -= m_CameraMoveSpeed * ts;
00027         else if (Input::IsKeyPressed(VZ_KEY_D))
00028             m_CameraPosition.x += m_CameraMoveSpeed * ts;
00029 
00030         if (Input::IsKeyPressed(VZ_KEY_W))
00031             m_CameraPosition.y += m_CameraMoveSpeed * ts;
00032         else if (Input::IsKeyPressed(VZ_KEY_S))
00033             m_CameraPosition.y -= m_CameraMoveSpeed * ts;
00034 
00035         if (m_Rotation)
00036         {
00037             if (Input::IsKeyPressed(VZ_KEY_Q))
00038                 m_CameraRotation -= m_CameraRotationSpeed * ts;
00039             else if (Input::IsKeyPressed(VZ_KEY_E))
00040                 m_CameraRotation += m_CameraRotationSpeed * ts;
00041 
00042             camera.SetRotation(m_CameraRotation);
00043         }
00044         camera.SetPosition(m_CameraPosition);
00045     }
```

References Vesper::Input::IsKeyPressed(), m_CameraRotation, m_CameraRotationSpeed, and m_Rotation.

9.52.2.17 OnUpdateBounds()

```
void Vesper::OrthographicCameraController::OnUpdateBounds ()  [private]
```

```
00091     {
00092         // Nothing for now
00093     }
```

Referenced by UpdateCameraBounds().

9.52.2.18 OnWindowResized()

```
bool Vesper::OrthographicCameraController::OnWindowResized (
            WindowResizeEvent & e)  [private]
```

```
00075     {
00076         VZ_PROFILE_FUNCTION();
00077         OnResize((float)e.GetWidth(), (float)e.GetHeight());
00078         return false;
00079     }
```

References Vesper::WindowResizeEvent::GetHeight(), Vesper::WindowResizeEvent::GetWidth(), and OnResize().

Referenced by OnEvent().

9.52.2.19 SetAspectRatio()

```
void Vesper::OrthographicCameraController::SetAspectRatio (
            float aspectRatio)
```

```
00129     {
00130         m_AspectRatio = aspectRatio;
00131         CalculateView();
00132     }
```

References CalculateView(), and m_AspectRatio.

9.52.2.20 SetCanRotate()

```
void Vesper::OrthographicCameraController::SetCanRotate (
            bool canRotate)  [inline]
00054 { m_Rotation = canRotate; }
```

References m_Rotation.

Referenced by OnImGuiRender().

9.52.2.21 SetMoveSpeed()

```
void Vesper::OrthographicCameraController::SetMoveSpeed (
            float speed)
00108     {
00109         m_CameraMoveSpeed = speed;
00110     }
```

References m_CameraMoveSpeed.

Referenced by OnImGuiRender().

9.52.2.22 SetPosition()

```
void Vesper::OrthographicCameraController::SetPosition (
            float x,
            float y)
00102     {
00103         m_CameraPosition = { x, y, 0.0f };
00104         camera.SetPosition(m_CameraPosition);
00105     }
```

Referenced by OnImGuiRender().

9.52.2.23 SetRotation()

```
void Vesper::OrthographicCameraController::SetRotation (
            float rotation)
00113     {
00114         m_CameraRotation = rotation;
00115         camera.SetRotation(m_CameraRotation);
00116     }
```

References m_CameraRotation.

Referenced by OnImGuiRender().

9.52.2.24 SetRotationSpeed()

```
void Vesper::OrthographicCameraController::SetRotationSpeed (
            float speed)
00119     {
00120         m_CameraRotationSpeed = speed;
00121     }
```

References m_CameraRotationSpeed.

Referenced by OnImGuiRender().

9.52.2.25 SetZoomLevel()

```
void Vesper::OrthographicCameraController::SetZoomLevel (
            float level)  [inline]
00056 { m_ZoomLevel = level; CalculateView(); }
```

References CalculateView(), and m_ZoomLevel.

9.52.2.26 UpdateCameraBounds()

```
void Vesper::OrthographicCameraController::UpdateCameraBounds ()  [private]
00082     {
00083         m_Bounds.Left = -m_AspectRatio * m_ZoomLevel;
00084         m_Bounds.Right = m_AspectRatio * m_ZoomLevel;
00085         m_Bounds.Bottom = -m_ZoomLevel;
00086         m_Bounds.Top = m_ZoomLevel;
00087         OnUpdateBounds();
00088     }
```

References Vesper::OrthographicCameraBounds::Bottom, Vesper::OrthographicCameraBounds::Left, m_AspectRatio, m_Bounds, m_ZoomLevel, OnUpdateBounds(), Vesper::OrthographicCameraBounds::Right, and Vesper::OrthographicCameraBounc

Referenced by CalculateView().

9.52.3 Member Data Documentation

9.52.3.1 camera

```
OrthographicCamera Vesper::OrthographicCameraController::camera  [private]
```

9.52.3.2 m_AspectRatio

```
float Vesper::OrthographicCameraController::m_AspectRatio  [private]
```

Referenced by GetAspectRatio(), OnResize(), OrthographicCameraController(), SetAspectRatio(), and UpdateCameraBounds().

9.52.3.3 m_Bounds

```
OrthographicCameraBounds Vesper::OrthographicCameraController::m_Bounds  [private]
```

Referenced by GetBounds(), OrthographicCameraController(), and UpdateCameraBounds().

9.52.3.4 m_CameraMoveSpeed

```
float Vesper::OrthographicCameraController::m_CameraMoveSpeed = 5.0f  [private]
```

Referenced by GetMoveSpeed(), and SetMoveSpeed().

9.52.3.5 m_CameraPosition

```
glm::vec3 Vesper::OrthographicCameraController::m_CameraPosition = { 0.0f, 0.0f, 0.0f }  [private]
00074 { 0.0f, 0.0f, 0.0f };
```

9.52.3.6 m_CameraRotation

```
float Vesper::OrthographicCameraController::m_CameraRotation = 0.0f  [private]
```

Referenced by GetRotation(), OnUpdate(), and SetRotation().

9.52.3.7 m_CameraRotationSpeed

```
float Vesper::OrthographicCameraController::m_CameraRotationSpeed = 180.0f  [private]
```

Referenced by GetRotationSpeed(), OnUpdate(), and SetRotationSpeed().

9.52.3.8 m_Rotation

```
bool Vesper::OrthographicCameraController::m_Rotation = true  [private]
```

Referenced by CanRotate(), OnUpdate(), OrthographicCameraController(), and SetCanRotate().

9.52.3.9 m_ZoomLevel

```
float Vesper::OrthographicCameraController::m_ZoomLevel = 1.0f  [private]
```

Referenced by GetZoomLevel(), OnMouseScrolled(), OrthographicCameraController(), SetZoomLevel(), and UpdateCameraBounds().

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

- Vesper/src/Vesper/Renderer/OrthographicCameraController.h
- Vesper/src/Vesper/Renderer/OrthographicCameraController.cpp

9.53 Vesper::ParticleSystem Class Reference

```
#include <ParticleSystem.h>
```

Classes

- struct Particle

Public Member Functions

- ParticleSystem ()
- ParticleSystem (uint32_t maxParticles)
- void OnUpdate (Timestep ts)
- void OnRender (OrthographicCamera &camera)
- void Emit (const ParticleProps &particleProps)
- void SetParticleProps (const ParticleProps &particleProps)

Private Attributes

- std::vector< Particle > m_ParticlePool
- uint32_t m_PoolIndex = 999
- ParticleProps m_Props

9.53.1 Class Documentation

9.53.1.1 struct Vesper::ParticleSystem::Particle

Class Members

bool	Active = false	
vec4	ColorBegin	
vec4	ColorEnd	
float	LifeRemaining = 0.0f	
float	LifeTime = 0.0f	
vec3	Position	
float	Rotation	
float	SizeBegin	
float	SizeEnd	
vec3	Velocity	

9.53.2 Constructor & Destructor Documentation

9.53.2.1 ParticleSystem() [1/2]

```
Vesper::ParticleSystem::ParticleSystem ()
00013     {
00014         m_PoolIndex = 999;
00015         m_ParticlePool.resize(1000);
00016     }
```

References m_PoolIndex.

9.53.2.2 ParticleSystem() [2/2]

```
Vesper::ParticleSystem::ParticleSystem (
            uint32_t maxParticles)
00019         : m_PoolIndex(maxParticles - 1)
00020     {
00021         m_ParticlePool.resize(maxParticles);
00022     }
```

References m_PoolIndex.

9.53.3 Member Function Documentation

9.53.3.1 Emit()

```
void Vesper::ParticleSystem::Emit (
            const ParticleProps & particleProps)
00064     {
00065         Particle& particle = m_ParticlePool[m_PoolIndex];
00066         particle.Active = true;
00067         particle.Position = particleProps.Position;
00068         particle.Rotation = particleProps.Rotation + particleProps.RotationVariation *
      (Vesper::Random::Float1() - 0.5f);
00069
00070         particle.Velocity = particleProps.Velocity;
00071         particle.Velocity.x += particleProps.VelocityVariation.x * (Vesper::Random::Float1() - 0.5f);
00072         particle.Velocity.y += particleProps.VelocityVariation.y * (Vesper::Random::Float1() - 0.5f);
00073
00074         particle.ColorBegin = particleProps.ColorBegin;
00075         particle.ColorEnd = particleProps.ColorEnd;
00076
00077         particle.SizeBegin = particleProps.SizeBegin + particleProps.SizeVariation *
      (Vesper::Random::Float1() - 0.5f);
00078         particle.SizeEnd = particleProps.SizeEnd;
00079
00080         particle.LifeTime = particleProps.LifeTime + particleProps.LifetimeVariation *
      (Vesper::Random::Float1() - 0.5f);
00081         particle.LifeRemaining = particle.LifeTime;
00082         m_PoolIndex = --m_PoolIndex % m_ParticlePool.size();
00083     }
```

References Vesper::ParticleSystem::Particle::Active, Vesper::Random::Float1(), Vesper::ParticleSystem::Particle::LifeRemaining, Vesper::ParticleProps::LifeTime, Vesper::ParticleSystem::Particle::LifeTime, Vesper::ParticleProps::LifetimeVariation, Vesper::ParticleProps::Rotation, Vesper::ParticleSystem::Particle::Rotation, Vesper::ParticleProps::RotationVariation, Vesper::ParticleProps::SizeBegin, Vesper::ParticleSystem::Particle::SizeBegin, Vesper::ParticleProps::SizeEnd, Vesper::ParticleSystem::Particle::SizeEnd, and Vesper::ParticleProps::SizeVariation.

9.53.3.2 OnRender()

```
void Vesper::ParticleSystem::OnRender (
            Vesper::OrthographicCamera & camera)
00044     {
00045         for (auto& particle : m_ParticlePool)
00046         {
00047             if (!particle.Active)
00048                 continue;
00049
00050             float life = particle.LifeRemaining / particle.LifeTime;
00051             glm::vec4 color = glm::lerp(particle.ColorEnd, particle.ColorBegin, life);
00052             float size = glm::lerp(particle.SizeEnd, particle.SizeBegin, life);
00053
00054             Vesper::Renderer2D::DrawQuadRotatedWithTexture(
00055                 { particle.Position },
00056                 { size, size },
00057                 Vesper::Renderer2D::GetWhiteTexture(),
00058                 particle.Rotation, 1.0f, color);
00059
00060         }
00061     }
```

9.53.3.3 OnUpdate()

```
void Vesper::ParticleSystem::OnUpdate (
            Vesper::Timestep ts)
```

```
00025     {
00026         for (auto& particle : m_ParticlePool)
00027         {
00028             if (!particle.Active)
00029                 continue;
00030
00031             if (particle.LifeRemaining <= 0.0f)
00032             {
00033                 particle.Active = false;
00034                 continue;
00035             }
00036
00037             particle.LifeRemaining -= ts;
00038             particle.Position += particle.Velocity * (float)ts;
00039             particle.Rotation += 0.01f * ts;
00040         }
00041     }
```

9.53.3.4 SetParticleProps()

```
void Vesper::ParticleSystem::SetParticleProps (
            const ParticleProps & particleProps)  [inline]
```

```
00033 { m_Props = particleProps; }
```

9.53.4 Member Data Documentation

9.53.4.1 m_ParticlePool

```
std::vector<Particle> Vesper::ParticleSystem::m_ParticlePool  [private]
```

9.53.4.2 m_PoolIndex

```
uint32_t Vesper::ParticleSystem::m_PoolIndex = 999  [private]
```

Referenced by ParticleSystem(), and ParticleSystem().

9.53.4.3 m_Props

```
ParticleProps Vesper::ParticleSystem::m_Props  [private]
```

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

- Vesper/src/Vesper/ParticleSystem/ParticleSystem.h
- Vesper/src/Vesper/ParticleSystem/ParticleSystem.cpp

9.54 Vesper::RenderCommand Class Reference

```
#include <RenderCommand.h>
```

Static Public Member Functions

- static void Init ()
- static void SetViewport (uint32_t x, uint32_t y, uint32_t width, uint32_t height)
- static void SetClearColor (const glm::vec4 &color)
- static void Clear ()
- static void DrawIndexed (const Ref< VertexArray > &vertexArray, uint32_t indexCount=0)

Static Private Attributes

- static RendererAPI * s_RendererAPI = new OpenGLRendererAPI()

9.54.1 Member Function Documentation

9.54.1.1 Clear()

```
void Vesper::RendererCommand::Clear ()  [inline], [static]
00027         {
00028             s_RendererAPI->Clear();
00029         }
```

References Vesper::RendererAPI::Clear(), and s_RendererAPI.

Referenced by Vesper::EditorLayer::OnUpdate().

9.54.1.2 DrawIndexed()

```
void Vesper::RendererCommand::DrawIndexed (
             const Ref< VertexArray > & vertexArray,
             uint32_t indexCount = 0)  [inline], [static]
00032         {
00033             s_RendererAPI->DrawIndexed(vertexArray, indexCount);
00034         }
```

References s_RendererAPI.

9.54.1.3 Init()

```
void Vesper::RendererCommand::Init ()  [inline], [static]
00012         {
00013             s_RendererAPI->Init();
00014         }
```

References Vesper::RendererAPI::Init(), and s_RendererAPI.

Referenced by Vesper::Renderer::Init().

9.54.1.4 SetClearColor()

```
void Vesper::RenderCommand::SetClearColor (
            const glm::vec4 & color)  [inline], [static]
00022         {
00023             s_RendererAPI->SetClearColor(color);
00024         }
```

References s_RendererAPI.

9.54.1.5 SetViewport()

```
void Vesper::RenderCommand::SetViewport (
            uint32_t x,
            uint32_t y,
            uint32_t width,
            uint32_t height)  [inline], [static]
00017         {
00018             s_RendererAPI->SetViewport(x, y, width, height);
00019         }
```

References s_RendererAPI, and Vesper::RendererAPI::SetViewport().

Referenced by Vesper::Renderer::OnWindowResize().

9.54.2 Member Data Documentation

9.54.2.1 s_RendererAPI

```
RendererAPI * Vesper::RenderCommand::s_RendererAPI = new OpenGLRendererAPI()  [static], [private]
```

Referenced by Clear(), DrawIndexed(), Init(), SetClearColor(), and SetViewport().

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

- Vesper/src/Vesper/Renderer/RenderCommand.h
- Vesper/src/Vesper/Renderer/RenderCommand.cpp

9.55 Vesper::Renderer Class Reference

```
#include <Renderer.h>
```

Classes

- struct SceneData

Static Public Member Functions

- static void Init ()
- static void OnWindowResize (uint32_t width, uint32_t height)
- static void BeginScene (OrthographicCamera &camera)
- static void EndScene ()
- static void Submit (const Ref< Shader > &shader, const Ref< VertexArray > &vertexArray, const glm::mat4 &transform=glm::mat4(1.0f))
- static RendererAPI::API GetAPI ()

Static Private Attributes

- static SceneData * s_SceneData = new Renderer::SceneData

9.55.1 Class Documentation

9.55.1.1 struct Vesper::Renderer::SceneData

Class Members

mat4	ViewProjectionMatrix	

9.55.2 Member Function Documentation

9.55.2.1 BeginScene()

```
void Vesper::Renderer::BeginScene (
            OrthographicCamera & camera)  [static]
00027     {
00028         VZ_PROFILE_FUNCTION();
00029         s_SceneData->ViewProjectionMatrix = camera.GetViewProjectionMatrix();
00030     }
```

References s_SceneData.

9.55.2.2 EndScene()

```
void Vesper::Renderer::EndScene ()  [static]
00033     {
00034         VZ_PROFILE_FUNCTION();
00035
00036     }
```

9.55.2.3 GetAPI()

```
RendererAPI::API Vesper::Renderer::GetAPI ()  [inline], [static]
00023 { return RendererAPI::GetAPI(); }
```

References Vesper::RendererAPI::GetAPI().

Referenced by Vesper::Framebuffer::Create(), Vesper::IndexBuffer::Create(), Vesper::Shader::Create(), Vesper::Shader::Create(), Vesper::Texture2D::Create(), Vesper::Texture2D::Create(), Vesper::UniformBuffer::Create(), Vesper::VertexArray::Create(), Vesper::VertexBuffer::Create(), and Vesper::VertexBuffer::Create().

9.55.2.4 Init()

```
void Vesper::Renderer::Init ()  [static]
00013     {
00014         VZ_PROFILE_FUNCTION();
00015
00016         RenderCommand::Init();
00017         Renderer2D::Init();
00018     }
```

References Vesper::RenderCommand::Init(), and Vesper::Renderer2D::Init().

Referenced by Vesper::Application::Application().

9.55.2.5 OnWindowResize()

```
void Vesper::Renderer::OnWindowResize (
             uint32_t width,
             uint32_t height)  [static]
00021     {
00022         VZ_PROFILE_FUNCTION();
00023         RenderCommand::SetViewport(0, 0, width, height);
00024     }
```

References Vesper::RenderCommand::SetViewport().

Referenced by Vesper::Application::OnWindowResize().

9.55.2.6 Submit()

```
void Vesper::Renderer::Submit (
             const Ref< Shader > & shader,
             const Ref< VertexArray > & vertexArray,
             const glm::mat4 & transform = glm::mat4(1.0f))  [static]
00039     {
00040         VZ_PROFILE_FUNCTION();
00041         shader->Bind();
00042         std::dynamic_pointer_cast<OpenGLShader>(shader)->UploadUniformMat4("u_ViewProjection",
      s_SceneData->ViewProjectionMatrix);
00043         std::dynamic_pointer_cast<OpenGLShader>(shader)->UploadUniformMat4("u_Transform", transform);
00044
00045         vertexArray->Bind();
00046         RenderCommand::DrawIndexed(vertexArray);
00047     }
```

References s_SceneData.

9.55.3 Member Data Documentation

9.55.3.1 s_SceneData

```
Renderer::SceneData * Vesper::Renderer::s_SceneData = new Renderer::SceneData  [static], [private]
```

Referenced by BeginScene(), and Submit().

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

- Vesper/src/Vesper/Renderer/Renderer.h
- Vesper/src/Vesper/Renderer/Renderer.cpp

9.56 Vesper::Renderer2D Class Reference

`#include <Renderer2D.h>`

Classes

- struct Statistics

Static Public Member Functions

- static void Init ()
- static void Shutdown ()
- static void BeginScene (const Camera &camera, const glm::mat4 &transform)
- static void BeginScene (const EditorCamera &camera)
- static void BeginScene (const OrthographicCamera &camera)
- static void EndScene ()
- static void Flush ()
- static void DrawQuad (const glm::mat4 &transform, const glm::vec4 &color)
- static void DrawQuad (const glm::vec2 &position, const glm::vec2 &size, const glm::vec4 &color)
- static void DrawQuad (const glm::vec3 &position, const glm::vec2 &size, const glm::vec4 &color)
- static void DrawQuadWithTexture (const glm::mat4 &transform, const Ref< Texture2D > &texture, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadWithTexture (const glm::vec2 &position, const glm::vec2 &size, const Ref< Texture2D > &texture, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadWithTexture (const glm::vec3 &position, const glm::vec2 &size, const Ref< Texture2D > &texture, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadWithTexture (const glm::mat4 &transform, const Ref< SubTexture2D > &subtexture, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadWithTexture (const glm::vec2 &position, const glm::vec2 &size, const Ref< SubTexture2D > &subtexture, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadWithTexture (const glm::vec3 &position, const glm::vec2 &size, const Ref< SubTexture2D > &subtexture, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadRotated (const glm::mat4 &transform, const glm::vec4 &color)
- static void DrawQuadRotated (const glm::vec2 &position, const glm::vec2 &size, float rotationRads, const glm::vec4 &color)
- static void DrawQuadRotated (const glm::vec3 &position, const glm::vec2 &size, float rotationRads, const glm::vec4 &color)
- static void DrawQuadRotatedWithTexture (const glm::mat4 &transform, const Ref< Texture2D > &texture, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadRotatedWithTexture (const glm::vec2 &position, const glm::vec2 &size, const Ref< Texture2D > &texture, float rotationRads, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadRotatedWithTexture (const glm::vec3 &position, const glm::vec2 &size, const Ref< Texture2D > &texture, float rotationRads, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadRotatedWithTexture (const glm::mat4 &transform, const Ref< SubTexture2D > &subtexture, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadRotatedWithTexture (const glm::vec2 &position, const glm::vec2 &size, const Ref< SubTexture2D > &subtexture, float rotationRads, float tilingFactor, const glm::vec4 tintColor)
- static void DrawQuadRotatedWithTexture (const glm::vec3 &position, const glm::vec2 &size, const Ref< SubTexture2D > &subtexture, float rotationRads, float tilingFactor, const glm::vec4 tintColor)
- static Ref< Texture2D > GetWhiteTexture ()
- static void ResetStats ()
- static Statistics GetStats ()

Static Private Member Functions

- static void FlushAndReset ()
- static void StartBatch ()

9.56.1 Member Function Documentation

9.56.1.1 BeginScene() [1/3]

```
void Vesper::Renderer2D::BeginScene (
            const Camera & camera,
            const glm::mat4 & transform)  [static]
```

```
00124     {
00125         VZ_PROFILE_FUNCTION();
00126
00127         glm::mat4 viewProj = camera.GetProjection() * glm::inverse(transform);
00128
00129         s_Data.TextureShader->Bind();
00130         s_Data.TextureShader->SetMat4("u_ViewProjection", viewProj);
00131
00132         s_Data.QuadIndexCount = 0;
00133         s_Data.QuadVertexBufferPtr = s_Data.QuadVertexBufferBase;
00134
00135         s_Data.TextureSlotIndex = 1;
00136     }
```

References Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBufferBase, Vesper::Renderer2DData::QuadVertexBufferPtr, Vesper::s_Data, and Vesper::Renderer2DData::TextureSlotIndex.

9.56.1.2 BeginScene() [2/3]

```
void Vesper::Renderer2D::BeginScene (
            const EditorCamera & camera)  [static]
```

```
00139     {
00140         VZ_PROFILE_FUNCTION();
00141
00142         glm::mat4 viewProj = camera.GetViewProjection();
00143         s_Data.TextureShader->Bind();
00144         s_Data.TextureShader->SetMat4("u_ViewProjection", viewProj);
00145         StartBatch();
00146     }
```

References Vesper::s_Data, and StartBatch().

Referenced by Vesper::Scene::OnUpdateEditor().

9.56.1.3 BeginScene() [3/3]

```
void Vesper::Renderer2D::BeginScene (
            const OrthographicCamera & camera)  [static]
```

```
00149     {
00150         VZ_PROFILE_FUNCTION();
00151         s_Data.TextureShader->Bind();
00152         s_Data.TextureShader->SetMat4("u_ViewProjection", camera.GetViewProjectionMatrix());
00153
00154         s_Data.QuadIndexCount = 0;
00155         s_Data.QuadVertexBufferPtr = s_Data.QuadVertexBufferBase;
00156
00157         s_Data.TextureSlotIndex = 1;
00158     }
```

References Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBufferBase, Vesper::Renderer2DData::QuadVertexBufferPtr, Vesper::s_Data, and Vesper::Renderer2DData::TextureSlotIndex.

9.56.1.4 DrawQuad() [1/3]

```
void Vesper::Renderer2D::DrawQuad (
            const glm::mat4 & transform,
            const glm::vec4 & color)  [static]
```

```
00181     {
00182         VZ_PROFILE_FUNCTION();
00183         constexpr size_t quadVertexCount = 4;
00184         constexpr glm::vec2 texCoords[quadVertexCount] = {
00185             { 0.0f, 0.0f },
00186             { 1.0f, 0.0f },
00187             { 1.0f, 1.0f },
00188             { 0.0f, 1.0f }
00189         };
00190         if (s_Data.QuadIndexCount >= s_Data.MaxIndices)
00191             FlushAndReset();
00192         const float texIndex = 0.0f; // White Texture
00193         const float tilingFactor = 1.0f;
00194         for (size_t i = 0; i < quadVertexCount; i++)
00195         {
00196             s_Data.QuadVertexBufferPtr->Position = transform * s_Data.QuadVertexPositions[i];
00197             s_Data.QuadVertexBufferPtr->Color = color;
00198             s_Data.QuadVertexBufferPtr->TexCoord = texCoords[i];
00199             s_Data.QuadVertexBufferPtr->TexIndex = texIndex;
00200             s_Data.QuadVertexBufferPtr->TilingFactor = tilingFactor;
00201             s_Data.QuadVertexBufferPtr++;
00202         }
00203         s_Data.QuadIndexCount += 6;
00204         s_Data.Stats.QuadCount++;
00205
00206     }
```

References FlushAndReset(), Vesper::Renderer2DData::MaxIndices, Vesper::Renderer2D::Statistics::QuadCount, Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBufferPtr, Vesper::s_Data, Vesper::Renderer2DData::Stats, Vesper::QuadVertex::TexIndex, and Vesper::QuadVertex::TilingFactor.

9.56.1.5 DrawQuad() [2/3]

```
void Vesper::Renderer2D::DrawQuad (
            const glm::vec2 & position,
            const glm::vec2 & size,
            const glm::vec4 & color)  [static]
```

```
00208     {
00209         DrawQuad({ position.x, position.y, 0.0f }, size, color);
00210     }
```

9.56.1.6 DrawQuad() [3/3]

```
void Vesper::Renderer2D::DrawQuad (
            const glm::vec3 & position,
            const glm::vec2 & size,
            const glm::vec4 & color)  [static]
```

```
00212     {
00213         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position) * glm::scale(glm::mat4(1.0f),
      { size.x, size.y, 1.0f });
00214         DrawQuad(transform, color);
00215
00216     }
```

9.56.1.7 DrawQuadRotated() [1/3]

```
void Vesper::Renderer2D::DrawQuadRotated (
            const glm::mat4 & transform,
            const glm::vec4 & color)  [static]
```

```
00331     {
00332         VZ_PROFILE_FUNCTION();
00333         constexpr size_t quadVertexCount = 4;
00334         constexpr glm::vec2 texCoords[quadVertexCount] = {
00335             { 0.0f, 0.0f },
00336             { 1.0f, 0.0f },
00337             { 1.0f, 1.0f },
00338             { 0.0f, 1.0f }
00339         };
00340         if (s_Data.QuadIndexCount >= s_Data.MaxIndices)
00341             FlushAndReset();
00342         const float texIndex = 0.0f; // White Texture
00343         const float tilingFactor = 1.0f;
00344         for (size_t i = 0; i < quadVertexCount; i++)
00345         {
00346             s_Data.QuadVertexBufferPtr->Position = transform * s_Data.QuadVertexPositions[i];
00347             s_Data.QuadVertexBufferPtr->Color = color;
00348             s_Data.QuadVertexBufferPtr->TexCoord = texCoords[i];
00349             s_Data.QuadVertexBufferPtr->TexIndex = texIndex;
00350             s_Data.QuadVertexBufferPtr->TilingFactor = tilingFactor;
00351             s_Data.QuadVertexBufferPtr++;
00352         }
00353         s_Data.QuadIndexCount += 6;
00354         s_Data.Stats.QuadCount++;
00355     }
```

References FlushAndReset(), Vesper::Renderer2DData::MaxIndices, Vesper::Renderer2D::Statistics::QuadCount, Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBufferPtr, Vesper::s_Data, Vesper::Renderer2DData::Stats, Vesper::QuadVertex::TexIndex, and Vesper::QuadVertex::TilingFactor.

9.56.1.8 DrawQuadRotated() [2/3]

```
void Vesper::Renderer2D::DrawQuadRotated (
            const glm::vec2 & position,
            const glm::vec2 & size,
            float rotationRads,
            const glm::vec4 & color)  [static]
```

```
00357     {
00358         DrawQuadRotated({ position.x, position.y, 0.0f }, size, rotationRads, color);
00359     }
```

9.56.1.9 DrawQuadRotated() [3/3]

```
void Vesper::Renderer2D::DrawQuadRotated (
            const glm::vec3 & position,
            const glm::vec2 & size,
            float rotationRads,
            const glm::vec4 & color)  [static]
```

```
00361     {
00362         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position)
00363             * glm::rotate(glm::mat4(1.0f), rotationRads, { 0.0f, 0.0f, 1.0f })
00364             * glm::scale(glm::mat4(1.0f), { size.x, size.y, 1.0f });
00365         DrawQuadRotated(transform, color);
00366     }
```

9.56.1.10 DrawQuadRotatedWithTexture() [1/6]

```
void Vesper::Renderer2D::DrawQuadRotatedWithTexture (
            const glm::mat4 & transform,
            const Ref< SubTexture2D > & subtexture,
            float tilingFactor,
            const glm::vec4 tintColor)  [static]
```

```
00423     {
00424         VZ_PROFILE_FUNCTION();
00425         constexpr size_t quadVertexCount = 4;
00426         const glm::vec2* texCoords = subtexture->GetTexCoords();
00427         const Ref<Texture2D> texture = subtexture->GetTexture();
00428         if (s_Data.QuadIndexCount >= s_Data.MaxIndices)
00429             FlushAndReset();
00430         float textureIndex = 0.0f;
00431         for (uint32_t i = 1; i < s_Data.TextureSlotIndex; i++)
00432         {
00433             if (*s_Data.TextureSlots[i].get() == *texture.get())
00434             {
00435                 textureIndex = (float)i;
00436                 break;
00437             }
00438         }
00439         if (textureIndex == 0.0f)
00440         {
00441             if (s_Data.TextureSlotIndex >= s_Data.MaxTextureSlots)
00442                 FlushAndReset();
00443             textureIndex = (float)s_Data.TextureSlotIndex;
00444             s_Data.TextureSlots[s_Data.TextureSlotIndex] = texture;
00445             s_Data.TextureSlotIndex++;
00446         }
00447         for (size_t i = 0; i < quadVertexCount; i++)
00448         {
00449             s_Data.QuadVertexBufferPtr->Position = transform * s_Data.QuadVertexPositions[i];
00450             s_Data.QuadVertexBufferPtr->Color = tintColor;
00451             s_Data.QuadVertexBufferPtr->TexCoord = texCoords[i];
00452             s_Data.QuadVertexBufferPtr->TexIndex = textureIndex;
00453             s_Data.QuadVertexBufferPtr->TilingFactor = tilingFactor;
00454             s_Data.QuadVertexBufferPtr++;
00455         }
00456         s_Data.QuadIndexCount += 6;
00457         s_Data.Stats.QuadCount++;
00458
00459     }
```

References FlushAndReset(), Vesper::Renderer2DData::MaxIndices, Vesper::Renderer2DData::MaxTextureSlots, Vesper::Renderer2D::Statistics::QuadCount, Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBuff Vesper::s_Data, Vesper::Renderer2DData::Stats, Vesper::QuadVertex::TexIndex, Vesper::Renderer2DData::TextureSlotIndex, and Vesper::QuadVertex::TilingFactor.

9.56.1.11 DrawQuadRotatedWithTexture() [2/6]

```
void Vesper::Renderer2D::DrawQuadRotatedWithTexture (
            const glm::mat4 & transform,
            const Ref< Texture2D > & texture,
            float tilingFactor,
            const glm::vec4 tintColor)  [static]
```

```
00369     {
00370         VZ_PROFILE_FUNCTION();
00371         constexpr size_t quadVertexCount = 4;
00372         constexpr glm::vec2 texCoords[quadVertexCount] = {
00373             { 0.0f, 0.0f },
00374             { 1.0f, 0.0f },
00375             { 1.0f, 1.0f },
00376             { 0.0f, 1.0f }
00377         };
00378         if (s_Data.QuadIndexCount >= s_Data.MaxIndices)
00379             FlushAndReset();
00380         float textureIndex = 0.0f;
00381         for (uint32_t i = 1; i < s_Data.TextureSlotIndex; i++)
00382         {
00383             if (*s_Data.TextureSlots[i].get() == *texture.get())
00384             {
```

```
00385                 textureIndex = (float)i;
00386                 break;
00387             }
00388         }
00389         if (textureIndex == 0.0f)
00390         {
00391             if (s_Data.TextureSlotIndex >= s_Data.MaxTextureSlots)
00392                 FlushAndReset();
00393             textureIndex = (float)s_Data.TextureSlotIndex;
00394             s_Data.TextureSlots[s_Data.TextureSlotIndex] = texture;
00395             s_Data.TextureSlotIndex++;
00396         }
00397         for (size_t i = 0; i < quadVertexCount; i++)
00398         {
00399             s_Data.QuadVertexBufferPtr->Position = transform * s_Data.QuadVertexPositions[i];
00400             s_Data.QuadVertexBufferPtr->Color = tintColor;
00401             s_Data.QuadVertexBufferPtr->TexCoord = texCoords[i];
00402             s_Data.QuadVertexBufferPtr->TexIndex = textureIndex;
00403             s_Data.QuadVertexBufferPtr->TilingFactor = tilingFactor;
00404             s_Data.QuadVertexBufferPtr++;
00405         }
00406         s_Data.QuadIndexCount += 6;
00407         s_Data.Stats.QuadCount++;
00408     }
```

References FlushAndReset(), Vesper::Renderer2DData::MaxIndices, Vesper::Renderer2DData::MaxTextureSlots, Vesper::Renderer2D::Statistics::QuadCount, Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBuff Vesper::s_Data, Vesper::Renderer2DData::Stats, Vesper::QuadVertex::TexIndex, Vesper::Renderer2DData::TextureSlotIndex, and Vesper::QuadVertex::TilingFactor.

9.56.1.12 DrawQuadRotatedWithTexture() [3/6]

```
void Vesper::Renderer2D::DrawQuadRotatedWithTexture (
              const glm::vec2 & position,
              const glm::vec2 & size,
              const Ref< SubTexture2D > & subtexture,
              float rotationRads,
              float tilingFactor,
              const glm::vec4 tintColor)  [static]
00461     {
00462         DrawQuadRotatedWithTexture({ position.x, position.y, 0.0f }, size, subtexture, rotationRads,
      tilingFactor, tintColor);
00463     }
```

9.56.1.13 DrawQuadRotatedWithTexture() [4/6]

```
void Vesper::Renderer2D::DrawQuadRotatedWithTexture (
              const glm::vec2 & position,
              const glm::vec2 & size,
              const Ref< Texture2D > & texture,
              float rotationRads,
              float tilingFactor,
              const glm::vec4 tintColor)  [static]
00410     {
00411         DrawQuadRotatedWithTexture({ position.x, position.y, 0.0f }, size, texture, rotationRads,
      tilingFactor, tintColor);
00412     }
```

9.56.1.14 DrawQuadRotatedWithTexture() [5/6]

```
void Vesper::Renderer2D::DrawQuadRotatedWithTexture (
              const glm::vec3 & position,
              const glm::vec2 & size,
```

```
            const Ref< SubTexture2D > & subtexture,
            float rotationRads,
            float tilingFactor,
            const glm::vec4 tintColor)  [static]
00465     {
00466         VZ_PROFILE_FUNCTION();
00467 
00468         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position)
00469             * glm::rotate(glm::mat4(1.0f), rotationRads, { 0.0f, 0.0f, 1.0f })
00470             * glm::scale(glm::mat4(1.0f), { size.x, size.y, 1.0f });
00471 
00472         DrawQuadRotatedWithTexture(transform, subtexture, tilingFactor, tintColor);
00473     }
```

9.56.1.15 DrawQuadRotatedWithTexture() [6/6]

```
void Vesper::Renderer2D::DrawQuadRotatedWithTexture (
            const glm::vec3 & position,
            const glm::vec2 & size,
            const Ref< Texture2D > & texture,
            float rotationRads,
            float tilingFactor,
            const glm::vec4 tintColor)  [static]
00414     {
00415         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position)
00416             * glm::rotate(glm::mat4(1.0f), rotationRads, { 0.0f, 0.0f, 1.0f })
00417             * glm::scale(glm::mat4(1.0f), { size.x, size.y, 1.0f });
00418 
00419         DrawQuadRotatedWithTexture(transform, texture, tilingFactor, tintColor);
00420     }
```

9.56.1.16 DrawQuadWithTexture() [1/6]

```
void Vesper::Renderer2D::DrawQuadWithTexture (
            const glm::mat4 & transform,
            const Ref< SubTexture2D > & subtexture,
            float tilingFactor,
            const glm::vec4 tintColor)  [static]
00275     {
00276         VZ_PROFILE_FUNCTION();
00277 
00278         constexpr size_t quadVertexCount = 4;
00279         const glm::vec2* texCoords = subtexture->GetTexCoords();
00280         const Ref<Texture2D> texture = subtexture->GetTexture();
00281 
00282         if (s_Data.QuadIndexCount >= s_Data.MaxIndices)
00283             FlushAndReset();
00284 
00285         float textureIndex = 0.0f;
00286         for (uint32_t i = 1; i < s_Data.TextureSlotIndex; i++)
00287         {
00288             if (*s_Data.TextureSlots[i].get() == *texture.get())
00289             {
00290                 textureIndex = (float)i;
00291                 break;
00292             }
00293         }
00294 
00295         if (textureIndex == 0.0f)
00296         {
00297             if (s_Data.TextureSlotIndex >= s_Data.MaxTextureSlots)
00298                 FlushAndReset();
00299 
00300             textureIndex = (float)s_Data.TextureSlotIndex;
00301             s_Data.TextureSlots[s_Data.TextureSlotIndex] = texture;
00302             s_Data.TextureSlotIndex++;
00303         }
00304 
00305         for (size_t i = 0; i < quadVertexCount; i++)
00306         {
```

```
00307             s_Data.QuadVertexBufferPtr->Position = transform * s_Data.QuadVertexPositions[i];
00308             s_Data.QuadVertexBufferPtr->Color = tintColor;
00309             s_Data.QuadVertexBufferPtr->TexCoord = texCoords[i];
00310             s_Data.QuadVertexBufferPtr->TexIndex = textureIndex;
00311             s_Data.QuadVertexBufferPtr->TilingFactor = tilingFactor;
00312             s_Data.QuadVertexBufferPtr++;
00313         }
00314
00315         s_Data.QuadIndexCount += 6;
00316         s_Data.Stats.QuadCount++;
00317     }
```

References FlushAndReset(), Vesper::Renderer2DData::MaxIndices, Vesper::Renderer2DData::MaxTextureSlots, Vesper::Renderer2D::Statistics::QuadCount, Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBuff Vesper::s_Data, Vesper::Renderer2DData::Stats, Vesper::QuadVertex::TexIndex, Vesper::Renderer2DData::TextureSlotIndex, and Vesper::QuadVertex::TilingFactor.

9.56.1.17 DrawQuadWithTexture() [2/6]

```
void Vesper::Renderer2D::DrawQuadWithTexture (
             const glm::mat4 & transform,
             const Ref< Texture2D > & texture,
             float tilingFactor,
             const glm::vec4 tintColor)  [static]
```

```
00219     {
00220         VZ_PROFILE_FUNCTION();
00221         constexpr size_t quadVertexCount = 4;
00222         constexpr glm::vec4 color = { 1.0f, 1.0f, 1.0f, 1.0f };
00223         constexpr glm::vec2 texCoords[quadVertexCount] = {
00224             { 0.0f, 0.0f },
00225             { 1.0f, 0.0f },
00226             { 1.0f, 1.0f },
00227             { 0.0f, 1.0f }
00228         };
00229         if (s_Data.QuadIndexCount >= s_Data.MaxIndices)
00230             FlushAndReset();
00231         float textureIndex = 0.0f;
00232         for (uint32_t i = 1; i < s_Data.TextureSlotIndex; i++)
00233         {
00234             if (*s_Data.TextureSlots[i].get() == *texture.get())
00235             {
00236                 textureIndex = (float)i;
00237                 break;
00238             }
00239         }
00240         if (textureIndex == 0.0f)
00241         {
00242             if (s_Data.TextureSlotIndex >= s_Data.MaxTextureSlots)
00243                 FlushAndReset();
00244             textureIndex = (float)s_Data.TextureSlotIndex;
00245             s_Data.TextureSlots[s_Data.TextureSlotIndex] = texture;
00246             s_Data.TextureSlotIndex++;
00247         }
00248         for (size_t i = 0; i < quadVertexCount; i++)
00249         {
00250             s_Data.QuadVertexBufferPtr->Position = transform * s_Data.QuadVertexPositions[i];
00251             s_Data.QuadVertexBufferPtr->Color = tintColor;
00252             s_Data.QuadVertexBufferPtr->TexCoord = texCoords[i];
00253             s_Data.QuadVertexBufferPtr->TexIndex = textureIndex;
00254             s_Data.QuadVertexBufferPtr->TilingFactor = tilingFactor;
00255             s_Data.QuadVertexBufferPtr++;
00256         }
00257
00258         s_Data.QuadIndexCount += 6;
00259         s_Data.Stats.QuadCount++;
00260
00261     }
```

References FlushAndReset(), Vesper::Renderer2DData::MaxIndices, Vesper::Renderer2DData::MaxTextureSlots, Vesper::Renderer2D::Statistics::QuadCount, Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBuff Vesper::s_Data, Vesper::Renderer2DData::Stats, Vesper::QuadVertex::TexIndex, Vesper::Renderer2DData::TextureSlotIndex, and Vesper::QuadVertex::TilingFactor.

9.56.1.18 DrawQuadWithTexture() [3/6]

```
void Vesper::Renderer2D::DrawQuadWithTexture (
              const glm::vec2 & position,
              const glm::vec2 & size,
              const Ref< SubTexture2D > & subtexture,
              float tilingFactor,
              const glm::vec4 tintColor)  [static]
```

```
00319     {
00320         DrawQuadWithTexture({ position.x, position.y, 0.0f }, size, subtexture, tilingFactor,
      tintColor);
00321     }
```

9.56.1.19 DrawQuadWithTexture() [4/6]

```
void Vesper::Renderer2D::DrawQuadWithTexture (
              const glm::vec2 & position,
              const glm::vec2 & size,
              const Ref< Texture2D > & texture,
              float tilingFactor,
              const glm::vec4 tintColor)  [static]
```

```
00263     {
00264         DrawQuadWithTexture({ position.x, position.y, 0.0f }, size, texture, tilingFactor, tintColor);
00265     }
```

9.56.1.20 DrawQuadWithTexture() [5/6]

```
void Vesper::Renderer2D::DrawQuadWithTexture (
              const glm::vec3 & position,
              const glm::vec2 & size,
              const Ref< SubTexture2D > & subtexture,
              float tilingFactor,
              const glm::vec4 tintColor)  [static]
```

```
00323     {
00324         VZ_PROFILE_FUNCTION();
00325
00326         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position) * glm::scale(glm::mat4(1.0f),
      { size.x, size.y, 1.0f });
00327         DrawQuadWithTexture(transform, subtexture, tilingFactor, tintColor);
00328     }
```

9.56.1.21 DrawQuadWithTexture() [6/6]

```
void Vesper::Renderer2D::DrawQuadWithTexture (
              const glm::vec3 & position,
              const glm::vec2 & size,
              const Ref< Texture2D > & texture,
              float tilingFactor,
              const glm::vec4 tintColor)  [static]
```

```
00267     {
00268         VZ_PROFILE_FUNCTION();
00269
00270         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position) * glm::scale(glm::mat4(1.0f),
      { size.x, size.y, 1.0f });
00271         DrawQuadWithTexture(transform, texture, tilingFactor, tintColor);
00272     }
```

9.56.1.22 EndScene()

```
void Vesper::Renderer2D::EndScene ()  [static]
```

```
00161     {
00162         VZ_PROFILE_FUNCTION();
00163         uint32_t dataSize = (uint32_t)((uint8_t*)s_Data.QuadVertexBufferPtr -
      (uint8_t*)s_Data.QuadVertexBufferBase);
00164
00165         s_Data.QuadVertexBuffer->SetData(s_Data.QuadVertexBufferBase, dataSize);
00166         Flush();
00167     }
```

References Flush(), Vesper::Renderer2DData::QuadVertexBufferBase, Vesper::Renderer2DData::QuadVertexBufferPtr, and Vesper::s_Data.

Referenced by FlushAndReset(), Vesper::EditorLayer::OnUpdate(), Vesper::Scene::OnUpdateEditor(), and Vesper::Scene::OnUpdateRuntime().

9.56.1.23 Flush()

```
void Vesper::Renderer2D::Flush ()  [static]
```

```
00170     {
00171         VZ_PROFILE_FUNCTION();
00172         // Bind textures
00173         for (uint32_t i = 0; i < s_Data.TextureSlotIndex; i++)
00174             s_Data.TextureSlots[i]->Bind(i);
00175
00176         RenderCommand::DrawIndexed(s_Data.QuadVertexArray, s_Data.QuadIndexCount);
00177         s_Data.Stats.DrawCalls++;
00178     }
```

References Vesper::Renderer2D::Statistics::DrawCalls, Vesper::Renderer2DData::QuadIndexCount, Vesper::s_Data, Vesper::Renderer2DData::Stats, and Vesper::Renderer2DData::TextureSlotIndex.

Referenced by EndScene().

9.56.1.24 FlushAndReset()

```
void Vesper::Renderer2D::FlushAndReset ()  [static], [private]
```

```
00508     {
00509         EndScene();
00510         s_Data.QuadIndexCount = 0;
00511         s_Data.QuadVertexBufferPtr = s_Data.QuadVertexBufferBase;
00512         s_Data.TextureSlotIndex = 1;
00513     }
```

References EndScene(), Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBufferBase, Vesper::Renderer2DData::QuadVertexBufferPtr, Vesper::s_Data, and Vesper::Renderer2DData::TextureSlotIndex.

Referenced by DrawQuad(), DrawQuadRotated(), DrawQuadRotatedWithTexture(), DrawQuadRotatedWithTexture(), DrawQuadWithTexture(), and DrawQuadWithTexture().

9.56.1.25 GetStats()

```
Renderer2D::Statistics Vesper::Renderer2D::GetStats ()  [static]
```

```
00503     {
00504         return s_Data.Stats;
00505     }
```

References Vesper::s_Data, and Vesper::Renderer2DData::Stats.

Referenced by Vesper::EditorLayer::OnImGuiRender().

9.56.1.26 GetWhiteTexture()

```
Ref< Texture2D > Vesper::Renderer2D::GetWhiteTexture ()  [static]
```

```
00493     {
00494         return s_Data.WhiteTexture;
00495     }
```

References Vesper::s_Data.

9.56.1.27 Init()

```
void Vesper::Renderer2D::Init ()  [static]
```

```
00056     {
00057         VZ_PROFILE_FUNCTION();
00058 
00059         s_Data.QuadVertexArray = (VertexArray::Create());
00060 
00061 
00062         s_Data.QuadVertexBuffer = VertexBuffer::Create(s_Data.MaxVertices * sizeof(QuadVertex));
00063 
00064         s_Data.QuadVertexBuffer->SetLayout({
00065             { ShaderDataType::Float3, "a_Position"  },
00066             { ShaderDataType::Float4, "a_Color"     },
00067             { ShaderDataType::Float2, "a_TexCoord"    },
00068             { ShaderDataType::Float,  "a_TexIndex"    },
00069             { ShaderDataType::Float,  "a_TilingFactor"}
00070             });
00071         s_Data.QuadVertexArray->AddVertexBuffer(s_Data.QuadVertexBuffer);
00072 
00073         s_Data.QuadVertexBufferBase = new QuadVertex[s_Data.MaxVertices];
00074 
00075         uint32_t* quadIndices = new uint32_t[s_Data.MaxIndices];
00076         uint32_t offset = 0;
00077 
00078         for (uint32_t i = 0; i < s_Data.MaxIndices; i += 6) {
00079             quadIndices[i + 0] = offset + 0;
00080             quadIndices[i + 1] = offset + 1;
00081             quadIndices[i + 2] = offset + 2;
00082             quadIndices[i + 3] = offset + 2;
00083             quadIndices[i + 4] = offset + 3;
00084             quadIndices[i + 5] = offset + 0;
00085 
00086             offset += 4;
00087         }
00088 
00089         Ref<IndexBuffer> quadIB;
00090         quadIB = (IndexBuffer::Create(quadIndices, s_Data.MaxIndices));
00091         s_Data.QuadVertexArray->SetIndexBuffer(quadIB);
00092         delete[] quadIndices;
00093 
00094         s_Data.WhiteTexture = Texture2D::Create(1, 1);
00095         uint32_t whiteTextureData = 0xffffffff;
00096         s_Data.WhiteTexture->SetData(&whiteTextureData, sizeof(uint32_t));
00097 
00098         int32_t samplers[s_Data.MaxTextureSlots];
00099         for (uint32_t i = 0; i < s_Data.MaxTextureSlots; i++)
00100             samplers[i] = i;
00101 
00102 
00103         s_Data.TextureShader = Shader::Create("../../Vesper-Editor/assets/shaders/Texture.glsl");
00104         s_Data.TextureShader->Bind();
00105         s_Data.TextureShader->SetIntArray("u_Textures", samplers, s_Data.MaxTextureSlots);
00106 
00107         s_Data.TextureSlots[0] = s_Data.WhiteTexture;
00108 
00109         s_Data.QuadVertexPositions[0] = { -0.5f, -0.5f, 0.0f, 1.0f };
00110         s_Data.QuadVertexPositions[1] = { 0.5f, -0.5f, 0.0f, 1.0f };
00111         s_Data.QuadVertexPositions[2] = { 0.5f,  0.5f, 0.0f, 1.0f };
00112         s_Data.QuadVertexPositions[3] = { -0.5f,  0.5f, 0.0f, 1.0f };
00113 
00114     }
```

References Vesper::Float, Vesper::Float2, Vesper::Float3, Vesper::Float4, Vesper::Renderer2DData::MaxIndices, Vesper::Renderer2DData::MaxTextureSlots, Vesper::Renderer2DData::MaxVertices, Vesper::Renderer2DData::QuadVertexBufferBas and Vesper::s_Data.

Referenced by Vesper::Renderer::Init().

9.56.1.28 ResetStats()

```
void Vesper::Renderer2D::ResetStats ()  [static]
```

```
00498     {
00499         memset(&s_Data.Stats, 0, sizeof(Statistics));
00500     }
```

References Vesper::s_Data, and Vesper::Renderer2DData::Stats.

Referenced by Vesper::EditorLayer::OnUpdate().

9.56.1.29 Shutdown()

```
void Vesper::Renderer2D::Shutdown ()  [static]
```

```
00117     {
00118         VZ_PROFILE_FUNCTION();
00119
00120         delete[] s_Data.QuadVertexBufferBase;
00121     }
```

References Vesper::Renderer2DData::QuadVertexBufferBase, and Vesper::s_Data.

9.56.1.30 StartBatch()

```
void Vesper::Renderer2D::StartBatch ()  [static], [private]
```

```
00516     {
00517
00518         s_Data.QuadIndexCount = 0;
00519         s_Data.QuadVertexBufferPtr = s_Data.QuadVertexBufferBase;
00520
00521         //s_Data.CircleIndexCount = 0;
00522         //s_Data.CircleVertexBufferPtr = s_Data.CircleVertexBufferBase;
00523
00524         //s_Data.LineVertexCount = 0;
00525         //s_Data.LineVertexBufferPtr = s_Data.LineVertexBufferBase;
00526
00527         //s_Data.TextIndexCount = 0;
00528         //s_Data.TextVertexBufferPtr = s_Data.TextVertexBufferBase;
00529
00530         s_Data.TextureSlotIndex = 1;
00531
00532     }
```

References Vesper::Renderer2DData::QuadIndexCount, Vesper::Renderer2DData::QuadVertexBufferBase, Vesper::Renderer2DData::QuadVertexBufferPtr, Vesper::s_Data, and Vesper::Renderer2DData::TextureSlotIndex.

Referenced by BeginScene().

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

- Vesper/src/Vesper/Renderer/Renderer2D.h
- Vesper/src/Vesper/Renderer/Renderer2D.cpp

9.57 Vesper::Renderer2DData Struct Reference

Classes

- struct CameraData

Public Attributes

- Ref< VertexArray > QuadVertexArray
- Ref< VertexBuffer > QuadVertexBuffer
- Ref< Shader > TextureShader
- Ref< Texture2D > WhiteTexture
- uint32_t QuadIndexCount = 0
- QuadVertex * QuadVertexBufferBase = nullptr
- QuadVertex * QuadVertexBufferPtr = nullptr
- std::array< Ref< Texture2D >, MaxTextureSlots > TextureSlots
- uint32_t TextureSlotIndex = 1
- glm::vec4 QuadVertexPositions [4]
- Renderer2D::Statistics Stats
- CameraData CameraBuffer
- Ref< UniformBuffer > CameraUniformBuffer

Static Public Attributes

- static const uint32_t MaxQuads = 30000
- static const uint32_t MaxVertices = MaxQuads * 4
- static const uint32_t MaxIndices = MaxQuads * 6
- static const uint32_t MaxTextureSlots = 32

9.57.1 Class Documentation

9.57.1.1 struct Vesper::Renderer2DData::CameraData

Class Members

mat4	ViewProjection	

9.57.2 Member Data Documentation

9.57.2.1 CameraBuffer

```
CameraData Vesper::Renderer2DData::CameraBuffer
```

9.57.2.2 CameraUniformBuffer

```
Ref< UniformBuffer > Vesper::Renderer2DData::CameraUniformBuffer
```

9.57.2.3 MaxIndices

```
const uint32_t Vesper::Renderer2DData::MaxIndices = MaxQuads * 6  [static]
```

Referenced by Vesper::Renderer2D::DrawQuad(), Vesper::Renderer2D::DrawQuadRotated(), Vesper::Renderer2D::DrawQuadRotate
Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::DrawQu
and Vesper::Renderer2D::Init().

9.57.2.4 MaxQuads

```
const uint32_t Vesper::Renderer2DData::MaxQuads = 30000  [static]
```

9.57.2.5 MaxTextureSlots

```
const uint32_t Vesper::Renderer2DData::MaxTextureSlots = 32  [static]
```

Referenced by Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), and Vesper::Renderer2D::Init().

9.57.2.6 MaxVertices

```
const uint32_t Vesper::Renderer2DData::MaxVertices = MaxQuads * 4  [static]
```

Referenced by Vesper::Renderer2D::Init().

9.57.2.7 QuadIndexCount

```
uint32_t Vesper::Renderer2DData::QuadIndexCount = 0
```

Referenced by Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::DrawQuad(), Vesper::Renderer2D::DrawQuadRotated(), Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadR Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::Flush(), Vesper::Renderer2D::FlushAndReset(), and Vesper::Renderer2D::StartBatch().

9.57.2.8 QuadVertexArray

```
Ref< VertexArray > Vesper::Renderer2DData::QuadVertexArray
```

9.57.2.9 QuadVertexBuffer

```
Ref< VertexBuffer > Vesper::Renderer2DData::QuadVertexBuffer
```

9.57.2.10 QuadVertexBufferBase

```
QuadVertex * Vesper::Renderer2DData::QuadVertexBufferBase = nullptr
```

Referenced by Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::EndScene(), Vesper::Renderer2D::FlushAndReset(), Vesper::Renderer2D::Init(), Vesper::Renderer2D::Shutdown(), and Vesper::Renderer2D::StartBatch().

9.57.2.11 QuadVertexBufferPtr

```
QuadVertex * Vesper::Renderer2DData::QuadVertexBufferPtr = nullptr
```

Referenced by Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::DrawQuad(), Vesper::Renderer2D::DrawQuadRotated(), Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadR Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::EndScene(), Vesper::Renderer2D::FlushAndReset(), and Vesper::Renderer2D::StartBatch().

9.57.2.12 QuadVertexPositions

```
glm::vec4 Vesper::Renderer2DData::QuadVertexPositions
```

9.57.2.13 Stats

```
Renderer2D::Statistics Vesper::Renderer2DData::Stats
```

Referenced by Vesper::Renderer2D::DrawQuad(), Vesper::Renderer2D::DrawQuadRotated(), Vesper::Renderer2D::DrawQuadRotate Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::DrawQu Vesper::Renderer2D::Flush(), Vesper::Renderer2D::GetStats(), and Vesper::Renderer2D::ResetStats().

9.57.2.14 TextureShader

```
Ref< Shader > Vesper::Renderer2DData::TextureShader
```

9.57.2.15 TextureSlotIndex

```
uint32_t Vesper::Renderer2DData::TextureSlotIndex = 1
```

Referenced by Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::BeginScene(), Vesper::Renderer2D::DrawQuadRotatedWit Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::DrawQu Vesper::Renderer2D::Flush(), Vesper::Renderer2D::FlushAndReset(), and Vesper::Renderer2D::StartBatch().

9.57.2.16 TextureSlots

```
std::array< Ref< Texture2D >, MaxTextureSlots > Vesper::Renderer2DData::TextureSlots
```

9.57.2.17 WhiteTexture

```
Ref< Texture2D > Vesper::Renderer2DData::WhiteTexture
```

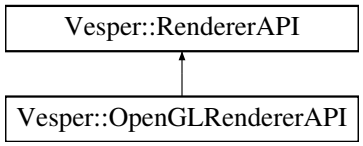
The documentation for this struct was generated from the following file:

- Vesper/src/Vesper/Renderer/Renderer2D.cpp

9.58 Vesper::RendererAPI Class Reference

`#include <RendererAPI.h>`

Inheritance diagram for Vesper::RendererAPI:



Public Types

- enum class API { None = 0 , OpenGL = 1 }

Public Member Functions

- virtual ~RendererAPI ()=default
- virtual void Init ()=0
- virtual void SetViewport (uint32_t x, uint32_t y, uint32_t width, uint32_t height)=0
- virtual void SetClearColor (const glm::vec4 &color)=0
- virtual void Clear ()=0
- virtual void DrawIndexed (const Ref< VertexArray > &vertexArray, uint32_t indexCount=0)=0

Static Public Member Functions

- static API GetAPI ()

Static Private Attributes

- static API s_API = RendererAPI::API::OpenGL

9.58.1 Member Enumeration Documentation

9.58.1.1 API

`enum class Vesper::RendererAPI::API [strong]`

Enumerator

None	
OpenGL	

```
00011                       {
00012             None = 0,
00013             OpenGL = 1,
00014         };
```

9.58.2 Constructor & Destructor Documentation

9.58.2.1 ~RendererAPI()

```
virtual Vesper::RendererAPI::~RendererAPI ()  [virtual], [default]
```

9.58.3 Member Function Documentation

9.58.3.1 Clear()

```
virtual void Vesper::RendererAPI::Clear ()  [pure virtual]
```

Implemented in Vesper::OpenGLRendererAPI.

Referenced by Vesper::RenderCommand::Clear().

9.58.3.2 DrawIndexed()

```
virtual void Vesper::RendererAPI::DrawIndexed (
            const Ref< VertexArray > & vertexArray,
            uint32_t indexCount = 0)  [pure virtual]
```

Implemented in Vesper::OpenGLRendererAPI.

9.58.3.3 GetAPI()

```
API Vesper::RendererAPI::GetAPI ()  [inline], [static]
00025 { return s_API; }
```

References s_API.

Referenced by Vesper::Renderer::GetAPI().

9.58.3.4 Init()

```
virtual void Vesper::RendererAPI::Init ()  [pure virtual]
```

Implemented in Vesper::OpenGLRendererAPI.

Referenced by Vesper::RenderCommand::Init().

9.58.3.5 SetClearColor()

```
virtual void Vesper::RendererAPI::SetClearColor (
            const glm::vec4 & color)  [pure virtual]
```

Implemented in Vesper::OpenGLRendererAPI.

9.58.3.6 SetViewport()

```
virtual void Vesper::RendererAPI::SetViewport (
            uint32_t x,
            uint32_t y,
            uint32_t width,
            uint32_t height)  [pure virtual]
```

Implemented in Vesper::OpenGLRendererAPI.

Referenced by Vesper::RenderCommand::SetViewport().

9.58.4 Member Data Documentation

9.58.4.1 s_API

```
RendererAPI::API Vesper::RendererAPI::s_API = RendererAPI::API::OpenGL  [static], [private]
```

Referenced by GetAPI().

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

- Vesper/src/Vesper/Renderer/RendererAPI.h
- Vesper/src/Vesper/Renderer/RendererAPI.cpp

9.59 Vesper::Scene Class Reference

```
#include <Scene.h>
```

Public Member Functions

- Scene ()
- Scene (const std::string &name)
- ~Scene ()
- Entity CreateEntity (const std::string &name=std::string())
- Entity CreateEntity (const std::string &name, const std::string &uuid)
- void DestroyEntity (Entity entity)
- void OnUpdateRuntime (Timestep ts)
- void OnUpdateEditor (Timestep ts, EditorCamera &camera)
- void OnViewportResize (uint32_t width, uint32_t height)
- Entity GetPrimaryCameraEntity ()

Private Member Functions

- template<typename T>
  void OnComponentAdded (Entity entity, T &component)
- void SetName (const std::string &name)
- const std::string & GetName () const
- template<> void OnComponentAdded (Entity entity, UUIDComponent &component)
- template<> void OnComponentAdded (Entity entity, NameComponent &component)
- template<> void OnComponentAdded (Entity entity, TransformComponent &component)
- template<> void OnComponentAdded (Entity entity, CameraComponent &component)
- template<> void OnComponentAdded (Entity entity, SpriteRendererComponent &component)
- template<> void OnComponentAdded (Entity entity, SubTextureComponent &component)
- template<> void OnComponentAdded (Entity entity, TextureAnimationComponent &component)
- template<> void OnComponentAdded (Entity entity, NativeScriptComponent &component)
- template<> void OnComponentAdded (Entity entity, UUIDComponent &component)
- template<> void OnComponentAdded (Entity entity, NameComponent &component)
- template<> void OnComponentAdded (Entity entity, TransformComponent &component)
- template<> void OnComponentAdded (Entity entity, CameraComponent &component)
- template<> void OnComponentAdded (Entity entity, SpriteRendererComponent &component)
- template<> void OnComponentAdded (Entity entity, SubTextureComponent &component)
- template<> void OnComponentAdded (Entity entity, TextureAnimationComponent &component)
- template<> void OnComponentAdded (Entity entity, NativeScriptComponent &component)

Private Attributes

- std::string m_Name
- entt::registry m_Registry
- uint32_t m_ViewportWidth = 160
- uint32_t m_ViewportHeight = 90

Friends

- class Entity
- class SceneSerializer
- class SceneHierarchyPanel

9.59.1 Constructor & Destructor Documentation

9.59.1.1 Scene() [1/2]

```
Vesper::Scene::Scene ()
00012         : m_Name("Untitled Scene")
00013     {
00014         VZ_PROFILE_FUNCTION();
00015
00016     }
```

References Scene().

Referenced by Scene().

9.59.1.2 Scene() [2/2]

```
Vesper::Scene::Scene (
            const std::string & name)
00019         : m_Name(name)
00020     {
00021     }
```

9.59.1.3 ~Scene()

```
Vesper::Scene::~Scene ()
00024     {
00025     }
```

9.59.2 Member Function Documentation

9.59.2.1 CreateEntity() [1/2]

```
Entity Vesper::Scene::CreateEntity (
            const std::string & name,
            const std::string & uuid)
00039     {
00040         Entity entity = { m_Registry.create(), this };
00041         entity.AddComponent<TransformComponent>();
00042         auto& nameTag = entity.AddComponent<NameComponent>();
00043         nameTag.Name = name.empty() ? "Entity" + std::to_string(static_cast<std::uint32_t>(entity)) :
      name;
00044         entity.AddComponent<UUIDComponent>(uuid);
00045         return entity;
00046
00047     }
```

9.59.2.2 CreateEntity() [2/2]

```
Entity Vesper::Scene::CreateEntity (
            const std::string & name = std::string())
00028     {
00029         Entity entity = { m_Registry.create(), this };
00030         entity.AddComponent<TransformComponent>();
00031         auto& nameTag = entity.AddComponent<NameComponent>();
00032         nameTag.Name = name.empty() ? "Entity" + std::to_string(static_cast<std::uint32_t>(entity)) :
      name;
00033         entity.AddComponent<UUIDComponent>();
00034
00035         return entity;
00036     }
```

9.59.2.3 DestroyEntity()

```
void Vesper::Scene::DestroyEntity (
            Entity entity)
00050     {
00051         m_Registry.destroy(entity);
00052     }
```

9.59.2.4 GetName()

```
const std::string & Vesper::Scene::GetName () const  [inline], [private]
00043 { return m_Name; }
```

9.59.2.5 GetPrimaryCameraEntity()

```
Entity Vesper::Scene::GetPrimaryCameraEntity ()
00209     {
00210         auto view = m_Registry.view<CameraComponent>();
00211         for (auto entity : view) {
00212 
00213             const auto& camera = view.get<CameraComponent>(entity);
00214             if (camera.Primary)
00215                 return Entity{ entity, this };
00216         }
00217         return {};
00218     }
```

9.59.2.6 OnComponentAdded() [1/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
             Entity entity,
             CameraComponent & component)  [private]
00244                                                                                 {
00245         component.Camera.SetViewportSize(m_ViewportWidth, m_ViewportHeight);
00246     }
```

References m_ViewportHeight, and m_ViewportWidth.

9.59.2.7 OnComponentAdded() [2/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
             Entity entity,
             CameraComponent & component)  [private]
00244                                                                                 {
00245         component.Camera.SetViewportSize(m_ViewportWidth, m_ViewportHeight);
00246     }
```

9.59.2.8 OnComponentAdded() [3/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
             Entity entity,
             NameComponent & component)  [private]
00236                                                                                 {
00237     }
```

9.59.2.9 OnComponentAdded() [4/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
             Entity entity,
             NameComponent & component)  [private]
00236                                                                                 {
00237     }
```

9.59.2.10 OnComponentAdded() [5/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            NativeScriptComponent & component)  [private]
00265
     {
00266     }
```

9.59.2.11 OnComponentAdded() [6/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            NativeScriptComponent & component)  [private]
00265
     {
00266     }
```

9.59.2.12 OnComponentAdded() [7/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            SpriteRendererComponent & component)  [private]
00249
     {
00250     }
```

9.59.2.13 OnComponentAdded() [8/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            SpriteRendererComponent & component)  [private]
00249
     {
00250     }
```

9.59.2.14 OnComponentAdded() [9/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            SubTextureComponent & component)  [private]
00253                                                                                                                {
00254         auto& src = entity.GetOrAddComponent<SpriteRendererComponent>();
00255         if (!src.TextureEnabled) src.TextureEnabled = true;
00256         component.SetTexture(src.Texture ? src.Texture : VZ_DEFAULT_TEXTURE);
00257     }
```

9.59.2.15 OnComponentAdded() [10/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            SubTextureComponent & component)  [private]
00253                                                                                                 {
00254         auto& src = entity.GetOrAddComponent<SpriteRendererComponent>();
00255         if (!src.TextureEnabled) src.TextureEnabled = true;
00256         component.SetTexture(src.Texture ? src.Texture : VZ_DEFAULT_TEXTURE);
00257     }
```

9.59.2.16 OnComponentAdded() [11/17]

```
template<typename T>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            T & component)  [private]
00226                                                 {
00227         static_assert(false);
00228     }
```

9.59.2.17 OnComponentAdded() [12/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            TextureAnimationComponent & component)  [private]
00260
      {
00261
00262     }
```

9.59.2.18 OnComponentAdded() [13/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            TextureAnimationComponent & component)  [private]
00260
      {
00261
00262     }
```

9.59.2.19 OnComponentAdded() [14/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            TransformComponent & component)  [private]
00240                                                                                                 {
00241     }
```

9.59.2.20 OnComponentAdded() [15/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            TransformComponent & component)  [private]
00240                                                                                                {
00241     }
```

9.59.2.21 OnComponentAdded() [16/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            UUIDComponent & component)  [private]
00231                                                                                                {
00232         // TODO: search registry to ensure unique UUID
00233     }
```

9.59.2.22 OnComponentAdded() [17/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
            Entity entity,
            UUIDComponent & component)  [private]
00231                                                                                                {
00232         // TODO: search registry to ensure unique UUID
00233     }
```

9.59.2.23 OnUpdateEditor()

```
void Vesper::Scene::OnUpdateEditor (
            Timestep ts,
            EditorCamera & camera)
00151     {
00152         VZ_PROFILE_FUNCTION();
00153         Renderer2D::BeginScene(camera);
00154         auto view = m_Registry.group<SpriteRendererComponent>();
00155         for (auto entity : view)
00156         {
00157             auto& transform = m_Registry.get<TransformComponent>(entity);
00158             auto& sprite = m_Registry.get<SpriteRendererComponent>(entity);
00159             // Do not render subtextures here
00160             auto stc = m_Registry.try_get<SubTextureComponent>(entity);
00161             if (stc) {
00162                 continue;
00163             }
00164             if (sprite.TextureEnabled && !sprite.Texture)
00165                 Renderer2D::DrawQuadWithTexture(transform.GetTransform(), VZ_DEFAULT_TEXTURE,
      sprite.TilingFactor, sprite.Color);
00166             else if (sprite.TextureEnabled && sprite.Texture)
00167                 Renderer2D::DrawQuadWithTexture(transform.GetTransform(), sprite.Texture,
      sprite.TilingFactor, sprite.Color);
00168             else
00169                 Renderer2D::DrawQuad(transform.GetTransform(), sprite.Color);
00170         }
00171         auto subTextureView = m_Registry.group<SubTextureComponent>();
00172         for (auto entity : subTextureView) {
00173             auto& transform = m_Registry.get<TransformComponent>(entity);
00174             auto& sprite = m_Registry.get<SpriteRendererComponent>(entity);
00175             auto& subTexture = m_Registry.get<SubTextureComponent>(entity);
00176             if (!sprite.TextureEnabled) sprite.TextureEnabled = true;
00177             if (subTexture.SubTexture == nullptr)
00178             {
```

```
00179                 subTexture.SetTexture(VZ_DEFAULT_TEXTURE);
00180             }
00181             else {
00182                 // TODO: find way to avoid this check every frame
00183                 // Ensure the subtexture's texture matches the sprite's texture
00184                 if (sprite.Texture && subTexture.SubTexture->GetTexture() != sprite.Texture) {
00185                     subTexture.SetTexture(sprite.Texture);
00186                 }
00187             }
00188             Renderer2D::DrawQuadWithTexture(transform.GetTransform(), subTexture.GetSubTexture(),
      sprite.TilingFactor, sprite.Color);
00189         }
00190         Renderer2D::EndScene();
00191     }
```

References Vesper::Renderer2D::BeginScene(), and Vesper::Renderer2D::EndScene().

9.59.2.24 OnUpdateRuntime()

```
void Vesper::Scene::OnUpdateRuntime (
            Timestep ts)
```

TODO: Move to Scene::OnScenePlay()

```
00056     {
00057         VZ_PROFILE_FUNCTION();
00058 
00059         // Update scripts
00060         {
00061             m_Registry.view<NativeScriptComponent>().each([=](auto entity, NativeScriptComponent& nsc)
00062                 {
00064                     if (!nsc.Instance)
00065                     {
00066                         nsc.Instance = nsc.InstantiateScript();
00067                         nsc.Instance->m_Entity = Entity{ entity, this };
00068                         nsc.Instance->OnCreate();
00069                     }
00070                     nsc.Instance->OnUpdate(ts);
00071                 });
00072         }
00073 
00074         Camera* mainCamera = nullptr;
00075         glm::mat4* camTransform = nullptr;
00076         {
00077             auto view = m_Registry.view<TransformComponent, CameraComponent>();
00078             for (auto entity : view)
00079             {
00080                 auto [transform, camera] = view.get<TransformComponent, CameraComponent>(entity);
00081                 if (camera.Primary) {
00082                     mainCamera = &camera.Camera;
00083                     camTransform = &transform.GetTransform();
00084                     break;
00085                 }
00086             }
00087         }
00088         if (!mainCamera)
00089             return;
00090 
00091         Renderer2D::BeginScene(mainCamera->GetProjection(), *camTransform);
00092 
00093 
00094         auto view = m_Registry.group<SpriteRendererComponent>();
00095         for (auto entity : view)
00096         {
00097             auto& transform = m_Registry.get<TransformComponent>(entity);
00098             auto& sprite = m_Registry.get<SpriteRendererComponent>(entity);
00099 
00100             // Do not render subtextures here
00101             auto stc = m_Registry.try_get<SubTextureComponent>(entity);
00102             if (stc) {
00103                 continue;
00104             }
00105 
00106             if (sprite.TextureEnabled && !sprite.Texture)
00107                 Renderer2D::DrawQuadWithTexture(transform.GetTransform(), VZ_DEFAULT_TEXTURE,
      sprite.TilingFactor, sprite.Color);
00108             else if (sprite.TextureEnabled && sprite.Texture)
00109                 Renderer2D::DrawQuadWithTexture(transform.GetTransform(), sprite.Texture,
      sprite.TilingFactor, sprite.Color);
00110             else
00111                 Renderer2D::DrawQuad(transform.GetTransform(), sprite.Color);
```

```
00112         }
00113
00114         auto subTextureView = m_Registry.group<SubTextureComponent>();
00115         for (auto entity : subTextureView) {
00116             auto& transform = m_Registry.get<TransformComponent>(entity);
00117             auto& sprite = m_Registry.get<SpriteRendererComponent>(entity);
00118             auto& subTexture = m_Registry.get<SubTextureComponent>(entity);
00119
00120
00121             if (!sprite.TextureEnabled) sprite.TextureEnabled = true;
00122             if (subTexture.SubTexture == nullptr)
00123             {
00124                 subTexture.SetTexture(VZ_DEFAULT_TEXTURE);
00125             }
00126             else {
00127                 // Ensure the subtexture's texture matches the sprite's texture
00128                 if (sprite.Texture && subTexture.SubTexture->GetTexture() != sprite.Texture) {
00129                     subTexture.SetTexture(sprite.Texture);
00130                 }
00131             }
00132
00133             Renderer2D::DrawQuadWithTexture(transform.GetTransform(), subTexture.GetSubTexture(),
      sprite.TilingFactor, sprite.Color);
00134         }
00135
00136         //auto group1 = m_Registry.group<TextureAnimationComponent>();
00137         //for (auto entity : group1)
00138         //{
00139         //  auto& transform = m_Registry.get<TransformComponent>(entity);
00140         //  auto& texAnim = m_Registry.get<TextureAnimationComponent>(entity);
00141         //  auto& sprite = m_Registry.get<SpriteRendererComponent>(entity);
00142         //  texAnim.Update(ts.GetSeconds());
00143         //  Renderer2D::DrawQuadWithTexture(transform.GetTransform(),
      texAnim.SubTextures[texAnim.CurrentFrame], 1.0f, sprite.Color);
00144         //}
00145
00146         Renderer2D::EndScene();
00147
00148     }
```

References Vesper::Renderer2D::EndScene().

9.59.2.25 OnViewportResize()

```
void Vesper::Scene::OnViewportResize (
             uint32_t width,
             uint32_t height)
```

```
00194     {
00195         m_ViewportWidth = width;
00196         m_ViewportHeight = height;
00197
00198         // resize non fixed aspect ratio cameras
00199         auto view = m_Registry.view<CameraComponent>();
00200         for (auto entity : view)
00201         {
00202             auto& cameraComponent = view.get<CameraComponent>(entity);
00203             if (!cameraComponent.FixedAspectRatio)
00204                 cameraComponent.Camera.SetViewportSize(width, height);
00205         }
00206     }
```

References m_ViewportHeight, and m_ViewportWidth.

9.59.2.26 SetName()

```
void Vesper::Scene::SetName (
             const std::string & name)  [inline], [private]
```

TODO: friend class SceneCamera; TODO: friend class SceneRenderer;

```
00042 { m_Name = name; }
```

9.59.3 Friends And Related Symbol Documentation

9.59.3.1 Entity

```
friend class Entity  [friend]
```

9.59.3.2 SceneHierarchyPanel

```
friend class SceneHierarchyPanel  [friend]
```

9.59.3.3 SceneSerializer

```
friend class SceneSerializer  [friend]
```

9.59.4 Member Data Documentation

9.59.4.1 m_Name

```
std::string Vesper::Scene::m_Name  [private]
```

9.59.4.2 m_Registry

```
entt::registry Vesper::Scene::m_Registry  [private]
```

9.59.4.3 m_ViewportHeight

```
uint32_t Vesper::Scene::m_ViewportHeight = 90  [private]
```

Referenced by OnComponentAdded(), and OnViewportResize().

9.59.4.4 m_ViewportWidth

```
uint32_t Vesper::Scene::m_ViewportWidth = 160  [private]
```

Referenced by OnComponentAdded(), and OnViewportResize().

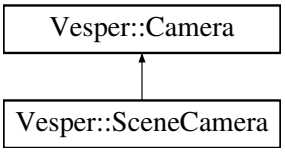
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/Scene/Scene.h
- Vesper/src/Vesper/Scene/Scene.cpp

9.60 Vesper::SceneCamera Class Reference

`#include <SceneCamera.h>`

Inheritance diagram for Vesper::SceneCamera:



Public Types

- enum class ProjectionType { Perspective = 0 , Orthographic = 1 }

Public Member Functions

- SceneCamera ()
- virtual ~SceneCamera ()=default
- void SetOrthographic (float size, float nearClip, float farClip)
- void SetPerspective (float verticalFOV, float nearClip, float farClip)
- void SetViewportSize (uint32_t width, uint32_t height)
- float GetPerspectiveVerticalFOV () const
- void SetPerspectiveVerticalFOV (float verticalFov)
- float GetPerspectiveNearClip () const
- void SetPerspectiveNearClip (float nearClip)
- float GetPerspectiveFarClip () const
- void SetPerspectiveFarClip (float farClip)
- float GetOrthographicSize () const
- void SetOrthographicSize (float size)
- float GetOrthographicNearClip () const
- void SetOrthographicNearClip (float nearClip)
- float GetOrthographicFarClip () const
- void SetOrthographicFarClip (float farClip)
- ProjectionType GetProjectionType () const
- void SetProjectionType (ProjectionType type)

Public Member Functions inherited from Vesper::Camera

- Camera ()=default
- Camera (const glm::mat4 &projection)
- ~Camera ()=default
- const glm::mat4 & GetProjection () const

Private Member Functions

- void RecalculateProjection ()

Private Attributes

- ProjectionType m_ProjectionType = ProjectionType::Orthographic
- float m_PerspectiveFOV = glm::radians(45.0f)
- float m_PerspectiveNear = 0.01f
- float m_PerspectiveFar = 1000.0f
- float m_OrthographicSize = 10.0f
- float m_OrthographicNear = -1.0f
- float m_OrthographicFar = 1.0f
- float m_AspectRatio = 0.0f

Additional Inherited Members

Protected Attributes inherited from Vesper::Camera

- glm::mat4 m_Projection = glm::mat4(1.0f)

9.60.1 Member Enumeration Documentation

9.60.1.1 ProjectionType

```
enum class Vesper::SceneCamera::ProjectionType  [strong]
```

Enumerator

Perspective	
Orthographic	

```
00009 { Perspective = 0, Orthographic = 1 };
```

9.60.2 Constructor & Destructor Documentation

9.60.2.1 SceneCamera()

```
Vesper::SceneCamera::SceneCamera ()
00010     {
00011         RecalculateProjection();
00012     }
```

References RecalculateProjection().

9.60.2.2 ~SceneCamera()

```
virtual Vesper::SceneCamera::~SceneCamera ()  [virtual], [default]
```

9.60.3 Member Function Documentation

9.60.3.1 GetOrthographicFarClip()

```
float Vesper::SceneCamera::GetOrthographicFarClip () const  [inline]
00030 { return m_OrthographicFar; }
```

References m_OrthographicFar.

9.60.3.2 GetOrthographicNearClip()

```
float Vesper::SceneCamera::GetOrthographicNearClip () const  [inline]
00028 { return m_OrthographicNear; }
```

References m_OrthographicNear.

9.60.3.3 GetOrthographicSize()

```
float Vesper::SceneCamera::GetOrthographicSize () const  [inline]
00026 { return m_OrthographicSize; }
```

References m_OrthographicSize.

9.60.3.4 GetPerspectiveFarClip()

```
float Vesper::SceneCamera::GetPerspectiveFarClip () const  [inline]
00023 { return m_PerspectiveFar; }
```

References m_PerspectiveFar.

9.60.3.5 GetPerspectiveNearClip()

```
float Vesper::SceneCamera::GetPerspectiveNearClip () const  [inline]
00021 { return m_PerspectiveNear; }
```

References m_PerspectiveNear.

9.60.3.6 GetPerspectiveVerticalFOV()

```
float Vesper::SceneCamera::GetPerspectiveVerticalFOV () const  [inline]
00019 { return m_PerspectiveFOV; }
```

9.60.3.7 GetProjectionType()

```
ProjectionType Vesper::SceneCamera::GetProjectionType () const  [inline]
00033 { return m_ProjectionType; }
```

References m_ProjectionType.

9.60.3.8 RecalculateProjection()

```
void Vesper::SceneCamera::RecalculateProjection ()  [private]
```

```
00038     {
00039         if (m_ProjectionType == ProjectionType::Perspective)
00040         {
00041             m_Projection = glm::perspective(m_PerspectiveFOV, m_AspectRatio, m_PerspectiveNear,
      m_PerspectiveFar);
00042         }
00043         else
00044         {
00045             float orthoLeft = -m_OrthographicSize * m_AspectRatio * 0.5f;
00046             float orthoRight = m_OrthographicSize * m_AspectRatio * 0.5f;
00047             float orthoBottom = -m_OrthographicSize * 0.5f;
00048             float orthoTop = m_OrthographicSize * 0.5f;
00049 
00050             m_Projection = glm::ortho(orthoLeft, orthoRight,
00051                 orthoBottom, orthoTop, m_OrthographicNear, m_OrthographicFar);
00052         }
00053 
00054     }
```

References m_AspectRatio, m_OrthographicSize, m_ProjectionType, and Perspective.

Referenced by SceneCamera(), SetOrthographic(), SetOrthographicFarClip(), SetOrthographicNearClip(), SetOrthographicSize(), SetPerspective(), SetPerspectiveFarClip(), SetPerspectiveNearClip(), SetPerspectiveVerticalFOV(), SetProjectionType(), and SetViewportSize().

9.60.3.9 SetOrthographic()

```
void Vesper::SceneCamera::SetOrthographic (
             float size,
             float nearClip,
             float farClip)
```

```
00024     {
00025         m_OrthographicSize = size;
00026         m_OrthographicNear = nearClip;
00027         m_OrthographicFar = farClip;
00028         RecalculateProjection();
00029     }
```

References m_OrthographicFar, m_OrthographicNear, m_OrthographicSize, and RecalculateProjection().

9.60.3.10 SetOrthographicFarClip()

```
void Vesper::SceneCamera::SetOrthographicFarClip (
             float farClip)  [inline]
```

```
00031 { m_OrthographicFar = farClip; RecalculateProjection(); }
```

References m_OrthographicFar, and RecalculateProjection().

9.60.3.11 SetOrthographicNearClip()

```
void Vesper::SceneCamera::SetOrthographicNearClip (
             float nearClip)  [inline]
```

```
00029 { m_OrthographicNear = nearClip; RecalculateProjection(); }
```

References m_OrthographicNear, and RecalculateProjection().

9.60.3.12 SetOrthographicSize()

```
void Vesper::SceneCamera::SetOrthographicSize (
            float size)  [inline]
00027 { m_OrthographicSize = size; RecalculateProjection(); }
```

References m_OrthographicSize, and RecalculateProjection().

9.60.3.13 SetPerspective()

```
void Vesper::SceneCamera::SetPerspective (
            float verticalFOV,
            float nearClip,
            float farClip)
00015     {
00016         m_ProjectionType = ProjectionType::Perspective;
00017         m_PerspectiveFOV = verticalFOV;
00018         m_PerspectiveNear = nearClip;
00019         m_PerspectiveFar = farClip;
00020         RecalculateProjection();
00021     }
```

References m_PerspectiveFar, m_PerspectiveNear, m_ProjectionType, Perspective, and RecalculateProjection().

9.60.3.14 SetPerspectiveFarClip()

```
void Vesper::SceneCamera::SetPerspectiveFarClip (
            float farClip)  [inline]
00024 { m_PerspectiveFar = farClip; RecalculateProjection(); }
```

References m_PerspectiveFar, and RecalculateProjection().

9.60.3.15 SetPerspectiveNearClip()

```
void Vesper::SceneCamera::SetPerspectiveNearClip (
            float nearClip)  [inline]
00022 { m_PerspectiveNear = nearClip; RecalculateProjection(); }
```

References m_PerspectiveNear, and RecalculateProjection().

9.60.3.16 SetPerspectiveVerticalFOV()

```
void Vesper::SceneCamera::SetPerspectiveVerticalFOV (
            float verticalFov)  [inline]
00020 { m_PerspectiveFOV = verticalFov; RecalculateProjection(); }
```

References RecalculateProjection().

9.60.3.17 SetProjectionType()

```
void Vesper::SceneCamera::SetProjectionType (
            ProjectionType type)  [inline]
00034 { m_ProjectionType = type; RecalculateProjection(); }
```

References m_ProjectionType, and RecalculateProjection().

9.60.3.18 SetViewportSize()

```
void Vesper::SceneCamera::SetViewportSize (
            uint32_t width,
            uint32_t height)
00032     {
00033         m_AspectRatio = (float)width / (float)height;
00034         RecalculateProjection();
00035     }
```

References m_AspectRatio, and RecalculateProjection().

9.60.4 Member Data Documentation

9.60.4.1 m_AspectRatio

```
float Vesper::SceneCamera::m_AspectRatio = 0.0f  [private]
```

Referenced by RecalculateProjection(), and SetViewportSize().

9.60.4.2 m_OrthographicFar

```
float Vesper::SceneCamera::m_OrthographicFar = 1.0f  [private]
```

Referenced by GetOrthographicFarClip(), SetOrthographic(), and SetOrthographicFarClip().

9.60.4.3 m_OrthographicNear

```
float Vesper::SceneCamera::m_OrthographicNear = -1.0f  [private]
```

Referenced by GetOrthographicNearClip(), SetOrthographic(), and SetOrthographicNearClip().

9.60.4.4 m_OrthographicSize

```
float Vesper::SceneCamera::m_OrthographicSize = 10.0f  [private]
```

Referenced by GetOrthographicSize(), RecalculateProjection(), SetOrthographic(), and SetOrthographicSize().

9.60.4.5 m_PerspectiveFar

```
float Vesper::SceneCamera::m_PerspectiveFar = 1000.0f  [private]
```

Referenced by GetPerspectiveFarClip(), SetPerspective(), and SetPerspectiveFarClip().

9.60.4.6 m_PerspectiveFOV

```
float Vesper::SceneCamera::m_PerspectiveFOV = glm::radians(45.0f)  [private]
```

9.60.4.7 m_PerspectiveNear

```
float Vesper::SceneCamera::m_PerspectiveNear = 0.01f  [private]
```

Referenced by GetPerspectiveNearClip(), SetPerspective(), and SetPerspectiveNearClip().

9.60.4.8 m_ProjectionType

```
ProjectionType Vesper::SceneCamera::m_ProjectionType = ProjectionType::Orthographic  [private]
```

Referenced by GetProjectionType(), RecalculateProjection(), SetPerspective(), and SetProjectionType().

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

- Vesper/src/Vesper/Scene/SceneCamera.h
- Vesper/src/Vesper/Scene/SceneCamera.cpp

9.61 Vesper::SceneHierarchyPanel Class Reference

```
#include <SceneHierarchyPanel.h>
```

Public Member Functions

- SceneHierarchyPanel ()=default
- SceneHierarchyPanel (const Ref< Scene > &context)
- void SetContext (const Ref< Scene > &context)
- void OnImGuiRender ()
- Entity GetSelectedEntity () const
- void SetSelectedEntity (Entity entity)

Private Member Functions

- template<typename T>
  void DisplayAddComponentEntry (const std::string &entryName)
- void DrawEntityNode (Entity entity)
- void DrawComponents (Entity entity)

Private Attributes

- Ref< Scene > m_Context
- Entity m_SelectionContext
- Ref< Framebuffer > m_Framebuffer

9.61.1 Constructor & Destructor Documentation

9.61.1.1 SceneHierarchyPanel() [1/2]

```
Vesper::SceneHierarchyPanel::SceneHierarchyPanel () [default]
```

9.61.1.2 SceneHierarchyPanel() [2/2]

```
Vesper::SceneHierarchyPanel::SceneHierarchyPanel (
            const Ref< Scene > & context)
00017     {
00018         SetContext(context);
00019     }
```

9.61.2 Member Function Documentation

9.61.2.1 DisplayAddComponentEntry()

```
template<typename T>
void Vesper::SceneHierarchyPanel::DisplayAddComponentEntry (
            const std::string & entryName) [private]
00545                                                                                 {
00546         if (!m_SelectionContext.HasComponent<T>())
00547         {
00548             if (ImGui::MenuItem(entryName.c_str()))
00549             {
00550                 m_SelectionContext.AddComponent<T>();
00551                 ImGui::CloseCurrentPopup();
00552             }
00553         }
00554     }
```

9.61.2.2 DrawComponents()

```
void Vesper::SceneHierarchyPanel::DrawComponents (
            Entity entity) [private]
00373     {
00374         if (entity.HasComponent<NameComponent>())
00375         {
00376             auto& name = entity.GetComponent<NameComponent>().Name;
00377
00378             char buffer[256];
00379             memset(buffer, 0, sizeof(buffer));
00380             strncpy_s(buffer, sizeof(buffer), name.c_str(), sizeof(buffer));
00381             if (ImGui::InputText("##Name", buffer, sizeof(buffer)))
00382             {
00383                 name = std::string(buffer);
00384             }
00385         }
00386
00387         ImGui::SameLine();
00388         ImGui::PushItemWidth(-1);
```

```
00389
00390         if (ImGui::Button("Add Component"))
00391             ImGui::OpenPopup("AddComponent");
00392
00393         if (ImGui::BeginPopup("AddComponent"))
00394         {
00395             DisplayAddComponentEntry<CameraComponent>("Camera");
00396             //DisplayAddComponentEntry<ScriptComponent>("Script");
00397             DisplayAddComponentEntry<SpriteRendererComponent>("Sprite Renderer");
00398             DisplayAddComponentEntry<SubTextureComponent>("SubTexture");
00399
00400             ImGui::EndPopup();
00401         }
00402
00403         ImGui::PopItemWidth();
00404
00405         DrawComponent<TransformComponent>("Transform", entity, [](auto& component)
00406             {
00407                 DrawVec3Control("Translation", component.Translation);
00408                 glm::vec3 rotation = glm::degrees(component.Rotation);
00409                 DrawVec3Control("Rotation", rotation);
00410                 component.Rotation = glm::radians(rotation);
00411                 DrawVec3Control("Scale", component.Scale, 1.0f);
00412             });
00413
00414         DrawComponent<CameraComponent>("Camera", entity, [](auto& component)
00415             {
00416                 auto& camera = component.Camera;
00417
00418                 ImGui::Checkbox("Primary", &component.Primary);
00419
00420                 const char* projectionTypeStrings[] = { "Perspective", "Orthographic" };
00421                 const char* currentProjectionTypeString =
    projectionTypeStrings[(int)camera.GetProjectionType()];
00422                 if (ImGui::BeginCombo("Projection", currentProjectionTypeString))
00423                 {
00424                     for (int i = 0; i < 2; i++)
00425                     {
00426                         bool isSelected = currentProjectionTypeString == projectionTypeStrings[i];
00427                         if (ImGui::Selectable(projectionTypeStrings[i], isSelected))
00428                         {
00429                             currentProjectionTypeString = projectionTypeStrings[i];
00430                             camera.SetProjectionType((SceneCamera::ProjectionType)i);
00431                         }
00432
00433                         if (isSelected)
00434                             ImGui::SetItemDefaultFocus();
00435                     }
00436
00437                     ImGui::EndCombo();
00438                 }
00439
00440                 if (camera.GetProjectionType() == SceneCamera::ProjectionType::Perspective)
00441                 {
00442                     float perspectiveVerticalFov = glm::degrees(camera.GetPerspectiveVerticalFOV());
00443                     if (ImGui::DragFloat("Vertical FOV", &perspectiveVerticalFov))
00444                         camera.SetPerspectiveVerticalFOV(glm::radians(perspectiveVerticalFov));
00445
00446                     float perspectiveNear = camera.GetPerspectiveNearClip();
00447                     if (ImGui::DragFloat("Near", &perspectiveNear))
00448                         camera.SetPerspectiveNearClip(perspectiveNear);
00449
00450                     float perspectiveFar = camera.GetPerspectiveFarClip();
00451                     if (ImGui::DragFloat("Far", &perspectiveFar))
00452                         camera.SetPerspectiveFarClip(perspectiveFar);
00453                 }
00454
00455                 if (camera.GetProjectionType() == SceneCamera::ProjectionType::Orthographic)
00456                 {
00457                     float orthoSize = camera.GetOrthographicSize();
00458                     if (ImGui::DragFloat("Size", &orthoSize))
00459                         camera.SetOrthographicSize(orthoSize);
00460
00461                     float orthoNear = camera.GetOrthographicNearClip();
00462                     if (ImGui::DragFloat("Near", &orthoNear))
00463                         camera.SetOrthographicNearClip(orthoNear);
00464
00465                     float orthoFar = camera.GetOrthographicFarClip();
00466                     if (ImGui::DragFloat("Far", &orthoFar))
00467                         camera.SetOrthographicFarClip(orthoFar);
00468
00469                     ImGui::Checkbox("Fixed Aspect Ratio", &component.FixedAspectRatio);
00470                 }
00471             });
00472
00473         DrawComponent<SpriteRendererComponent>("Sprite Renderer", entity, [](auto& component)
00474             {
```

```
00475                 ImGui::ColorEdit4("Color", glm::value_ptr(component.Color));
00476
00477                 // Separate checkbox for enabling/disabling texture usage
00478                 ImGui::Checkbox("Texture Enabled", &component.TextureEnabled);
00479                 ImGui::SameLine();
00480
00481                 // Display current texture name (if any) and buttons to Set / Change / Clear the
      texture
00482                 if (component.Texture)
00483                 {
00484                     ImGui::TextUnformatted(component.Texture->GetName().c_str());
00485                     ImGui::SameLine();
00486                     if (ImGui::Button("Change Texture"))
00487                     {
00488                         std::string filePath = FileDialogs::OpenFile("Image Files
      (*.png;*.jpg;*.jpeg;*.bmp;*.tga)\0*.png;*.jpg;*.jpeg;*.bmp;*.tga\0All Files (*.*)\0*.*\0");
00489                         if (!filePath.empty())
00490                         {
00491                             auto tex = Texture2D::Create(filePath);
00492                             if (tex)
00493                             {
00494                                 component.Texture = tex;
00495                                 component.TextureEnabled = true;
00496                             }
00497                             else
00498                             {
00499                                 VZ_WARN("Could not load texture {0}", filePath);
00500                             }
00501                         }
00502                     }
00503                     ImGui::SameLine();
00504                     if (ImGui::Button("Clear Texture"))
00505                     {
00506                         component.Texture = nullptr;
00507                         component.TextureEnabled = false;
00508                     }
00509                 }
00510                 else
00511                 {
00512                     ImGui::SameLine();
00513                     if (ImGui::Button("Set Texture"))
00514                     {
00515                         std::string filePath = FileDialogs::OpenFile("Image Files
      (*.png;*.jpg;*.jpeg;*.bmp;*.tga)\0*.png;*.jpg;*.jpeg;*.bmp;*.tga\0All Files (*.*)\0*.*\0");
00516                         if (!filePath.empty())
00517                         {
00518                             auto tex = Texture2D::Create(filePath);
00519                             if (tex)
00520                             {
00521                                 component.Texture = tex;
00522                                 component.TextureEnabled = true;
00523                             }
00524                             else
00525                             {
00526                                 VZ_WARN("Could not load texture {0}", filePath);
00527                             }
00528                         }
00529                     }
00530                 }
00531
00532                 ImGui::DragFloat("Tiling Factor", &component.TilingFactor, 0.1f, 0.0f, 100.0f);
00533
00534             });
00535
00536         DrawComponent<SubTextureComponent>("SubTexture", entity, [](auto& component)
00537             {
00538                 SubTextureEdit(component.SubTexture->GetTexture()->GetName(), component);
00539
00540             });
00541     }
```

9.61.2.3 DrawEntityNode()

```
void Vesper::SceneHierarchyPanel::DrawEntityNode (
            Entity entity)  [private]
```

TODO: Improve name duplication logic

TODO: Draw child entities here in the future

```
00068     {
```

```
00069         auto& name = entity.GetComponent<NameComponent>().Name;
00070         void* nodeID = (void*)(uint64_t)(uint32_t)entity;
00071 
00072         ImGuiTreeNodeFlags flags = ((m_SelectionContext == entity) ? ImGuiTreeNodeFlags_Selected : 0)
    | ImGuiTreeNodeFlags_OpenOnArrow;
00073         flags |= ImGuiTreeNodeFlags_SpanAvailWidth;
00074         bool opened = ImGui::TreeNodeEx(nodeID, flags, name.c_str());
00075         if (ImGui::IsItemClicked())
00076         {
00077             m_SelectionContext = entity;
00078         }
00079 
00080         bool entityDeleted = false;
00081         if (ImGui::BeginPopupContextItem())
00082         {
00083             if (ImGui::MenuItem("Delete Entity"))
00084                 entityDeleted = true;
00085 
00086             if (ImGui::MenuItem("Duplicate Entity"))
00087             {
00088                 Entity newEntity = m_Context->CreateEntity(name);
00089                 // Copy components
00090                 if (entity.HasComponent<NameComponent>()) {
00091                     auto& src = entity.GetComponent<NameComponent>();
00092                     auto& newEntName = newEntity.GetComponent<NameComponent>();
00094                     if (src.Name.capacity() > 0 && isdigit(src.Name.back()))
00095                     {
00096                         // Increment trailing number
00097                         size_t i = src.Name.size() - 1;
00098                         while (i > 0 && isdigit(src.Name[i - 1]))
00099                             --i;
00100                         std::string baseName = src.Name.substr(0, i);
00101                         std::string numberStr = src.Name.substr(i);
00102                         int number = std::stoi(numberStr);
00103                         newEntName.Name = baseName + std::to_string(number + 1);
00104                     }
00105                     else
00106                         newEntName.Name = src.Name + "1";
00107                 }
00108                 if (entity.HasComponent<SpriteRendererComponent>())
00109                 {
00110                     auto& src = entity.GetComponent<SpriteRendererComponent>();
00111                     newEntity.AddComponent<SpriteRendererComponent>(src);
00112                 }
00113                 if (entity.HasComponent<CameraComponent>())
00114                 {
00115                     auto& cc = entity.GetComponent<CameraComponent>();
00116                     newEntity.AddComponent<CameraComponent>(cc);
00117                 }
00118                 if (entity.HasComponent<NativeScriptComponent>())
00119                 {
00120                     auto& nsc = entity.GetComponent<NativeScriptComponent>();
00121                     newEntity.AddComponent<NativeScriptComponent>(nsc);
00122                 }
00123                 if (entity.HasComponent<TextureAnimationComponent>())
00124                 {
00125                     auto& tac = entity.GetComponent<TextureAnimationComponent>();
00126                     newEntity.AddComponent<TextureAnimationComponent>(tac);
00127                 }
00128                 // Add more components as needed
00129                 m_SelectionContext = newEntity;
00130             }
00131 
00132             ImGui::EndPopup();
00133         }
00134 
00135         if (opened)
00136         {
00137             ImGuiTreeNodeFlags flags = ImGuiTreeNodeFlags_OpenOnArrow |
    ImGuiTreeNodeFlags_SpanAvailWidth;
00138             bool opened = ImGui::TreeNodeEx((void*)9817239, flags, name.c_str());
00139             if (opened)
00140             {
00142                 ImGui::TreePop();
00143             }
00144             ImGui::TreePop();
00145         }
00146 
00147         if (entityDeleted)
00148         {
00149             m_Context->DestroyEntity(entity);
00150             if (m_SelectionContext == entity)
00151                 m_SelectionContext = {};
00152         }
00153     }
```

9.61.2.4 GetSelectedEntity()

```
Entity Vesper::SceneHierarchyPanel::GetSelectedEntity () const  [inline]
00021 { return m_SelectionContext; }
```

9.61.2.5 OnImGuiRender()

```
void Vesper::SceneHierarchyPanel::OnImGuiRender ()
00028     {
00029         ImGui::Begin("Scene Hierarchy");
00030         if (m_Context) {
00031 
00032             auto view = m_Context->m_Registry.view<NameComponent>();
00033 
00034             for (auto entity : view) {
00035                 Entity e{ entity, m_Context.get() };
00036                 DrawEntityNode(e);
00037             }
00038 
00039             if (ImGui::IsMouseDown(0) && ImGui::IsWindowHovered())
00040                 m_SelectionContext = {};
00041 
00042             // Right-click on blank space
00043             if (ImGui::BeginPopupContextWindow(0, ImGuiPopupFlags_NoOpenOverItems |
      ImGuiPopupFlags_MouseButtonRight))
00044             {
00045                 if (ImGui::MenuItem("Create Empty Entity"))
00046                     m_SelectionContext = m_Context->CreateEntity("Empty Entity");
00047 
00048                 ImGui::EndPopup();
00049             }
00050         }
00051         ImGui::End();
00052 
00053         ImGui::Begin("Properties");
00054         if (m_SelectionContext)
00055         {
00056             DrawComponents(m_SelectionContext);
00057         }
00058 
00059         ImGui::End();
00060     }
```

9.61.2.6 SetContext()

```
void Vesper::SceneHierarchyPanel::SetContext (
             const Ref< Scene > & context)
00022     {
00023         m_Context = context;
00024         m_SelectionContext = {};
00025     }
```

9.61.2.7 SetSelectedEntity()

```
void Vesper::SceneHierarchyPanel::SetSelectedEntity (
             Entity entity)
00063     {
00064         m_SelectionContext = entity;
00065     }
```

9.61.3 Member Data Documentation

9.61.3.1 m_Context

```
Ref<Scene> Vesper::SceneHierarchyPanel::m_Context  [private]
```

9.61.3.2 m_Framebuffer

```
Ref<Framebuffer> Vesper::SceneHierarchyPanel::m_Framebuffer  [private]
```

9.61.3.3 m_SelectionContext

```
Entity Vesper::SceneHierarchyPanel::m_SelectionContext  [private]
```

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

- Vesper-Editor/src/Panels/SceneHierarchyPanel.h
- Vesper-Editor/src/Panels/SceneHierarchyPanel.cpp

9.62 Vesper::SceneSerializer Class Reference

```
#include <SceneSerializer.h>
```

Public Member Functions

- SceneSerializer (const Ref< Scene > &scene)
- void Serialize (const std::string &filepath)
- void SerializeRuntime (const std::string &filepath)
- bool Deserialize (const std::string &filepath)
- bool DeserializeRuntime (const std::string &filepath)

Private Attributes

- Ref< Scene > m_Scene

9.62.1 Constructor & Destructor Documentation

9.62.1.1 SceneSerializer()

```
Vesper::SceneSerializer::SceneSerializer (
            const Ref< Scene > & scene)
00117         : m_Scene(scene)
00118     {
00119     }
```

References SceneSerializer().

Referenced by SceneSerializer().

9.62.2 Member Function Documentation

9.62.2.1 Deserialize()

```
bool Vesper::SceneSerializer::Deserialize (
            const std::string & filepath)
```

```
00206                                                                   {
00207
00208         std::ifstream stream(filepath);
00209         std::stringstream strStream;
00210         strStream << stream.rdbuf();
00211
00212         YAML::Node data = YAML::Load(strStream.str());
00213         if (!data["Scene"])
00214             return false;
00215
00216         std::string sceneName = data["Scene"].as<std::string>();
00217         VZ_CORE_TRACE("Deserializing scene: {0}", sceneName);
00218
00219         YAML::Node entities = data["Entities"];
00220         if (entities)
00221         {
00222             for (auto entityNode : entities)
00223             {
00224                 std::string uuid = entityNode["Entity"].as<std::string>();
00225                 std::string name;
00226                 YAML::Node nameNode = entityNode["NameComponent"];
00227                 if (nameNode)
00228                     name = nameNode.as<std::string>();
00229                 VZ_CORE_TRACE("Deserialized entity with ID: {0}, name: {1}", uuid, name);
00230                 Entity deserializedEntity = m_Scene->CreateEntity(name, uuid);
00231                 YAML::Node transformNode = entityNode["TransformComponent"];
00232                 if (transformNode)
00233                 {
00234                     auto& tc = deserializedEntity.GetComponent<TransformComponent>();
00235                     tc.Translation = transformNode["Translation"].as<glm::vec3>();
00236                     tc.Rotation = transformNode["Rotation"].as<glm::vec3>();
00237                     tc.Scale = transformNode["Scale"].as<glm::vec3>();
00238                 }
00239                 YAML::Node cameraNode = entityNode["CameraComponent"];
00240                 if (cameraNode)
00241                 {
00242                     auto& cameraComp = deserializedEntity.AddComponent<CameraComponent>();
00243                     auto& camera = cameraComp.Camera;
00244                     YAML::Node camProps = cameraNode["Camera"];
00245                     camera.SetOrthographic(camProps["OrthographicSize"].as<float>(),
    camProps["OrthographicNear"].as<float>(), camProps["OrthographicFar"].as<float>());
00246                     camera.SetPerspective(camProps["PerspectiveFOV"].as<float>(),
    camProps["PerspectiveNear"].as<float>(), camProps["PerspectiveFar"].as<float>());
00247                     cameraComp.Primary = cameraNode["Primary"].as<bool>();
00248
    camera.SetProjectionType((SceneCamera::ProjectionType)cameraNode["ProjectionType"].as<int>());
00249                     cameraComp.FixedAspectRatio = cameraNode["FixedAspectRatio"].as<bool>();
00250                 }
00251                 YAML::Node spriteNode = entityNode["SpriteRendererComponent"];
00252                 if (spriteNode)
00253                 {
00254                     auto& src = deserializedEntity.AddComponent<SpriteRendererComponent>();
00255                     src.Color = spriteNode["Color"].as<glm::vec4>();
00256                 }
00257             }
00258         }
00259         return true;
00260
00261     }
```

Referenced by Vesper::EditorLayer::OnAttach(), and Vesper::EditorLayer::OpenScene().

9.62.2.2 DeserializeRuntime()

```
bool Vesper::SceneSerializer::DeserializeRuntime (
            const std::string & filepath)
```

```
00264     {
00265         VZ_CORE_ASSERT(false, "Not implemented");
00266         return false;
00267     }
```

9.62.2.3 Serialize()

```
void Vesper::SceneSerializer::Serialize (
            const std::string & filepath)
```

```
00179                                                           {
00180         YAML::Emitter out;
00181         out « YAML::BeginMap; // Scene
00182         out « YAML::Key « "Scene" « YAML::Value « m_Scene->GetName();
00183         out « YAML::Key « "Entities" « YAML::Value « YAML::BeginSeq; // Entities
00184         m_Scene->m_Registry.view<entt::entity>().each([&](auto entityID) {
00185
00186             Entity entity = { entityID, m_Scene.get() };
00187             if (!entity)
00188                 return;
00189
00190             SerializeEntity(out, entity);
00191
00192             });
00193
00194         out « YAML::EndSeq; // Entities
00195         out « YAML::EndMap; // Scene
00196
00197         std::ofstream fout(filepath);
00198         fout « out.c_str();
00199     }
```

Referenced by Vesper::EditorLayer::SaveSceneAs().

9.62.2.4 SerializeRuntime()

```
void Vesper::SceneSerializer::SerializeRuntime (
            const std::string & filepath)
```

```
00202     {
00203         VZ_CORE_ASSERT(false, "Not implemented");
00204     }
```

9.62.3 Member Data Documentation

9.62.3.1 m_Scene

```
Ref<Scene> Vesper::SceneSerializer::m_Scene  [private]
```

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

- Vesper/src/Vesper/Scene/SceneSerializer.h
- Vesper/src/Vesper/Scene/SceneSerializer.cpp

9.63 Vesper::ScriptableEntity Class Reference

```
#include <ScriptableEntity.h>
```

Public Member Functions

- virtual ~ScriptableEntity ()
- template<typename T>
  T & GetComponent ()

Protected Member Functions

- virtual void OnCreate ()
- virtual void OnDestroy ()
- virtual void OnUpdate (Timestep ts)

Private Attributes

- Entity m_Entity

Friends

- class Scene

9.63.1 Constructor & Destructor Documentation

9.63.1.1 ~ScriptableEntity()

```
virtual Vesper::ScriptableEntity::~ScriptableEntity ()  [inline], [virtual]
00010 {};
```

9.63.2 Member Function Documentation

9.63.2.1 GetComponent()

```
template<typename T>
T & Vesper::ScriptableEntity::GetComponent ()  [inline]
00014         {
00015             return m_Entity.GetComponent<T>();
00016         }
```

9.63.2.2 OnCreate()

```
virtual void Vesper::ScriptableEntity::OnCreate ()  [inline], [protected], [virtual]
00019 {}
```

9.63.2.3 OnDestroy()

```
virtual void Vesper::ScriptableEntity::OnDestroy ()  [inline], [protected], [virtual]
00020 {}
```

9.63.2.4 OnUpdate()

```
virtual void Vesper::ScriptableEntity::OnUpdate (
             Timestep ts)  [inline], [protected], [virtual]
00021 {}
```

9.63.3 Friends And Related Symbol Documentation

9.63.3.1 Scene

```
friend class Scene  [friend]
```

9.63.4 Member Data Documentation

9.63.4.1 m_Entity

```
Entity Vesper::ScriptableEntity::m_Entity  [private]
```

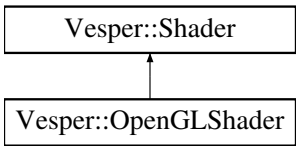
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Scene/ScriptableEntity.h

9.64 Vesper::Shader Class Reference

```
#include <Shader.h>
```

Inheritance diagram for Vesper::Shader:



Public Member Functions

- virtual ~Shader ()=default
- virtual void Bind () const =0
- virtual void Unbind () const =0
- virtual void SetMat4 (const std::string &name, const glm::mat4 &value)=0
- virtual void SetFloat4 (const std::string &name, const glm::vec4 &value)=0
- virtual void SetFloat3 (const std::string &name, const glm::vec3 &value)=0
- virtual void SetFloat (const std::string &name, float value)=0
- virtual void SetInt (const std::string &name, int value)=0
- virtual void SetIntArray (const std::string &name, int *values, uint32_t count)=0
- virtual const std::string & GetName () const =0

Static Public Member Functions

- static Ref< Shader > Create (const std::string &name, const std::string &vertexSrc, const std::string &fragmentSrc)
- static Ref< Shader > Create (const std::string &filepath)

9.64.1 Constructor & Destructor Documentation

9.64.1.1 ~Shader()

```
virtual Vesper::Shader::~Shader ()  [virtual], [default]
```

9.64.2 Member Function Documentation

9.64.2.1 Bind()

```
virtual void Vesper::Shader::Bind () const  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

9.64.2.2 Create() [1/2]

```
Ref< Shader > Vesper::Shader::Create (
            const std::string & filepath)  [static]
```

```
00010     {
00011         switch (Renderer::GetAPI())
00012         {
00013         case RendererAPI::API::None:    VZ_CORE_ASSERT(false, "RendererAPI::None is currently not
      supported!"); return nullptr;
00014         case RendererAPI::API::OpenGL:  return CreateRef<OpenGLShader>(filepath);
00015         }
00016         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00017         return nullptr;
00018     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.64.2.3 Create() [2/2]

```
Ref< Shader > Vesper::Shader::Create (
            const std::string & name,
            const std::string & vertexSrc,
            const std::string & fragmentSrc)  [static]
```

```
00021     {
00022         switch (Renderer::GetAPI())
00023         {
00024             case RendererAPI::API::None:    VZ_CORE_ASSERT(false, "RendererAPI::None is currently not
      supported!"); return nullptr;
00025             case RendererAPI::API::OpenGL:  return CreateRef<OpenGLShader>(name, vertexSrc,
      fragmentSrc);
00026         }
00027         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00028         return nullptr;
00029     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.64.2.4 GetName()

```
virtual const std::string & Vesper::Shader::GetName () const  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

9.64.2.5 SetFloat()

```
virtual void Vesper::Shader::SetFloat (
            const std::string & name,
            float value)  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

9.64.2.6 SetFloat3()

```
virtual void Vesper::Shader::SetFloat3 (
            const std::string & name,
            const glm::vec3 & value)  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

9.64.2.7 SetFloat4()

```
virtual void Vesper::Shader::SetFloat4 (
            const std::string & name,
            const glm::vec4 & value)  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

9.64.2.8 SetInt()

```
virtual void Vesper::Shader::SetInt (
            const std::string & name,
            int value)  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

9.64.2.9 SetIntArray()

```
virtual void Vesper::Shader::SetIntArray (
            const std::string & name,
            int * values,
            uint32_t count)  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

9.64.2.10 SetMat4()

```
virtual void Vesper::Shader::SetMat4 (
            const std::string & name,
            const glm::mat4 & value)  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

9.64.2.11 Unbind()

```
virtual void Vesper::Shader::Unbind () const  [pure virtual]
```

Implemented in Vesper::OpenGLShader.

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

- Vesper/src/Vesper/Renderer/Shader.h
- Vesper/src/Vesper/Renderer/Shader.cpp

9.65 Vesper::ShaderLibrary Class Reference

```
#include <Shader.h>
```

Public Member Functions

- void Add (const std::string &name, const Ref< Shader > &shader)
- void Add (const Ref< Shader > &shader)
- Ref< Shader > Load (const std::string &filepath)
- Ref< Shader > Load (const std::string &name, const std::string &filepath)
- Ref< Shader > Get (const std::string &name)
- bool Exists (const std::string &name) const

Private Attributes

- std::unordered_map< std::string, Ref< Shader > > m_Shaders

9.65.1 Member Function Documentation

9.65.1.1 Add() [1/2]

```
void Vesper::ShaderLibrary::Add (
            const Ref< Shader > & shader)
00039     {
00040         VZ_PROFILE_FUNCTION();
00041         auto& name = shader->GetName();
00042         Add(name, shader);
00043     }
```

9.65.1.2 Add() [2/2]

```
void Vesper::ShaderLibrary::Add (
            const std::string & name,
            const Ref< Shader > & shader)
00032     {
00033         VZ_PROFILE_FUNCTION();
00034         VZ_CORE_ASSERT(!Exists(name), "Shader already exists!");
00035         m_Shaders[name] = shader;
00036     }
```

9.65.1.3 Exists()

```
bool Vesper::ShaderLibrary::Exists (
            const std::string & name) const
```

```
00068     {
00069         VZ_PROFILE_FUNCTION();
00070         return m_Shaders.find(name) != m_Shaders.end();
00071     }
```

9.65.1.4 Get()

```
Vesper::Ref< Vesper::Shader > Vesper::ShaderLibrary::Get (
            const std::string & name)
```

```
00062     {
00063         VZ_CORE_ASSERT(Exists(name), "Shader not found!");
00064         return m_Shaders[name];
00065     }
```

9.65.1.5 Load() [1/2]

```
Vesper::Ref< Vesper::Shader > Vesper::ShaderLibrary::Load (
            const std::string & filepath)
```

```
00046     {
00047         VZ_PROFILE_FUNCTION();
00048         auto shader = Shader::Create(filepath);
00049         Add(Ref<Shader>(shader));
00050         return shader;
00051     }
```

9.65.1.6 Load() [2/2]

```
Vesper::Ref< Vesper::Shader > Vesper::ShaderLibrary::Load (
            const std::string & name,
            const std::string & filepath)
```

```
00054     {
00055         VZ_PROFILE_FUNCTION();
00056         auto shader = Shader::Create(filepath);
00057         Add(Ref<Vesper::Shader>(shader));
00058         return shader;
00059     }
```

9.65.2 Member Data Documentation

9.65.2.1 m_Shaders

```
std::unordered_map<std::string, Ref<Shader> > Vesper::ShaderLibrary::m_Shaders  [private]
```

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

- Vesper/src/Vesper/Renderer/Shader.h
- Vesper/src/Vesper/Renderer/Shader.cpp

9.66 Vesper::SpriteRendererComponent Struct Reference

`#include <Components.h>`

Public Member Functions

- SpriteRendererComponent ()=default
- SpriteRendererComponent (const SpriteRendererComponent &)=default
- SpriteRendererComponent (const glm::vec4 &color)
- operator glm::vec4 & ()
- operator const glm::vec4 & () const
- glm::vec4 & GetColor ()

Public Attributes

- glm::vec4 Color { 1.0f, 1.0f, 1.0f, 1.0f }
- Ref< Texture2D > Texture = nullptr
- float TilingFactor = 1.0f
- bool TextureEnabled = false
- bool Billboard = false

9.66.1 Constructor & Destructor Documentation

9.66.1.1 SpriteRendererComponent() [1/3]

```
Vesper::SpriteRendererComponent::SpriteRendererComponent ()  [default]
```

9.66.1.2 SpriteRendererComponent() [2/3]

```
Vesper::SpriteRendererComponent::SpriteRendererComponent (
            const SpriteRendererComponent & )  [default]
```

9.66.1.3 SpriteRendererComponent() [3/3]

```
Vesper::SpriteRendererComponent::SpriteRendererComponent (
            const glm::vec4 & color)  [inline]
00080           : Color(color) {
00081       }
```

9.66.2 Member Function Documentation

9.66.2.1 GetColor()

```
glm::vec4 & Vesper::SpriteRendererComponent::GetColor ()  [inline]
00086 { return Color; }
```

9.66.2.2 operator const glm::vec4 &()

```
Vesper::SpriteRendererComponent::operator const glm::vec4 & () const  [inline]
00084 { return Color; }
```

9.66.2.3 operator glm::vec4 &()

```
Vesper::SpriteRendererComponent::operator glm::vec4 & ()  [inline]
00083 { return Color; }
```

9.66.3 Member Data Documentation

9.66.3.1 Billboard

```
bool Vesper::SpriteRendererComponent::Billboard = false
```

9.66.3.2 Color

```
glm::vec4 Vesper::SpriteRendererComponent::Color { 1.0f, 1.0f, 1.0f, 1.0f }
00073 { 1.0f, 1.0f, 1.0f, 1.0f };
```

9.66.3.3 Texture

```
Ref<Texture2D> Vesper::SpriteRendererComponent::Texture = nullptr
```

9.66.3.4 TextureEnabled

```
bool Vesper::SpriteRendererComponent::TextureEnabled = false
```

9.66.3.5 TilingFactor

```
float Vesper::SpriteRendererComponent::TilingFactor = 1.0f
```

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

- Vesper/src/Vesper/Scene/Components.h

9.67 Vesper::Renderer2D::Statistics Struct Reference

```
#include <Renderer2D.h>
```

Public Member Functions

- uint32_t GetTotalVertexCount ()
- uint32_t GetTotalIndexCount ()

Public Attributes

- uint32_t DrawCalls = 0
- uint32_t QuadCount = 0

9.67.1 Member Function Documentation

9.67.1.1 GetTotalIndexCount()

```
uint32_t Vesper::Renderer2D::Statistics::GetTotalIndexCount ()  [inline]
00061 { return QuadCount * 6; }
```

References QuadCount.

9.67.1.2 GetTotalVertexCount()

```
uint32_t Vesper::Renderer2D::Statistics::GetTotalVertexCount ()  [inline]
00060 { return QuadCount * 4; }
```

References QuadCount.

9.67.2 Member Data Documentation

9.67.2.1 DrawCalls

```
uint32_t Vesper::Renderer2D::Statistics::DrawCalls = 0
```

Referenced by Vesper::Renderer2D::Flush().

9.67.2.2 QuadCount

```
uint32_t Vesper::Renderer2D::Statistics::QuadCount = 0
```

Referenced by Vesper::Renderer2D::DrawQuad(), Vesper::Renderer2D::DrawQuadRotated(), Vesper::Renderer2D::DrawQuadRotate
Vesper::Renderer2D::DrawQuadRotatedWithTexture(), Vesper::Renderer2D::DrawQuadWithTexture(), Vesper::Renderer2D::DrawQu
GetTotalIndexCount(), and GetTotalVertexCount().

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

- Vesper/src/Vesper/Renderer/Renderer2D.h

9.68 Vesper::SubTexture2D Class Reference

`#include <SubTexture2D.h>`

Public Member Functions

- SubTexture2D (const Ref< Texture2D > &texture, const glm::vec2 &min, const glm::vec2 &max)
- const Ref< Texture2D > GetTexture ()
- glm::vec2 * GetTexCoords ()

Static Public Member Functions

- static Ref< SubTexture2D > CreateFromCoords (const Ref< Texture2D > &texture, const glm::vec2 &coords, const glm::vec2 &cellSize, const glm::vec2 &spriteSize={1, 1})

Private Attributes

- Ref< Texture2D > m_Texture
- glm::vec2 m_TexCoords [4]

9.68.1 Constructor & Destructor Documentation

9.68.1.1 SubTexture2D()

```
Vesper::SubTexture2D::SubTexture2D (
            const Ref< Texture2D > & texture,
            const glm::vec2 & min,
            const glm::vec2 & max)
00009         : m_Texture(texture)
00010     {
00011         m_TexCoords[0] = { min.x, min.y };
00012         m_TexCoords[1] = { max.x, min.y };
00013         m_TexCoords[2] = { max.x, max.y };
00014         m_TexCoords[3] = { min.x, max.y };
00015     }
```

References SubTexture2D().

Referenced by SubTexture2D().

9.68.2 Member Function Documentation

9.68.2.1 CreateFromCoords()

```
Ref< SubTexture2D > Vesper::SubTexture2D::CreateFromCoords (
            const Ref< Texture2D > & texture,
            const glm::vec2 & coords,
            const glm::vec2 & cellSize,
            const glm::vec2 & spriteSize = {1, 1})  [static]
00018     {
00019         glm::vec2 min = { (coords.x * cellSize.x) / texture->GetWidth(), (coords.y * cellSize.y) /
    texture->GetHeight() };
00020         glm::vec2 max = { ((coords.x + spriteSize.x) * cellSize.x) / texture->GetWidth(), ((coords.y +
    spriteSize.y) * cellSize.y) / texture->GetHeight() };
00021
00022         return CreateRef<SubTexture2D>(texture, min, max);
00023     }
```

9.68.2.2 GetTexCoords()

```
glm::vec2 * Vesper::SubTexture2D::GetTexCoords ()  [inline]
00016 { return m_TexCoords; }
```

9.68.2.3 GetTexture()

```
const Ref< Texture2D > Vesper::SubTexture2D::GetTexture ()  [inline]
00015 { return m_Texture; }
```

9.68.3 Member Data Documentation

9.68.3.1 m_TexCoords

```
glm::vec2 Vesper::SubTexture2D::m_TexCoords[4]  [private]
```

9.68.3.2 m_Texture

```
Ref<Texture2D> Vesper::SubTexture2D::m_Texture  [private]
```

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

- Vesper/src/Vesper/Renderer/SubTexture2D.h
- Vesper/src/Vesper/Renderer/SubTexture2D.cpp

9.69 Vesper::SubTextureComponent Struct Reference

```
#include <Components.h>
```

Public Member Functions

- SubTextureComponent ()=default
- SubTextureComponent (const Ref< Texture2D > &texture)
- SubTextureComponent (const Ref< SubTexture2D > &subTexture)
- void SetTexture (const Ref< Texture2D > &texture)
- void SetTilingFactor (const glm::vec2 &tiling)
- void SetOffset (const glm::vec2 &offset)
- operator Ref< SubTexture2D > & ()
- operator const Ref< SubTexture2D > & () const
- Ref< SubTexture2D > & GetSubTexture ()

Public Attributes

- Ref< SubTexture2D > SubTexture
- glm::vec2 TilingFactor = { 1.0f, 1.0f }
- glm::vec2 Offset = { 0.0f, 0.0f }

9.69.1 Constructor & Destructor Documentation

9.69.1.1 SubTextureComponent() [1/3]

```
Vesper::SubTextureComponent::SubTextureComponent () [default]
```

9.69.1.2 SubTextureComponent() [2/3]

```
Vesper::SubTextureComponent::SubTextureComponent (
            const Ref< Texture2D > & texture) [inline]
00099             : SubTexture(SubTexture2D::CreateFromCoords(texture, { 0, 0 }, { texture->GetWidth(),
    texture->GetHeight() })) {
00100         }
```

9.69.1.3 SubTextureComponent() [3/3]

```
Vesper::SubTextureComponent::SubTextureComponent (
            const Ref< SubTexture2D > & subTexture) [inline]
00102             : SubTexture(subTexture) {
00103         }
```

9.69.2 Member Function Documentation

9.69.2.1 GetSubTexture()

```
Ref< SubTexture2D > & Vesper::SubTextureComponent::GetSubTexture () [inline]
00119 { return SubTexture; }
```

9.69.2.2 operator const Ref< SubTexture2D > &()

```
Vesper::SubTextureComponent::operator const Ref< SubTexture2D > & () const [inline]
00118 { return SubTexture; }
```

9.69.2.3 operator Ref< SubTexture2D > &()

```
Vesper::SubTextureComponent::operator Ref< SubTexture2D > & () [inline]
00117 { return SubTexture; }
```

9.69.2.4 SetOffset()

```
void Vesper::SubTextureComponent::SetOffset (
            const glm::vec2 & offset) [inline]
00113                                         {
00114             Offset = offset;
00115         }
```

9.69.2.5 SetTexture()

```
void Vesper::SubTextureComponent::SetTexture (
            const Ref< Texture2D > & texture)  [inline]
00105                                                   {
00106             SubTexture = SubTexture2D::CreateFromCoords(texture, { 0, 0 }, { texture->GetWidth(),
      texture->GetHeight() });
00107         }
```

9.69.2.6 SetTilingFactor()

```
void Vesper::SubTextureComponent::SetTilingFactor (
            const glm::vec2 & tiling)  [inline]
00109                                                 {
00110             TilingFactor = tiling;
00111         }
```

9.69.3 Member Data Documentation

9.69.3.1 Offset

```
glm::vec2 Vesper::SubTextureComponent::Offset = { 0.0f, 0.0f }
00096 { 0.0f, 0.0f };
```

9.69.3.2 SubTexture

```
Ref<SubTexture2D> Vesper::SubTextureComponent::SubTexture
```

9.69.3.3 TilingFactor

```
glm::vec2 Vesper::SubTextureComponent::TilingFactor = { 1.0f, 1.0f }
00095 { 1.0f, 1.0f };
```

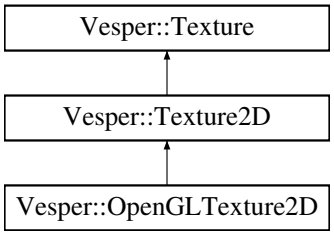
The documentation for this struct was generated from the following file:

- Vesper/src/Vesper/Scene/Components.h

9.70 Vesper::Texture Class Reference

```
#include <Texture.h>
```

Inheritance diagram for Vesper::Texture:



Public Member Functions

- virtual ~Texture ()=default
- virtual uint32_t GetWidth () const =0
- virtual uint32_t GetHeight () const =0
- virtual uint32_t GetRendererID () const =0
- virtual void Bind (uint32_t slot=0) const =0
- virtual void SetData (void *data, uint32_t size)=0
- virtual bool operator== (const Texture2D &other) const =0
- virtual std::string GetName () const =0

9.70.1 Constructor & Destructor Documentation

9.70.1.1 ~Texture()

```
virtual Vesper::Texture::~Texture ()  [virtual], [default]
```

9.70.2 Member Function Documentation

9.70.2.1 Bind()

```
virtual void Vesper::Texture::Bind (
            uint32_t slot = 0) const  [pure virtual]
```

Implemented in Vesper::OpenGLTexture2D.

9.70.2.2 GetHeight()

```
virtual uint32_t Vesper::Texture::GetHeight () const  [pure virtual]
```

Implemented in Vesper::OpenGLTexture2D.

9.70.2.3 GetName()

```
virtual std::string Vesper::Texture::GetName () const  [pure virtual]
```

Implemented in Vesper::OpenGLTexture2D.

9.70.2.4 GetRendererID()

```
virtual uint32_t Vesper::Texture::GetRendererID () const  [pure virtual]
```

Implemented in Vesper::OpenGLTexture2D.

9.70.2.5 GetWidth()

```
virtual uint32_t Vesper::Texture::GetWidth () const  [pure virtual]
```

Implemented in Vesper::OpenGLTexture2D.

9.70.2.6 operator==()

```
virtual bool Vesper::Texture::operator== (
            const Texture2D & other) const  [pure virtual]
```

Implemented in Vesper::OpenGLTexture2D.

9.70.2.7 SetData()

```
virtual void Vesper::Texture::SetData (
            void * data,
            uint32_t size)  [pure virtual]
```

Implemented in Vesper::OpenGLTexture2D.

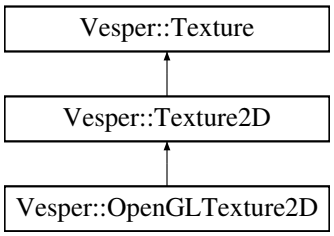
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Renderer/Texture.h

9.71 Vesper::Texture2D Class Reference

```
#include <Texture.h>
```

Inheritance diagram for Vesper::Texture2D:



Static Public Member Functions

- static Ref< Texture2D > Create (uint32_t width, uint32_t height)
- static Ref< Texture2D > Create (const std::string &path)

Additional Inherited Members

Public Member Functions inherited from Vesper::Texture

- virtual ~Texture ()=default
- virtual uint32_t GetWidth () const =0
- virtual uint32_t GetHeight () const =0
- virtual uint32_t GetRendererID () const =0
- virtual void Bind (uint32_t slot=0) const =0
- virtual void SetData (void *data, uint32_t size)=0
- virtual bool operator== (const Texture2D &other) const =0
- virtual std::string GetName () const =0

9.71.1 Member Function Documentation

9.71.1.1 Create() [1/2]

```
Ref< Texture2D > Vesper::Texture2D::Create (
            const std::string & path)  [static]
```

```
00022     {
00023         switch (Renderer::GetAPI())
00024         {
00025             case RendererAPI::API::None:        VZ_CORE_ASSERT(false, "RendererAPI::None is currently
      not supported!"); return nullptr;
00026             case RendererAPI::API::OpenGL:      return CreateRef<OpenGLTexture2D>(path);
00027         }
00028         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00029         return nullptr;
00030     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.71.1.2 Create() [2/2]

```
Ref< Texture2D > Vesper::Texture2D::Create (
            uint32_t width,
            uint32_t height)  [static]
```

```
00011     {
00012         switch (Renderer::GetAPI())
00013         {
00014             case RendererAPI::API::None:        VZ_CORE_ASSERT(false, "RendererAPI::None is currently
      not supported!"); return nullptr;
00015             case RendererAPI::API::OpenGL:      return CreateRef<OpenGLTexture2D>(width, height);
00016         }
00017         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00018         return nullptr;
00019     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

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

- Vesper/src/Vesper/Renderer/Texture.h
- Vesper/src/Vesper/Renderer/Texture.cpp

9.72 Vesper::TextureAnimationComponent Struct Reference

`#include <Components.h>`

Public Member Functions

- TextureAnimationComponent ()=default
- TextureAnimationComponent (const TextureAnimationComponent &)=default
- TextureAnimationComponent (const std::vector< Ref< SubTexture2D > > &subTextures, float frameTime)
- operator std::vector< Ref< SubTexture2D > > & ()
- operator const std::vector< Ref< SubTexture2D > > & () const
- std::vector< Ref< SubTexture2D > > & GetSubTextures ()
- uint32_t GetCurrentFrame () const
- void Update (float deltaTime)

Public Attributes

- std::vector< Ref< SubTexture2D > > SubTextures
- uint32_t CurrentFrame = 0
- float FrameTime = 0.6f
- float TimeAccumulator = 0.0f

9.72.1 Constructor & Destructor Documentation

9.72.1.1 TextureAnimationComponent() [1/3]

```
Vesper::TextureAnimationComponent::TextureAnimationComponent ()  [default]
```

9.72.1.2 TextureAnimationComponent() [2/3]

```
Vesper::TextureAnimationComponent::TextureAnimationComponent (
            const TextureAnimationComponent & )  [default]
```

9.72.1.3 TextureAnimationComponent() [3/3]

```
Vesper::TextureAnimationComponent::TextureAnimationComponent (
            const std::vector< Ref< SubTexture2D > > & subTextures,
            float frameTime)  [inline]
00133           : SubTextures(subTextures), FrameTime(frameTime) {
00134       }
```

9.72.2 Member Function Documentation

9.72.2.1 GetCurrentFrame()

```
uint32_t Vesper::TextureAnimationComponent::GetCurrentFrame () const  [inline]
00139 { return CurrentFrame; }
```

9.72.2.2 GetSubTextures()

```
std::vector< Ref< SubTexture2D > > & Vesper::TextureAnimationComponent::GetSubTextures ()
[inline]
00138 { return SubTextures; }
```

9.72.2.3 operator const std::vector< Ref< SubTexture2D > > &()

```
Vesper::TextureAnimationComponent::operator const std::vector< Ref< SubTexture2D > > & ()
const  [inline]
00136 { return SubTextures; }
```

9.72.2.4 operator std::vector< Ref< SubTexture2D > > &()

```
Vesper::TextureAnimationComponent::operator std::vector< Ref< SubTexture2D > > & ()  [inline]
00135 { return SubTextures; }
```

9.72.2.5 Update()

```
void Vesper::TextureAnimationComponent::Update (
             float deltaTime)  [inline]
00141                                    {
00142             if (SubTextures.empty() || FrameTime <= 0.0f)
00143                 return;
00144
00145             TimeAccumulator += deltaTime;
00146             while (TimeAccumulator >= FrameTime) {
00147                 CurrentFrame = (CurrentFrame + 1) % static_cast<uint32_t>(SubTextures.size());
00148                 TimeAccumulator -= FrameTime;
00149             }
00150         }
```

9.72.3 Member Data Documentation

9.72.3.1 CurrentFrame

```
uint32_t Vesper::TextureAnimationComponent::CurrentFrame = 0
```

9.72.3.2 FrameTime

```
float Vesper::TextureAnimationComponent::FrameTime = 0.6f
```

9.72.3.3 SubTextures

```
std::vector<Ref<SubTexture2D> > Vesper::TextureAnimationComponent::SubTextures
```

9.72.3.4 TimeAccumulator

```
float Vesper::TextureAnimationComponent::TimeAccumulator = 0.0f
```

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

- Vesper/src/Vesper/Scene/Components.h

9.73 Vesper::TextureLibrary Class Reference

```
#include <Texture.h>
```

Public Member Functions

- void Add (const std::string &name, const Ref< Texture2D > &texture)
- void Add (const Ref< Texture2D > &texture)
- Ref< Texture2D > Load (const std::string &filepath)
- Ref< Texture2D > Load (const std::string &name, const std::string &filepath)
- Ref< Texture2D > Get (const std::string &name) const
- bool Exists (const std::string &name) const

Private Attributes

- std::unordered_map< std::string, Ref< Texture2D > > m_Textures

9.73.1 Member Function Documentation

9.73.1.1 Add() [1/2]

```
void Vesper::TextureLibrary::Add (
            const Ref< Texture2D > & texture)
00040     {
00041         VZ_PROFILE_FUNCTION();
00042         auto& name = texture->GetName();
00043         Add(name, texture);
00044     }
```

9.73.1.2 Add() [2/2]

```
void Vesper::TextureLibrary::Add (
            const std::string & name,
            const Ref< Texture2D > & texture)
00033     {
00034         VZ_PROFILE_FUNCTION();
00035         VZ_CORE_ASSERT(!Exists(name), "Texture already exists!");
00036         m_Textures[name] = texture;
00037     }
```

9.73.1.3 Exists()

```
bool Vesper::TextureLibrary::Exists (
            const std::string & name) const
```

```
00070     {
00071         VZ_PROFILE_FUNCTION();
00072         return m_Textures.find(name) != m_Textures.end();
00073     }
```

9.73.1.4 Get()

```
Ref< Texture2D > Vesper::TextureLibrary::Get (
            const std::string & name) const
```

```
00063     {
00064         VZ_PROFILE_FUNCTION();
00065         VZ_CORE_ASSERT(Exists(name), "Texture not found!");
00066         return m_Textures.at(name);
00067     }
```

9.73.1.5 Load() [1/2]

```
Ref< Texture2D > Vesper::TextureLibrary::Load (
            const std::string & filepath)
```

```
00047     {
00048         VZ_PROFILE_FUNCTION();
00049         auto texture = Texture2D::Create(filepath);
00050         Add(texture);
00051         return texture;
00052     }
```

9.73.1.6 Load() [2/2]

```
Ref< Texture2D > Vesper::TextureLibrary::Load (
            const std::string & name,
            const std::string & filepath)
```

```
00055     {
00056         VZ_PROFILE_FUNCTION();
00057         auto texture = Texture2D::Create(filepath);
00058         Add(texture);
00059         return texture;
00060     }
```

9.73.2 Member Data Documentation

9.73.2.1 m_Textures

```
std::unordered_map<std::string, Ref<Texture2D> > Vesper::TextureLibrary::m_Textures  [private]
```

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

- Vesper/src/Vesper/Renderer/Texture.h
- Vesper/src/Vesper/Renderer/Texture.cpp

9.74 Vesper::Timestep Class Reference

`#include <Timestep.h>`

Public Member Functions

- Timestep (float time=0.0f)
- operator float () const
- float GetSeconds () const
- float GetMilliseconds () const

Private Attributes

- float m_Time

9.74.1 Constructor & Destructor Documentation

9.74.1.1 Timestep()

```
Vesper::Timestep::Timestep (
             float time = 0.0f)  [inline]
00009             : m_Time(time)
00010         {
00011         }
```

References m_Time.

9.74.2 Member Function Documentation

9.74.2.1 GetMilliseconds()

```
float Vesper::Timestep::GetMilliseconds () const  [inline]
00015 { return m_Time * 1000.0f; }
```

References m_Time.

9.74.2.2 GetSeconds()

```
float Vesper::Timestep::GetSeconds () const  [inline]
00014 { return m_Time; }
```

References m_Time.

9.74.2.3 operator float()

```
Vesper::Timestep::operator float () const  [inline]
00013 { return m_Time; }
```

References m_Time.

9.74.3 Member Data Documentation

9.74.3.1 m_Time

```
float Vesper::Timestep::m_Time  [private]
```

Referenced by GetMilliseconds(), GetSeconds(), operator float(), and Timestep().

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

- Vesper/src/Vesper/Core/Timestep.h

9.75 Vesper::TransformComponent Struct Reference

```
#include <Components.h>
```

Public Member Functions

- TransformComponent ()=default
- TransformComponent (const TransformComponent &)=default
- TransformComponent (const glm::vec3 &translation)
- glm::mat4 GetTransform () const

Public Attributes

- glm::vec3 Translation = { 0.0f, 0.0f, 0.0f }
- glm::vec3 Rotation = { 0.0f, 0.0f, 0.0f }
- glm::vec3 Scale = { 1.0f, 1.0f, 1.0f }

9.75.1 Constructor & Destructor Documentation

9.75.1.1 TransformComponent() [1/3]

```
Vesper::TransformComponent::TransformComponent ()  [default]
```

9.75.1.2 TransformComponent() [2/3]

```
Vesper::TransformComponent::TransformComponent (
            const TransformComponent & )  [default]
```

9.75.1.3 TransformComponent() [3/3]

```
Vesper::TransformComponent::TransformComponent (
            const glm::vec3 & translation)  [inline]
00058            : Translation(translation) {
00059        }
```

9.75.2 Member Function Documentation

9.75.2.1 GetTransform()

```
glm::mat4 Vesper::TransformComponent::GetTransform () const  [inline]
00062         {
00063             glm::mat4 rotation = glm::toMat4(glm::quat(Rotation));
00064
00065             return glm::translate(glm::mat4(1.0f), Translation)
00066                 * rotation
00067                 * glm::scale(glm::mat4(1.0f), Scale);
00068         }
```

9.75.3 Member Data Documentation

9.75.3.1 Rotation

```
glm::vec3 Vesper::TransformComponent::Rotation = { 0.0f, 0.0f, 0.0f }
00052 { 0.0f, 0.0f, 0.0f };
```

9.75.3.2 Scale

```
glm::vec3 Vesper::TransformComponent::Scale = { 1.0f, 1.0f, 1.0f }
00053 { 1.0f, 1.0f, 1.0f };
```

9.75.3.3 Translation

```
glm::vec3 Vesper::TransformComponent::Translation = { 0.0f, 0.0f, 0.0f }
00051 { 0.0f, 0.0f, 0.0f };
```

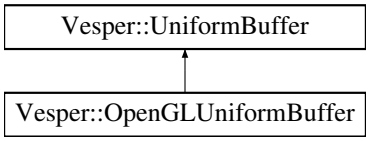
The documentation for this struct was generated from the following file:

- Vesper/src/Vesper/Scene/Components.h

9.76 Vesper::UniformBuffer Class Reference

```
#include <UniformBuffer.h>
```

Inheritance diagram for Vesper::UniformBuffer:



Public Member Functions

- virtual ~UniformBuffer ()
- virtual void SetData (const void *data, uint32_t size, uint32_t offset=0)=0

Static Public Member Functions

- static Ref< UniformBuffer > Create (uint32_t size, uint32_t binding)

9.76.1 Constructor & Destructor Documentation

9.76.1.1 ~UniformBuffer()

```
virtual Vesper::UniformBuffer::~UniformBuffer ()  [inline], [virtual]
00010 {}
```

9.76.2 Member Function Documentation

9.76.2.1 Create()

```
Ref< UniformBuffer > Vesper::UniformBuffer::Create (
            uint32_t size,
            uint32_t binding)  [static]
00010     {
00011         switch (Renderer::GetAPI())
00012         {
00013         case RendererAPI::API::None:    VZ_CORE_ASSERT(false, "RendererAPI::None is currently not
    supported!"); return nullptr;
00014         case RendererAPI::API::OpenGL:  return CreateRef<OpenGLUniformBuffer>(size, binding);
00015         }
00016
00017         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00018         return nullptr;
00019     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.76.2.2 SetData()

```
virtual void Vesper::UniformBuffer::SetData (
            const void * data,
            uint32_t size,
            uint32_t offset = 0)  [pure virtual]
```

Implemented in Vesper::OpenGLUniformBuffer.

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

- Vesper/src/Vesper/Renderer/UniformBuffer.h
- Vesper/src/Vesper/Renderer/UniformBuffer.cpp

9.77 Vesper::UUID Struct Reference

`#include <Components.h>`

Public Member Functions

- UUID ()
- UUID (const std::string &id)
- operator std::string & ()
- operator const std::string & () const

Public Attributes

- std::string ID

9.77.1 Constructor & Destructor Documentation

9.77.1.1 UUID() [1/2]

```
Vesper::UUID::UUID ()  [inline]
00016 { ID = Random::UUID(); }
```

9.77.1.2 UUID() [2/2]

```
Vesper::UUID::UUID (
            const std::string & id)  [inline]
00018             : ID{ id } {
00019         }
```

9.77.2 Member Function Documentation

9.77.2.1 operator const std::string &()

```
Vesper::UUID::operator const std::string & () const  [inline]
00021 { return ID; }
```

9.77.2.2 operator std::string &()

```
Vesper::UUID::operator std::string & ()  [inline]
00020 { return ID; }
```

9.77.3 Member Data Documentation

9.77.3.1 ID

```
std::string Vesper::UUID::ID
```

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

- Vesper/src/Vesper/Scene/Components.h

9.78 Vesper::UUIDComponent Struct Reference

```
#include <Components.h>
```

Public Member Functions

- UUIDComponent ()
- UUIDComponent (const UUIDComponent &)=default
- UUIDComponent (const std::string &id)

Public Attributes

- UUID ID

9.78.1 Constructor & Destructor Documentation

9.78.1.1 UUIDComponent() [1/3]

```
Vesper::UUIDComponent::UUIDComponent ()  [inline]
00027             : ID() {
00028         }
```

9.78.1.2 UUIDComponent() [2/3]

```
Vesper::UUIDComponent::UUIDComponent (
              const UUIDComponent & )  [default]
```

9.78.1.3 UUIDComponent() [3/3]

```
Vesper::UUIDComponent::UUIDComponent (
              const std::string & id)  [inline]
00031             : ID{ id } {
00032         }
```

9.78.2 Member Data Documentation

9.78.2.1 ID

```
UUID Vesper::UUIDComponent::ID
```

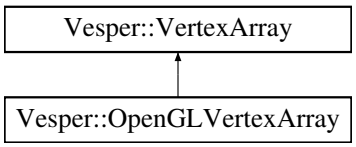
The documentation for this struct was generated from the following file:

- Vesper/src/Vesper/Scene/Components.h

9.79 Vesper::VertexArray Class Reference

```
#include <VertexArray.h>
```

Inheritance diagram for Vesper::VertexArray:



Public Member Functions

- virtual ~VertexArray ()
- virtual void Bind () const =0
- virtual void Unbind () const =0
- virtual void AddVertexBuffer (const Ref< VertexBuffer > &vertexBuffer)=0
- virtual void SetIndexBuffer (const Ref< IndexBuffer > &indexBuffer)=0
- virtual const std::vector< Ref< VertexBuffer > > & GetVertexBuffers ()=0
- virtual const Ref< IndexBuffer > & GetIndexBuffer () const =0

Static Public Member Functions

- static Ref< VertexArray > Create ()

9.79.1 Constructor & Destructor Documentation

9.79.1.1 ~VertexArray()

```
virtual Vesper::VertexArray::~VertexArray ()  [inline], [virtual]
00012 {}
```

9.79.2 Member Function Documentation

9.79.2.1 AddVertexBuffer()

```
virtual void Vesper::VertexArray::AddVertexBuffer (
            const Ref< VertexBuffer > & vertexBuffer)  [pure virtual]
```

Implemented in Vesper::OpenGLVertexArray.

9.79.2.2 Bind()

```
virtual void Vesper::VertexArray::Bind () const  [pure virtual]
```

Implemented in Vesper::OpenGLVertexArray.

9.79.2.3 Create()

```
Ref< VertexArray > Vesper::VertexArray::Create ()  [static]
```

```
00008     {
00009         switch (Renderer::GetAPI())
00010         {
00011         case RendererAPI::API::None:        VZ_CORE_ASSERT(false, "RendererAPI::None is currently not
      supported!"); return nullptr;
00012         case RendererAPI::API::OpenGL:      return CreateRef<OpenGLVertexArray>();
00013         }
00014         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00015         return nullptr;
00016     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.79.2.4 GetIndexBuffer()

```
virtual const Ref< IndexBuffer > & Vesper::VertexArray::GetIndexBuffer () const  [pure virtual]
```

Implemented in Vesper::OpenGLVertexArray.

9.79.2.5 GetVertexBuffers()

```
virtual const std::vector< Ref< VertexBuffer > > & Vesper::VertexArray::GetVertexBuffers ()
[pure virtual]
```

Implemented in Vesper::OpenGLVertexArray.

9.79.2.6 SetIndexBuffer()

```
virtual void Vesper::VertexArray::SetIndexBuffer (
            const Ref< IndexBuffer > & indexBuffer)  [pure virtual]
```

Implemented in Vesper::OpenGLVertexArray.

9.79.2.7 Unbind()

```
virtual void Vesper::VertexArray::Unbind () const  [pure virtual]
```

Implemented in Vesper::OpenGLVertexArray.

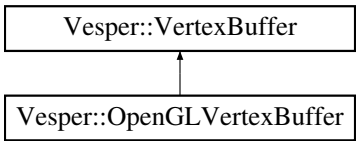
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/Renderer/VertexArray.h
- Vesper/src/Vesper/Renderer/VertexArray.cpp

9.80 Vesper::VertexBuffer Class Reference

```
#include <Buffer.h>
```

Inheritance diagram for Vesper::VertexBuffer:



Public Member Functions

- virtual ~VertexBuffer ()
- virtual void Bind () const =0
- virtual void Unbind () const =0
- virtual void SetLayout (const BufferLayout &layout)=0
- virtual const BufferLayout & GetLayout () const =0
- virtual void SetData (const void *data, uint32_t size)=0

Static Public Member Functions

- static Ref< VertexBuffer > Create (uint32_t size)
- static Ref< VertexBuffer > Create (float *vertices, uint32_t size)

9.80.1 Constructor & Destructor Documentation

9.80.1.1 ~VertexBuffer()

```
virtual Vesper::VertexBuffer::~VertexBuffer ()  [inline], [virtual]
00105 {}
```

9.80.2 Member Function Documentation

9.80.2.1 Bind()

```
virtual void Vesper::VertexBuffer::Bind () const  [pure virtual]
```

Implemented in Vesper::OpenGLVertexBuffer.

9.80.2.2 Create() [1/2]

```
Ref< VertexBuffer > Vesper::VertexBuffer::Create (
            float * vertices,
            uint32_t size)  [static]
```

```
00023     {
00024         switch (Renderer::GetAPI())
00025         {
00026             case RendererAPI::API::None:        VZ_CORE_ASSERT(false, "RendererAPI::None is currently
      not supported!"); return nullptr;
00027             case RendererAPI::API::OpenGL:      return CreateRef<OpenGLVertexBuffer>(vertices, size);
00028         }
00029         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00030         return nullptr;
00031     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.80.2.3 Create() [2/2]

```
Ref< VertexBuffer > Vesper::VertexBuffer::Create (
            uint32_t size)  [static]
```

```
00012     {
00013         switch (Renderer::GetAPI())
00014         {
00015         case RendererAPI::API::None:        VZ_CORE_ASSERT(false, "RendererAPI::None is currently not
      supported!"); return nullptr;
00016         case RendererAPI::API::OpenGL:      return CreateRef<OpenGLVertexBuffer>(size);
00017         }
00018         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00019         return nullptr;
00020     }
```

References Vesper::Renderer::GetAPI(), Vesper::RendererAPI::None, and Vesper::RendererAPI::OpenGL.

9.80.2.4 GetLayout()

```
virtual const BufferLayout & Vesper::VertexBuffer::GetLayout () const  [pure virtual]
```

Implemented in Vesper::OpenGLVertexBuffer.

9.80.2.5 SetData()

```
virtual void Vesper::VertexBuffer::SetData (
            const void * data,
            uint32_t size)  [pure virtual]
```

Implemented in Vesper::OpenGLVertexBuffer.

9.80.2.6 SetLayout()

```
virtual void Vesper::VertexBuffer::SetLayout (
            const BufferLayout & layout)  [pure virtual]
```

Implemented in Vesper::OpenGLVertexBuffer.

9.80.2.7 Unbind()

```
virtual void Vesper::VertexBuffer::Unbind () const  [pure virtual]
```

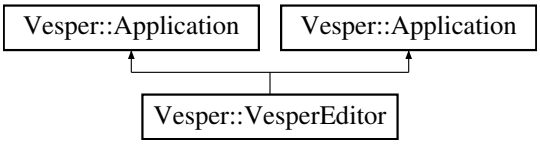
Implemented in Vesper::OpenGLVertexBuffer.

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

- Vesper/src/Vesper/Renderer/Buffer.h
- Vesper/src/Vesper/Renderer/Buffer.cpp

9.81 Vesper::VesperEditor Class Reference

Inheritance diagram for Vesper::VesperEditor:



Public Member Functions

- VesperEditor ()
- ~VesperEditor ()
- VesperEditor ()
- ~VesperEditor ()

Public Member Functions inherited from Vesper::Application

- Application (const std::string &name="")
- virtual ~Application ()
- void Run ()
- void OnEvent (Event &e)
- void PushLayer (Layer *layer)
- void PushOverlay (Layer *overlay)
- void Close ()
- ImGuiLayer * GetImGuiLayer ()
- Window & GetWindow ()

Additional Inherited Members

Static Public Member Functions inherited from Vesper::Application

- static Application & Get ()

9.81.1 Constructor & Destructor Documentation

9.81.1.1 VesperEditor() [1/2]

```
Vesper::VesperEditor::VesperEditor ()  [inline]
00012             : Application("Vesper Editor")
00013         {
00014
00015             PushLayer(new EditorLayer());
00016         }
```

References Vesper::EditorLayer::EditorLayer(), and Vesper::Application::PushLayer().

Referenced by Vesper::CreateApplication().

9.81.1.2 ~VesperEditor() [1/2]

```
Vesper::VesperEditor::~VesperEditor ()  [inline]
00018                         {
00019
00020         }
```

9.81.1.3 VesperEditor() [2/2]

```
Vesper::VesperEditor::VesperEditor ()  [inline]
00012             : Application("Vesper Editor")
00013         {
00014
00015             PushLayer(new EditorLayer());
00016         }
```

9.81.1.4 ~VesperEditor() [2/2]

```
Vesper::VesperEditor::~VesperEditor ()  [inline]
00018                         {
00019
00020         }
```

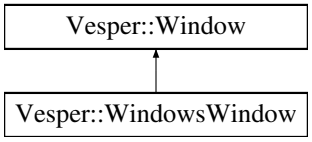
The documentation for this class was generated from the following file:

- Vesper-Editor/src/VesperEditorApp.cpp

9.82 Vesper::Window Class Reference

`#include <Window.h>`

Inheritance diagram for Vesper::Window:



Public Types

- using EventCallbackFn = std::function<void(Event&)>

Public Member Functions

- virtual ~Window ()
- virtual void OnUpdate ()=0
- virtual uint32_t GetWidth () const =0
- virtual uint32_t GetHeight () const =0
- virtual void SetEventCallback (const EventCallbackFn &callback)=0
- virtual void SetVSync (bool enabled)=0
- virtual bool IsVSync () const =0
- virtual void * GetNativeWindow () const =0

Static Public Member Functions

- static Scope< Window > Create (const WindowProps &props=WindowProps())

9.82.1 Member Typedef Documentation

9.82.1.1 EventCallbackFn

```
using Vesper::Window::EventCallbackFn = std::function<void(Event&)>
```

9.82.2 Constructor & Destructor Documentation

9.82.2.1 ~Window()

```
virtual Vesper::Window::~Window ()  [inline], [virtual]
00024 {}
```

9.82.3 Member Function Documentation

9.82.3.1 Create()

```
Scope< Window > Vesper::Window::Create (
            const WindowProps & props = WindowProps())  [static]
```

```
00021     {
00022         return CreateScope<WindowsWindow>(props);
00023     }
```

9.82.3.2 GetHeight()

```
virtual uint32_t Vesper::Window::GetHeight () const  [pure virtual]
```

Implemented in Vesper::WindowsWindow.

Referenced by Vesper::ImGuiLayer::End().

9.82.3.3 GetNativeWindow()

```
virtual void * Vesper::Window::GetNativeWindow () const  [pure virtual]
```

Implemented in Vesper::WindowsWindow.

Referenced by Vesper::Input::GetMousePosition(), Vesper::Input::IsKeyPressed(), and Vesper::Input::IsMouseButtonPressed().

9.82.3.4 GetWidth()

```
virtual uint32_t Vesper::Window::GetWidth () const  [pure virtual]
```

Implemented in Vesper::WindowsWindow.

Referenced by Vesper::ImGuiLayer::End().

9.82.3.5 IsVSync()

```
virtual bool Vesper::Window::IsVSync () const  [pure virtual]
```

Implemented in Vesper::WindowsWindow.

9.82.3.6 OnUpdate()

```
virtual void Vesper::Window::OnUpdate ()  [pure virtual]
```

Implemented in Vesper::WindowsWindow.

9.82.3.7 SetEventCallback()

```
virtual void Vesper::Window::SetEventCallback (
            const EventCallbackFn & callback)  [pure virtual]
```

Implemented in Vesper::WindowsWindow.

9.82.3.8 SetVSync()

```
virtual void Vesper::Window::SetVSync (
            bool enabled)  [pure virtual]
```

Implemented in Vesper::WindowsWindow.

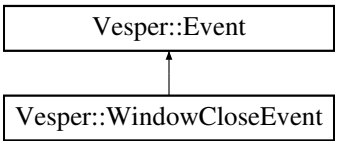
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/App/Window.h
- Vesper/src/Platform/Windows/WindowsWindow.cpp

9.83 Vesper::WindowCloseEvent Class Reference

```
#include <ApplicationEvent.h>
```

Inheritance diagram for Vesper::WindowCloseEvent:



Public Member Functions

- WindowCloseEvent ()

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- virtual std::string ToString () const
- bool IsInCategory (EventCategory category)

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.83.1 Constructor & Destructor Documentation

9.83.1.1 WindowCloseEvent()

```
Vesper::WindowCloseEvent::WindowCloseEvent ()  [inline]
00035 {}
```

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

- Vesper/src/Vesper/Events/ApplicationEvent.h

9.84 Vesper::WindowProps Struct Reference

```
#include <Window.h>
```

Public Member Functions

- WindowProps (const std::string &title="Vesper Engine", uint32_t width=1600, uint32_t height=900)

Public Attributes

- std::string Title
- uint32_t Width
- uint32_t Height

9.84.1 Constructor & Destructor Documentation

9.84.1.1 WindowProps()

```
Vesper::WindowProps::WindowProps (
            const std::string & title = "Vesper Engine",
            uint32_t width = 1600,
            uint32_t height = 900)  [inline]
00016           : Title(title), Width(width), Height(height) {}
```

References Height, and Width.

9.84.2 Member Data Documentation

9.84.2.1 Height

```
uint32_t Vesper::WindowProps::Height
```

Referenced by Vesper::WindowsWindow::Init(), and WindowProps().

9.84.2.2 Title

```
std::string Vesper::WindowProps::Title
```

9.84.2.3 Width

```
uint32_t Vesper::WindowProps::Width
```

Referenced by Vesper::WindowsWindow::Init(), and WindowProps().

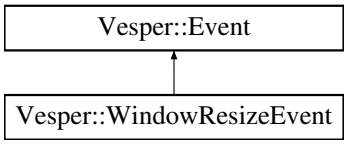
The documentation for this struct was generated from the following file:

- Vesper/src/Vesper/App/Window.h

9.85 Vesper::WindowResizeEvent Class Reference

```
#include <ApplicationEvent.h>
```

Inheritance diagram for Vesper::WindowResizeEvent:



Public Member Functions

- WindowResizeEvent (unsigned int width, unsigned int height)
- unsigned int GetWidth () const
- unsigned int GetHeight () const
- std::string ToString () const override

Public Member Functions inherited from Vesper::Event

- virtual ~Event ()=default
- virtual EventType GetEventType () const =0
- virtual const char * GetName () const =0
- virtual int GetCategoryFlags () const =0
- bool IsInCategory (EventCategory category)

Private Attributes

- unsigned int m_Width
- unsigned int m_Height

Additional Inherited Members

Public Attributes inherited from Vesper::Event

- bool Handled = false

9.85.1 Constructor & Destructor Documentation

9.85.1.1 WindowResizeEvent()

```
Vesper::WindowResizeEvent::WindowResizeEvent (
            unsigned int width,
            unsigned int height)  [inline]
00013           : m_Width(width), m_Height(height) {}
```

References m_Height, and m_Width.

9.85.2 Member Function Documentation

9.85.2.1 GetHeight()

```
unsigned int Vesper::WindowResizeEvent::GetHeight () const  [inline]
00016 { return m_Height; }
```

References m_Height.

Referenced by Vesper::Application::OnWindowResize(), and Vesper::OrthographicCameraController::OnWindowResized().

9.85.2.2 GetWidth()

```
unsigned int Vesper::WindowResizeEvent::GetWidth () const  [inline]
00015 { return m_Width; }
```

References m_Width.

Referenced by Vesper::Application::OnWindowResize(), and Vesper::OrthographicCameraController::OnWindowResized().

9.85.2.3 ToString()

```
std::string Vesper::WindowResizeEvent::ToString () const  [inline], [override], [virtual]
```

Reimplemented from Vesper::Event.

```
00019         {
00020             std::stringstream ss;
00021             ss « "WindowResizeEvent: " « m_Width « ", " « m_Height;
00022             return ss.str();
00023         }
```

References m_Height, and m_Width.

9.85.3 Member Data Documentation

9.85.3.1 m_Height

```
unsigned int Vesper::WindowResizeEvent::m_Height  [private]
```

Referenced by GetHeight(), ToString(), and WindowResizeEvent().

9.85.3.2 m_Width

```
unsigned int Vesper::WindowResizeEvent::m_Width  [private]
```

Referenced by GetWidth(), ToString(), and WindowResizeEvent().

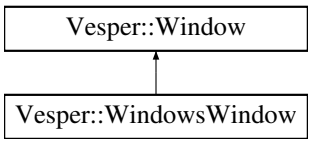
The documentation for this class was generated from the following file:

- Vesper/src/Vesper/Events/ApplicationEvent.h

9.86 Vesper::WindowsWindow Class Reference

```
#include <WindowsWindow.h>
```

Inheritance diagram for Vesper::WindowsWindow:



Classes

- struct WindowData

Public Member Functions

- WindowsWindow (const WindowProps &props)
- virtual ~WindowsWindow ()
- void OnUpdate () override
- unsigned int GetWidth () const override
- unsigned int GetHeight () const override
- void SetEventCallback (const EventCallbackFn &callback) override
- void SetVSync (bool enabled) override
- bool IsVSync () const override
- virtual void * GetNativeWindow () const override

Public Member Functions inherited from Vesper::Window

- virtual ~Window ()

Private Member Functions

- virtual void Init (const WindowProps &props)
- virtual void Shutdown ()

Private Attributes

- GLFWwindow * m_Window
- GraphicsContext * m_Context
- WindowData m_Data

Additional Inherited Members

Public Types inherited from Vesper::Window

- using EventCallbackFn = std::function<void(Event&)>

Static Public Member Functions inherited from Vesper::Window

- static Scope< Window > Create (const WindowProps &props=WindowProps())

9.86.1 Class Documentation

9.86.1.1 struct Vesper::WindowsWindow::WindowData

Class Members

EventCallbackFn	EventCallback	
unsigned int	Height	
string	Title	
bool	VSync	
unsigned int	Width	

9.86.2 Constructor & Destructor Documentation

9.86.2.1 WindowsWindow()

```
Vesper::WindowsWindow::WindowsWindow (
            const WindowProps & props)
00026     {
00027         VZ_PROFILE_FUNCTION();
00028         Init(props);
00029     }
```

References Init().

9.86.2.2 ~WindowsWindow()

```
Vesper::WindowsWindow::~WindowsWindow ()  [virtual]
```

```
00032     {
00033         Shutdown();
00034     }
```

References Shutdown().

9.86.3 Member Function Documentation

9.86.3.1 GetHeight()

```
unsigned int Vesper::WindowsWindow::GetHeight () const  [inline], [override], [virtual]
```

Implements Vesper::Window.

```
00019 { return m_Data.Height; }
```

9.86.3.2 GetNativeWindow()

```
virtual void * Vesper::WindowsWindow::GetNativeWindow () const  [inline], [override], [virtual]
```

Implements Vesper::Window.

```
00025 { return m_Window; }
```

9.86.3.3 GetWidth()

```
unsigned int Vesper::WindowsWindow::GetWidth () const  [inline], [override], [virtual]
```

Implements Vesper::Window.

```
00018 { return m_Data.Width; }
```

9.86.3.4 Init()

```
void Vesper::WindowsWindow::Init (
             const WindowProps & props)  [private], [virtual]
```

```
00037     {
00038         VZ_PROFILE_FUNCTION();
00039         m_Data.Title = props.Title;
00040         m_Data.Width = props.Width;
00041         m_Data.Height = props.Height;
00042 
00043         VZ_CORE_INFO("Creating window {0} ({1}, {2})", props.Title, props.Width, props.Height);
00044 
00045 
00046         if (!s_GLFWInitialized)
00047         {
00048             int success = glfwInit();
00049             VZ_CORE_ASSERT(success, "Could not initialize GLFW!");
00050             glfwSetErrorCallback(GLFWErrorCallback);
00051             s_GLFWInitialized = true;
00052         }
00053 
00054         m_Window = glfwCreateWindow((int)props.Width, (int)props.Height, m_Data.Title.c_str(),
      nullptr, nullptr);
00055 
00056         m_Context = new OpenGLContext(m_Window);
00057         m_Context->Init();
```

```
00058
00059
00060         glfwSetWindowUserPointer(m_Window, &m_Data);
00061         SetVSync(true);
00062
00063         // Set GLFW callbacks
00064         glfwSetWindowSizeCallback(m_Window, [](GLFWwindow* window, int width, int height)
00065         {
00066                 WindowData& data = *(WindowData*)glfwGetWindowUserPointer(window);
00067                 data.Height = height;
00068                 data.Width = width;
00069
00070                 WindowResizeEvent event(width, height);
00071                 VZ_CORE_WARN("Window resized to {0}, {1}", width, height);
00072                 data.EventCallback(event);
00073         });
00074
00075         glfwSetWindowCloseCallback(m_Window, [](GLFWwindow* window)
00076         {
00077                 WindowData& data = *(WindowData*)glfwGetWindowUserPointer(window);
00078                 WindowCloseEvent event;
00079                 data.EventCallback(event);
00080         });
00081
00082         glfwSetKeyCallback(m_Window, [](GLFWwindow* window, int key, int scancode, int action, int
    mods)
00083         {
00084                 WindowData& data = *(WindowData*)glfwGetWindowUserPointer(window);
00085                 switch (action)
00086                 {
00087                     case GLFW_PRESS:
00088                     {
00089                         KeyPressedEvent event(key, 0);
00090                         data.EventCallback(event);
00091                         break;
00092                     }
00093                     case GLFW_RELEASE:
00094                     {
00095                         KeyReleasedEvent event(key);
00096                         data.EventCallback(event);
00097                         break;
00098                     }
00099                     case GLFW_REPEAT:
00100                     {
00101                         KeyPressedEvent event(key, 1);
00102                         data.EventCallback(event);
00103                         break;
00104                     }
00105                 }
00106         });
00107
00108         glfwSetCharCallback(m_Window, [](GLFWwindow* window, unsigned int keycode)
00109         {
00110                 WindowData& data = *(WindowData*)glfwGetWindowUserPointer(window);
00111                 KeyTypedEvent event(keycode);
00112                 data.EventCallback(event);
00113         });
00114
00115         glfwSetMouseButtonCallback(m_Window, [](GLFWwindow* window, int button, int action, int mods)
00116         {
00117                 WindowData& data = *(WindowData*)glfwGetWindowUserPointer(window);
00118                 switch (action)
00119                 {
00120                     case GLFW_PRESS:
00121                     {
00122                         MouseButtonPressedEvent event(button);
00123                         data.EventCallback(event);
00124                         break;
00125                     }
00126                     case GLFW_RELEASE:
00127                     {
00128                         MouseButtonReleasedEvent event(button);
00129                         data.EventCallback(event);
00130                         break;
00131                     }
00132                 }
00133         });
00134
00135         glfwSetScrollCallback(m_Window, [](GLFWwindow* window, double xOffset, double yOffset)
00136         {
00137                 WindowData& data = *(WindowData*)glfwGetWindowUserPointer(window);
00138                 MouseScrolledEvent event((float)xOffset, (float)yOffset);
00139                 data.EventCallback(event);
00140         });
00141
00142         glfwSetCursorPosCallback(m_Window, [](GLFWwindow* window, double xPos, double yPos)
00143         {
```

```
00144                 WindowData& data = *(WindowData*)glfwGetWindowUserPointer(window);
00145                 MouseMovedEvent event((float)xPos, (float)yPos);
00146                 data.EventCallback(event);
00147         });
00148     }
```

References Vesper::GLFWErrorCallback(), Vesper::WindowProps::Height, Vesper::GraphicsContext::Init(), m_Context, Vesper::s_GLFWInitialized, SetVSync(), and Vesper::WindowProps::Width.

Referenced by WindowsWindow().

9.86.3.5 IsVSync()

```
bool Vesper::WindowsWindow::IsVSync () const  [override], [virtual]
```

Implements Vesper::Window.

```
00175     {
00176         return m_Data.VSync;
00177     }
```

9.86.3.6 OnUpdate()

```
void Vesper::WindowsWindow::OnUpdate ()  [override], [virtual]
```

Implements Vesper::Window.

```
00157     {
00158         VZ_PROFILE_FUNCTION();
00159         glfwPollEvents();
00160         m_Context->SwapBuffers();
00161     }
```

References m_Context, and Vesper::GraphicsContext::SwapBuffers().

9.86.3.7 SetEventCallback()

```
void Vesper::WindowsWindow::SetEventCallback (
             const EventCallbackFn & callback)  [inline], [override], [virtual]
```

Implements Vesper::Window.

```
00022 { m_Data.EventCallback = callback; }
```

9.86.3.8 SetVSync()

```
void Vesper::WindowsWindow::SetVSync (
             bool enabled)  [override], [virtual]
```

Implements Vesper::Window.

```
00164     {
00165         VZ_PROFILE_FUNCTION();
00166
00167         if (enabled)
00168             glfwSwapInterval(1);
00169         else
00170             glfwSwapInterval(0);
00171         m_Data.VSync = enabled;
00172     }
```

Referenced by Init().

9.86.3.9 Shutdown()

```
void Vesper::WindowsWindow::Shutdown ()  [private], [virtual]
```

```
00151     {
00152         VZ_PROFILE_FUNCTION();
00153         glfwDestroyWindow(m_Window);
00154     }
```

Referenced by ~WindowsWindow().

9.86.4 Member Data Documentation

9.86.4.1 m_Context

```
GraphicsContext* Vesper::WindowsWindow::m_Context  [private]
```

Referenced by Init(), and OnUpdate().

9.86.4.2 m_Data

```
WindowData Vesper::WindowsWindow::m_Data  [private]
```

9.86.4.3 m_Window

```
GLFWwindow* Vesper::WindowsWindow::m_Window  [private]
```

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

- Vesper/src/Platform/Windows/WindowsWindow.h
- Vesper/src/Platform/Windows/WindowsWindow.cpp

Chapter 10

File Documentation

10.1 README.md File Reference

10.2 Vesper-Editor/src/EditorLayer.cpp File Reference

```
#include <Vesper/ImGui/VesperImGui.h>
#include <ImGuizmo.h>
#include <Vesper/Utils/PlatformUtils.h>
#include "Vesper/Core/Math.h"
#include "EditorLayer.h"
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include "Vesper/Scene/SceneSerializer.h"
```

Namespaces

- namespace Vesper
  TEMPORARY.

Variables

- static const uint32_t s_MapWidth = 20
- static const uint32_t s_MapHeight = 10
- static const char * s_MapTiles

10.2.1 Variable Documentation

10.2.1.1 s_MapHeight

```
const uint32_t s_MapHeight = 10  [static]
```

Referenced by Vesper::EditorLayer::OnUpdate().

10.2.1.2 s_MapTiles

```
const char * s_MapTiles  [static]
```

Initial value:

```
                                =
"GGGGGGGGGGGGGGGGGGGG"
"GGGCCCCCCCCCCCCCCGGG"
"GGGCGGGGGGGGGGGGCGGG"
"GGGCGGRGGGGGGRGGCGGG"
"GGGCGGGGGGGGGGGGCGGG"
"GGGCGGGGGGGGGGGGCGGG"
"GGGCGGPGGGGGGPGGCGGG"
"GGGCGGGGGGGGGGGGCGGG"
"GGGCCCCCCCCCCCCCCGGG"
"GGGGGGGGGGGGGGGGGGGG"
```

Referenced by Vesper::EditorLayer::OnUpdate().

10.2.1.3 s_MapWidth

```
const uint32_t s_MapWidth = 20  [static]
```

Referenced by Vesper::EditorLayer::OnUpdate().

10.3 Vesper-Editor/src/EditorLayer.h File Reference

```
#include <Vesper.h>
#include "Vesper/App/Layer.h"
#include "Vesper/ParticleSystem/ParticleSystem.h"
#include "Vesper/Scene/Scene.h"
#include "Panels/SceneHierarchyPanel.h"
#include "Vesper/Renderer/EditorCamera.h"
```

Classes

- class Vesper::EditorLayer

Namespaces

- namespace Vesper

  TEMPORARY.

10.4 EditorLayer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <Vesper.h>
00004
00005 #include "Vesper/App/Layer.h"
00006 #include "Vesper/ParticleSystem/ParticleSystem.h"
00007 #include "Vesper/Scene/Scene.h"
00008 #include "Panels/SceneHierarchyPanel.h"
00009 #include "Vesper/Renderer/EditorCamera.h"
00010
00011 namespace Vesper {
00012
00013     class EditorLayer : public Layer
00014     {
00015     public:
00016         EditorLayer();
00017         virtual ~EditorLayer() = default;
00018
00019         virtual void OnAttach() override;
00020         virtual void OnDetach() override;
00021         virtual void OnUpdate(Timestep ts) override;
00022         virtual void OnImGuiRender() override;
00023         virtual void OnEvent(Event& e) override;
00024
00025     private:
00026         bool OnKeyPressed(KeyPressedEvent& e);
00027
00028         void NewScene();
00029         void OpenScene();
00030         void SaveSceneAs();
00031         void ResetScene();
00032     private:
00033         SceneHierarchyPanel m_SceneHierarchyPanel;
00034
00035         Ref<Scene> m_ActiveScene;
00036         Ref<Scene> m_EditorScene;
00037
00038         enum class SceneState
00039         {
00040             Edit = 0, Play = 1, Simulate = 2
00041         };
00042         SceneState m_SceneState = SceneState::Edit;
00043
00044         bool m_ViewportFocused = false, m_ViewportHovered = false;
00045         glm::vec2 m_ViewportSize = {0,0};
00046         glm::vec2 m_ViewportBounds[2] = { {0,0}, {0,0} };
00047         bool m_PrimaryCamera = true;
00048         Entity m_CameraEntity;
00049         //Entity m_SecondaryCameraEntity;
00050         int m_GizmoType = -1;
00051         float m_TranslationSnap = 0.5f, m_RotationSnap = 45.0f, m_ScaleSnap = 0.5f;
00052
00053         OrthographicCameraController m_CameraController;
00054
00055         float lastFrameTime = 0.0f;
00056         Entity m_FireEntity, m_SmokeEntity;
00057
00058         // Temp
00059         Ref<VertexArray> m_SquareVA;
00060         Ref<Shader> m_FlatColorShader;
00061         Ref<Texture2D> m_CheckerboardTexture;
00062
00063         Ref<Texture2D> m_SpriteSheetFire;
00064         Ref<Texture2D> m_SpriteSheetSmoke;
00065         Ref<Texture2D> m_SpriteSheetTown;
00066         Ref<Texture2D> m_SpriteSheetCrystals;
00067         Ref<Texture2D> m_SpriteSheetRocks;
00068         Ref<Texture2D> m_SpriteSheetCursedLands;
00069
00070         Ref<SubTexture2D> m_SubTextureFire;
00071         Ref<SubTexture2D> m_SubTextureSmoke;
00072         Ref<SubTexture2D> m_SubTextureTown;
00073         //Ref<SubTexture2D> m_SubTextureCrystal;
00074         //Ref<SubTexture2D> m_SubTextureRock;
00075         //Ref<SubTexture2D> m_SubTexturePlant;
00076
00077         Ref<Framebuffer> m_Framebuffer;
00078
00079         EditorCamera m_EditorCamera;
00080
00081         float m_textureScale = 1.0f;
00082         float m_squareRotation = 25.0f;
```

```
00083         float m_specialQuadRotation = 0.5f;
00084         int ParticleEmitCount = 100;
00085
00086         ParticleSystem m_ParticleSystem;
00087         ParticleProps m_ParticleProps;
00088
00089
00090         bool scene1 = false, scene2 = false, scene3 = false, scene4 = false;
00091         bool useEntityScene = true;
00092
00093         glm::vec4 m_SquareColor = { 0.2f, 0.3f, 0.8f, 1.0f };
00094         glm::vec4 m_TextureTintColor1 = { 1.0f, 1.0f, 1.0f, 1.0f };
00095         glm::vec4 m_TextureTintColor2 = { 1.0f, 1.0f, 1.0f, 1.0f };
00096         glm::vec4 m_BackgroundColor = { 0.1f, 0.1f, 0.1f, 1.0f };
00097         glm::vec4 m_ClearColor = { 0.1f, 0.3f, 0.3f, 1.0f };
00098         glm::vec4 m_SpecialQuadColor = { 0.9f, 0.2f, 0.8f, 1.0f };
00099         bool m_UseSpecialQuadColor = false;
00100
00101         std::unordered_map<char, Ref<SubTexture2D»s_TextureMap;
00102
00103     };
00104
00105
00106 }
```

10.5 Vesper-Editor/src/Panels/SceneHierarchyPanel.cpp File Reference

```
#include <Vesper/Utils/PlatformUtils.h>
#include "SceneHierarchyPanel.h"
#include "Vesper/Scene/Components.h"
#include <ImGui/imgui.h>
#include <imgui/imgui_internal.h>
#include <imgui/misc/cpp/imgui_stdlib.h>
#include <glm/gtc/type_ptr.hpp>
#include <cstring>
```

Namespaces

- namespace Vesper

  TEMPORARY.

Functions

- static void Vesper::DrawVec3Control (const std::string &label, glm::vec3 &values, float resetValue=0.0f, float columnWidth=100.0f)
- static void Vesper::DrawVec2Control (const std::string &label, glm::vec2 &values, float resetValue=0.0f, float columnWidth=100.0f)
- static void Vesper::SubTextureEdit (const std::string &label, SubTextureComponent &subTexture)
- template<typename T, typename UIFunction>
  static void Vesper::DrawComponent (const std::string &name, Entity entity, UIFunction uiFunction)

10.6 Vesper-Editor/src/Panels/SceneHierarchyPanel.h File Reference

```
#include "Vesper/Core/Base.h"
#include "Vesper/Core/Log.h"
#include "Vesper/Scene/Scene.h"
#include "Vesper/Scene/Entity.h"
#include "Vesper/Renderer/Framebuffer.h"
```

Classes

- class Vesper::SceneHierarchyPanel

Namespaces

- namespace Vesper

  TEMPORARY.

10.7 SceneHierarchyPanel.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Vesper/Core/Base.h"
00003 #include "Vesper/Core/Log.h"
00004 #include "Vesper/Scene/Scene.h"
00005 #include "Vesper/Scene/Entity.h"
00006
00007 #include "Vesper/Renderer/Framebuffer.h"
00008
00009 namespace Vesper {
00010
00011     class SceneHierarchyPanel
00012     {
00013     public:
00014         SceneHierarchyPanel() = default;
00015         SceneHierarchyPanel(const Ref<Scene>& context);
00016
00017         void SetContext(const Ref<Scene>& context);
00018
00019         void OnImGuiRender();
00020
00021         Entity GetSelectedEntity() const { return m_SelectionContext; }
00022         void SetSelectedEntity(Entity entity);
00023     private:
00024         template<typename T>
00025         void DisplayAddComponentEntry(const std::string& entryName);
00026
00027         void DrawEntityNode(Entity entity);
00028         void DrawComponents(Entity entity);
00029     private:
00030         Ref<Scene> m_Context;
00031         Entity m_SelectionContext;
00032         Ref<Framebuffer> m_Framebuffer;
00033
00034     };
00035
00036 }
```

10.8 Vesper-Editor/src/VesperEditorApp.cpp File Reference

```
#include <Vesper.h>
#include <Vesper/App/EntryPoint.h>
#include "EditorLayer.h"
```

Classes

- class Vesper::VesperEditor

Namespaces

- namespace Vesper
  TEMPORARY.

Functions

- Application * Vesper::CreateApplication ()

10.9 Vesper/src/Platform/Windows/WindowsInput.cpp File Reference

```
#include "vzpch.h"
#include "Vesper/Input/Input.h"
#include "Vesper/App/Application.h"
#include <GLFW/glfw3.h>
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.10 Vesper/src/Platform/Windows/WindowsPlatformUtils.cpp File Reference

```
#include "vzpch.h"
#include "Vesper/Utils/PlatformUtils.h"
#include "Vesper/App/Application.h"
#include <commdlg.h>
#include <GLFW/glfw3.h>
#include <GLFW/glfw3native.h>
```

Namespaces

- namespace Vesper
  TEMPORARY.

Macros

- #define GLFW_EXPOSE_NATIVE_WIN32

10.10.1 Macro Definition Documentation

10.10.1.1 GLFW_EXPOSE_NATIVE_WIN32

```
#define GLFW_EXPOSE_NATIVE_WIN32
```

10.11 Vesper/src/Platform/Windows/WindowsWindow.cpp File Reference

```
#include "vzpch.h"
#include "WindowsWindow.h"
#include "Vesper/Events/ApplicationEvent.h"
#include "Vesper/Events/MouseEvent.h"
#include "Vesper/Events/KeyEvent.h"
#include "RenderAPI/OpenGL/OpenGLContext.h"
```

Namespaces

- namespace Vesper
  TEMPORARY.

Functions

- static void Vesper::GLFWErrorCallback (int error, const char *description)

Variables

- static bool Vesper::s_GLFWInitialized = false

10.12 Vesper/src/Platform/Windows/WindowsWindow.h File Reference

```
#include "Vesper/App/Window.h"
#include <GLFW/glfw3.h>
#include "Vesper/Renderer/GraphicsContext.h"
```

Classes

- class Vesper::WindowsWindow
- struct Vesper::WindowsWindow::WindowData

Namespaces

- namespace Vesper
  TEMPORARY.

10.12.1 Class Documentation

10.12.1.1 struct Vesper::WindowsWindow::WindowData

Class Members

EventCallbackFn	EventCallback	
unsigned int	Height	
string	Title	
bool	VSync	
unsigned int	Width	

10.13 WindowsWindow.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/App/Window.h"
00004 #include <GLFW/glfw3.h>
00005
00006 #include "Vesper/Renderer/GraphicsContext.h"
00007
00008
00009 namespace Vesper {
00010
00011     class WindowsWindow : public Window
00012     {
00013     public:
00014
00015         WindowsWindow(const WindowProps& props);
00016         virtual ~WindowsWindow();
00017         void OnUpdate() override;
00018         unsigned int GetWidth() const override { return m_Data.Width; }
00019         unsigned int GetHeight() const override { return m_Data.Height; }
00020
00021         // Window attributes
00022         void SetEventCallback(const EventCallbackFn& callback) override { m_Data.EventCallback =
      callback; }
00023         void SetVSync(bool enabled) override;
00024         bool IsVSync() const override;
00025         inline virtual void* GetNativeWindow() const override { return m_Window; }
00026     private:
00027         virtual void Init(const WindowProps& props);
00028         virtual void Shutdown();
00029     private:
00030         GLFWwindow* m_Window;
00031         GraphicsContext* m_Context;
00032         struct WindowData
00033         {
00034             std::string Title;
00035             unsigned int Width, Height;
00036             bool VSync;
00037
00038             EventCallbackFn EventCallback;
00039         };
00040         WindowData m_Data;
00041     };
00042
00043 }
```

10.14 Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLBuffer.h"
#include <glad/glad.h>
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.15 Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.h File Reference

```
#include "Vesper/Renderer/Buffer.h"
```

Classes

- class Vesper::OpenGLVertexBuffer
- class Vesper::OpenGLIndexBuffer

Namespaces

- namespace Vesper
  TEMPORARY.

10.16 OpenGLBuffer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Renderer/Buffer.h"
00004
00005
00006 namespace Vesper {
00007
00008     class OpenGLVertexBuffer : public VertexBuffer
00009     {
00010     public:
00011         OpenGLVertexBuffer(uint32_t size);
00012         OpenGLVertexBuffer(float* vertices, uint32_t size);
00013         virtual ~OpenGLVertexBuffer();
00014         virtual void Bind() const override;
00015         virtual void Unbind() const override;
00016
00017         virtual void SetLayout(const BufferLayout& layout) override { m_Layout = layout; }
00018         virtual const BufferLayout& GetLayout() const override { return m_Layout; }
00019
00020         virtual void SetData(const void* data, uint32_t size) override;
00021     private:
00022         uint32_t m_RendererID;
00023         BufferLayout m_Layout;
00024     };
00025
00026     class OpenGLIndexBuffer : public IndexBuffer
00027     {
00028     public:
00029         OpenGLIndexBuffer(uint32_t* indices, uint32_t count);
00030         virtual ~OpenGLIndexBuffer();
00031         virtual void Bind() const override;
00032         virtual void Unbind() const override;
00033
00034         virtual uint32_t GetCount() const override { return m_Count; }
00035     private:
00036         uint32_t m_RendererID;
00037         uint32_t m_Count;
00038     };
00039
00040 }
```

10.17 Vesper/src/RenderAPI/OpenGL/OpenGLContext.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLContext.h"
#include <GLFW/glfw3.h>
#include <glad/glad.h>
#include "Vesper/Renderer/GraphicsContext.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.18 Vesper/src/RenderAPI/OpenGL/OpenGLContext.h File Reference

```
#include "Vesper/Renderer/GraphicsContext.h"
```

Classes

- class Vesper::OpenGLContext

Namespaces

- namespace Vesper

  TEMPORARY.

10.19 OpenGLContext.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Renderer/GraphicsContext.h"
00004
00005 struct GLFWwindow;
00006
00007 namespace Vesper {
00008     class VESPER_API OpenGLContext : public GraphicsContext
00009     {
00010     public:
00011         OpenGLContext(GLFWwindow* windowHandle);
00012         virtual ~OpenGLContext();
00013         void Init() override;
00014         void SwapBuffers() override;
00015     private:
00016         GLFWwindow* m_WindowHandle;
00017     };
00018 }
```

10.20 Vesper/src/RenderAPI/OpenGL/OpenGLFramebuffer.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLFramebuffer.h"
#include <glad/glad.h>
```

Namespaces

- namespace Vesper
  TEMPORARY.

Variables

- static const uint32_t Vesper::s_MaxFramebufferSize = 8192
  TODO: Get the actual maximum size from the GPU!

10.21 Vesper/src/RenderAPI/OpenGL/OpenGLFramebuffer.h File Reference

```
#include "Vesper/Renderer/Framebuffer.h"
```

Classes

- class Vesper::OpenGLFramebuffer

Namespaces

- namespace Vesper
  TEMPORARY.

10.22 OpenGLFramebuffer.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Vesper/Renderer/Framebuffer.h"
00003
00004
00005 namespace Vesper {
00006
00007     class OpenGLFramebuffer : public Framebuffer
00008     {
00009     public:
00010         OpenGLFramebuffer(const FramebufferSpecification& spec);
00011         virtual ~OpenGLFramebuffer();
00012         void Invalidate();
00013
00014         virtual void Bind() override;
```

```
00015         virtual void Unbind() override;
00016
00017         virtual void Resize(uint32_t width, uint32_t height) override;
00018
00019         virtual uint32_t GetColorAttachmentRendererID() const override { return m_ColorAttachment; }
00020         virtual const FramebufferSpecification& GetSpecification() const { return m_Specification; }
00021
00022
00023     private:
00024         uint32_t m_RendererID;
00025         uint32_t m_ColorAttachment, m_DepthAttachment;
00026         FramebufferSpecification m_Specification;
00027     };
00028
00029 }
```

10.23 Vesper/src/RenderAPI/OpenGL/OpenGLImGuiLayer.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLImGuiLayer.h"
#include "Vesper/ImGui/ImGuiLayer.h"
#include "imgui.h"
#include "backends/imgui_impl_glfw.h"
#include "backends/imgui_impl_opengl3.h"
#include "Vesper/App/Application.h"
#include "Vesper/App/Layer.h"
#include <GLFW/glfw3.h>
#include <glad/glad.h>
#include "ImGuizmo.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.24 Vesper/src/RenderAPI/OpenGL/OpenGLImGuiLayer.h File Reference

```
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
#include "Vesper/Events/ApplicationEvent.h"
#include "Vesper/Events/KeyEvent.h"
#include "Vesper/Events/MouseEvent.h"
#include "Vesper/App/Layer.h"
#include "Vesper/ImGui/ImGuiLayer.h"
```

Classes

- class Vesper::OpenGLImGuiLayer

Namespaces

- namespace Vesper
  TEMPORARY.

10.25 OpenGLImGuiLayer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Core/Base.h"
00004 #include "Vesper/Events/Event.h"
00005 #include "Vesper/Events/ApplicationEvent.h"
00006 #include "Vesper/Events/KeyEvent.h"
00007 #include "Vesper/Events/MouseEvent.h"
00008 #include "Vesper/App/Layer.h"
00009 #include "Vesper/ImGui/ImGuiLayer.h"
00010
00011 namespace Vesper {
00012
00013     class VESPER_API OpenGLImGuiLayer : public ImGuiLayer
00014     {
00015     public:
00016         OpenGLImGuiLayer();
00017         ~OpenGLImGuiLayer();
00018
00019         virtual void OnAttach() override;
00020         virtual void OnDetach() override;
00021         virtual void OnImGuiRender() override;
00022         virtual void OnEvent(Event& e) override;
00023
00024         virtual void Begin() override;
00025         virtual void End() override;
00026
00027         virtual void SetBlockEvents(bool block) { m_BlockEvents = block; }
00028         virtual void SetDarkThemeColors() override;
00029
00030     };
00031
00032 }
```

10.26 Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.cpp File Reference

```
#include "vzpch.h"
#include "Vesper/Renderer/RendererAPI.h"
#include "OpenGLRendererAPI.h"
#include <glad/glad.h>
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.27 Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.h File Reference

```
#include "Vesper/Renderer/RendererAPI.h"
#include <glm/glm.hpp>
#include <memory>
```

Classes

- class Vesper::OpenGLRendererAPI

Namespaces

- namespace Vesper
  TEMPORARY.

10.28 OpenGLRendererAPI.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Renderer/RendererAPI.h"
00004 #include <glm/glm.hpp>
00005 #include <memory>
00006
00007 namespace Vesper {
00008
00009     class OpenGLRendererAPI : public RendererAPI
00010     {
00011     public:
00012         virtual void Init() override;
00013         virtual void SetViewport(uint32_t x, uint32_t y, uint32_t width, uint32_t height) override;
00014         virtual void SetClearColor(const glm::vec4& color) override;
00015         virtual void Clear() override;
00016         virtual void DrawIndexed(const Ref<VertexArray>& vertexArray, uint32_t indexCount = 0)
      override;
00017     };
00018 }
```

10.29 Vesper/src/RenderAPI/OpenGL/OpenGLShader.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLShader.h"
#include <glad/glad.h>
#include <glm/gtc/type_ptr.hpp>
```

Namespaces

- namespace Vesper
  TEMPORARY.

Functions

- static GLenum Vesper::ShaderTypeFromString (const std::string &type)

10.30 Vesper/src/RenderAPI/OpenGL/OpenGLShader.h File Reference

```
#include "Vesper/Renderer/Shader.h"
#include <glm/glm.hpp>
```

Classes

- class Vesper::OpenGLShader

Namespaces

- namespace Vesper
  TEMPORARY.

Typedefs

- typedef unsigned int GLenum

10.30.1 Typedef Documentation

10.30.1.1 GLenum

```
typedef unsigned int GLenum
```

10.31 OpenGLShader.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Renderer/Shader.h"
00004 #include <glm/glm.hpp>
00005
00006 // TODO: Remove this
00007 typedef unsigned int GLenum;
00008
00009 namespace Vesper {
00010
00011     class OpenGLShader : public Shader
00012     {
00013     public:
00014         OpenGLShader(const std::string& filepath);
00015         OpenGLShader(const std::string& name, const std::string& vertexSrc, const std::string&
      fragmentSrc);
00016         ~OpenGLShader();
00017
00018         void Bind() const override;
00019         void Unbind() const override;
00020
00021         virtual void SetMat4(const std::string& name, const glm::mat4& value) override;
00022         virtual void SetFloat4(const std::string& name, const glm::vec4& value) override;
00023         virtual void SetFloat3(const std::string& name, const glm::vec3& value) override;
00024         virtual void SetFloat(const std::string& name, float value) override;
00025         virtual void SetInt(const std::string& name, int value) override;
00026         virtual void SetIntArray(const std::string& name, int* values, uint32_t count) override;
00027
00028         virtual const std::string& GetName() const override { return m_Name; }
00029
00030         void UploadUniformMat4(const std::string& name, const glm::mat4& matrix);
00031         void UploadUniformMat3(const std::string& name, const glm::mat3& matrix);
00032
00033         void UploadUniformFloat4(const std::string& name, const glm::vec4& values);
00034         void UploadUniformFloat3(const std::string& name, const glm::vec3& values);
00035         void UploadUniformFloat2(const std::string& name, const glm::vec2& values);
00036         void UploadUniformFloat(const std::string& name, float value);
00037
00038         void UploadUniformInt(const std::string& name, int value);
00039
00040     private:
00041         std::string ReadFile(const std::string& filepath);
00042         std::unordered_map<GLenum, std::string> PreProcess(const std::string& source);
00043         void Compile(std::unordered_map<GLenum, std::string>& shaderSources);
00044     private:
00045         unsigned int m_RendererID;
00046         std::string m_Name;
00047     };
00048 }
```

10.32 Vesper/src/RenderAPI/OpenGL/OpenGLTexture.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLTexture.h"
#include "stb_image.h"
#include <glad/glad.h>
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.33 Vesper/src/RenderAPI/OpenGL/OpenGLTexture.h File Reference

```
#include "Vesper/Renderer/Texture.h"
#include <glad/glad.h>
```

Classes

- class Vesper::OpenGLTexture2D

Namespaces

- namespace Vesper
  TEMPORARY.

10.34 OpenGLTexture.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Vesper/Renderer/Texture.h"
00003
00004 #include <glad/glad.h>
00005
00006 namespace Vesper {
00007
00008     class OpenGLTexture2D : public Texture2D
00009     {
00010     public:
00011         OpenGLTexture2D(uint32_t width, uint32_t height);
00012         OpenGLTexture2D(const std::string& path);
00013         virtual ~OpenGLTexture2D();
00014
00015         virtual uint32_t GetWidth() const override { return m_Width; }
00016         virtual uint32_t GetHeight() const override { return m_Height; }
00017         virtual uint32_t GetRendererID() const override { return m_RendererID; }
00018
00019         virtual void Bind(uint32_t slot) const override;
00020
00021         virtual void SetData(void* data, uint32_t size) override;
00022
00023         virtual bool operator==(const Texture2D& other) const override
00024         {
00025             return m_RendererID == ((OpenGLTexture2D&)other).m_RendererID;
```

```
00026         }
00027
00028         virtual std::string GetName() const override;
00029
00030     private:
00031         std::string m_Path;
00032         uint32_t m_Width, m_Height;
00033         uint32_t m_RendererID;
00034         GLenum m_InternalFormat, m_DataFormat;
00035     };
00036
00037 }
```

10.35 Vesper/src/RenderAPI/OpenGL/OpenGLUniformBuffer.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLUniformBuffer.h"
#include <glad/glad.h>
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.36 Vesper/src/RenderAPI/OpenGL/OpenGLUniformBuffer.h File Reference

```
#include "Vesper/Renderer/UniformBuffer.h"
```

Classes

- class Vesper::OpenGLUniformBuffer

Namespaces

- namespace Vesper
  TEMPORARY.

10.37 OpenGLUniformBuffer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Renderer/UniformBuffer.h"
00004
00005 namespace Vesper {
00006
00007     class OpenGLUniformBuffer : public UniformBuffer
00008     {
00009     public:
00010         OpenGLUniformBuffer(uint32_t size, uint32_t binding);
00011         virtual ~OpenGLUniformBuffer();
00012
00013         virtual void SetData(const void* data, uint32_t size, uint32_t offset = 0) override;
00014     private:
00015         uint32_t m_RendererID = 0;
00016     };
00017 }
```

10.38 Vesper/src/RenderAPI/OpenGL/OpenGLVertexArray.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLVertexArray.h"
#include <glad/glad.h>
```

Namespaces

- namespace Vesper
  TEMPORARY.

Functions

- static GLenum Vesper::ShaderDataTypeToOpenGLBaseType (ShaderDataType type)

10.39 Vesper/src/RenderAPI/OpenGL/OpenGLVertexArray.h File Reference

```
#include "Vesper/Renderer/VertexArray.h"
```

Classes

- class Vesper::OpenGLVertexArray

Namespaces

- namespace Vesper
  TEMPORARY.

10.40 OpenGLVertexArray.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Vesper/Renderer/VertexArray.h"
00003
00004 namespace Vesper {
00005
00006     class OpenGLVertexArray : public VertexArray {
00007     public:
00008         OpenGLVertexArray();
00009         ~OpenGLVertexArray();
00010
00011         void Bind() const override;
00012         void Unbind() const override;
00013
00014         void AddVertexBuffer(const Ref<VertexBuffer>& vertexBuffer) override;
00015         void SetIndexBuffer(const Ref<IndexBuffer>& indexBuffer) override;
00016
00017         const std::vector<Ref<VertexBuffer>>& GetVertexBuffers() override { return m_VertexBuffers; }
00018         const Ref<IndexBuffer>& GetIndexBuffer() const override { return m_IndexBuffer; }
00019
00020     private:
00021         uint32_t m_RendererID;
00022         uint32_t m_VertexBufferIndex = 0;
00023         std::vector<Ref<VertexBuffer>> m_VertexBuffers;
00024         Ref<IndexBuffer> m_IndexBuffer;
00025
00026     };
00027
00028 }
```

10.41 Vesper/src/Vesper.h File Reference

```
#include "Vesper/Core/Base.h"
#include "Vesper/App/Application.h"
#include "Vesper/App/Layer.h"
#include "Vesper/Debug/Instrumentor.h"
#include "Vesper/Core/Log.h"
#include "Vesper/Core/Random.h"
#include "Vesper/Core/Color.h"
#include "Vesper/Core/Timestep.h"
#include "Vesper/Core/Timer.h"
#include "Vesper/Core/Math.h"
#include "Vesper/Input/Input.h"
#include "Vesper/Input/KeyCodes.h"
#include "Vesper/Input/MouseButtonCodes.h"
#include "Vesper/ImGui/ImGuiLayer.h"
#include "Vesper/ParticleSystem/ParticleSystem.h"
#include "Vesper/Scene/Entity.h"
#include "Vesper/Scene/ScriptableEntity.h"
#include "Vesper/Scene/Components.h"
#include "Vesper/Scene/Scene.h"
#include "Vesper/Renderer/Renderer.h"
#include "Vesper/Renderer/Renderer2D.h"
#include "Vesper/Renderer/RenderCommand.h"
#include "Vesper/Renderer/Buffer.h"
#include "Vesper/Renderer/Framebuffer.h"
#include "Vesper/Renderer/Shader.h"
#include "Vesper/Renderer/Texture.h"
#include "Vesper/Renderer/SubTexture2D.h"
#include "Vesper/Renderer/VertexArray.h"
#include "Vesper/Renderer/Camera.h"
#include "Vesper/Renderer/EditorCamera.h"
#include "Vesper/Renderer/OrthographicCamera.h"
#include "Vesper/Renderer/OrthographicCameraController.h"
```

10.42 Vesper.h

Go to the documentation of this file.

```
00001 #pragma once
00002 // For use by Vesper applications
00003
00004 #include "Vesper/Core/Base.h"
00005
00006 #include "Vesper/App/Application.h"
00007 #include "Vesper/App/Layer.h"
00008
00009 #include "Vesper/Debug/Instrumentor.h"
00010
00011 #include "Vesper/Core/Log.h"
00012 #include "Vesper/Core/Random.h"
00013 #include "Vesper/Core/Color.h"
00014 #include "Vesper/Core/Timestep.h"
00015 #include "Vesper/Core/Timer.h"
00016 #include "Vesper/Core/Math.h"
00017
00018 #include "Vesper/Input/Input.h"
00019 //#include "Vesper/Input/InputContext.h"              /// TODO: Input Context class
00020 //#include "Vesper/Input/InputAction.h"              /// TODO: Input Action class
00021 #include "Vesper/Input/KeyCodes.h"
00022 #include "Vesper/Input/MouseButtonCodes.h"
00023
```

```
00024 /// GUI
00025 #include "Vesper/ImGui/ImGuiLayer.h"                    /// TODO: Abstract this to
      OpenGL/DirectX/Vulkan etc ImGui layers
00026
00027 // -- Particle System (Temporary) ------------------   /// Simple particle system for stress testing
      renderer
00028 #include "Vesper/ParticleSystem/ParticleSystem.h"       /// Temporary starter particle system
00029
00030 // -- Scene - Entity - Component - System ------------------
00031 #include "Vesper/Scene/Entity.h"
00032 #include "Vesper/Scene/ScriptableEntity.h"
00033 #include "Vesper/Scene/Components.h"
00034 #include "Vesper/Scene/Scene.h"                         /// TODO: Give scene a System variable
00035 // #include "Vesper/Scene/Systems.h"                    /// TODO: Systems class
00036 // #include "Vesper/Scene/SystemsManager.h"             /// TODO: Static SystemsManager class
00037
00038 // -- Renderer------------------
00039 #include "Vesper/Renderer/Renderer.h"
00040 #include "Vesper/Renderer/Renderer2D.h"
00041 #include "Vesper/Renderer/RenderCommand.h"
00042
00043 #include "Vesper/Renderer/Buffer.h"
00044 #include "Vesper/Renderer/Framebuffer.h"
00045 #include "Vesper/Renderer/Shader.h"
00046 #include "Vesper/Renderer/Texture.h"
00047 #include "Vesper/Renderer/SubTexture2D.h"
00048 #include "Vesper/Renderer/VertexArray.h"
00049
00050 #include "Vesper/Renderer/Camera.h"
00051 #include "Vesper/Renderer/EditorCamera.h"
00052 #include "Vesper/Renderer/OrthographicCamera.h"
00053 #include "Vesper/Renderer/OrthographicCameraController.h"
00054
00055
00056 // -- Renderer API----------------
```

10.43 Vesper/src/Vesper/App/Application.cpp File Reference

```
#include "vzpch.h"
#include "Application.h"
#include "Vesper/Renderer/Renderer.h"
#include "Vesper/Input/Input.h"
#include <GLFW/glfw3.h>
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.44 Vesper/src/Vesper/App/Application.h File Reference

```
#include "../Core/Base.h"
#include "Window.h"
#include "Vesper/App/LayerStack.h"
#include "Vesper/Events/Event.h"
#include "Vesper/Events/ApplicationEvent.h"
#include "Vesper/Core/Timestep.h"
#include "Vesper/ImGui/ImGuiLayer.h"
#include "Vesper/Renderer/RendererAPI.h"
```

Classes

- struct Vesper::ApplicationSettings
  WIP. More...
- class Vesper::Application

Namespaces

- namespace Vesper
  TEMPORARY.

Enumerations

- enum class Vesper::WindowMode { Vesper::Windowed = 0 , Vesper::Fullscreen = 1 , Vesper::Borderless = 2 }
  WIP.

Functions

- Application * Vesper::CreateApplication ()

10.44.1 Class Documentation

10.44.1.1 struct Vesper::ApplicationSettings

WIP.

Class Members

string	ApplicationName = "Vesper Application"	
bool	EnableImGui = true	
bool	EnableVSync = false	
uint32_t	Height = 720	
WindowMode	Mode = WindowMode::Windowed	
API	RendererAPI = RendererAPI::API::OpenGL	
uint32_t	Width = 1280	
string	WorkingDirectory	

10.45 Application.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "../Core/Base.h"
00003 #include "Window.h"
00004 #include "Vesper/App/LayerStack.h"
00005 #include "Vesper/Events/Event.h"
00006 #include "Vesper/Events/ApplicationEvent.h"
00007 #include "Vesper/Core/Timestep.h"
00008 #include "Vesper/ImGui/ImGuiLayer.h"
00009
00010 #include "Vesper/Renderer/RendererAPI.h"
00011
00012 namespace Vesper {
00013
00014
00015     /// WIP
00016     enum class WindowMode
00017     {
00018         Windowed = 0,
00019         Fullscreen = 1,
00020         Borderless = 2
00021     };
00022
00023     /// WIP
00024     struct ApplicationSettings {
00025         std::string ApplicationName = "Vesper Application";
00026         std::string WorkingDirectory;
00027         RendererAPI::API RendererAPI = RendererAPI::API::OpenGL;
00028         uint32_t Width = 1280;
00029         uint32_t Height = 720;
00030         WindowMode Mode = WindowMode::Windowed;
00031         bool EnableImGui = true;
00032         bool EnableVSync = false;
00033     };
00034
00035
00036     class Application
00037     {
00038     public:
00039         Application(const std::string& name = "");
00040         virtual ~Application();
00041         void Run();
00042
00043         void OnEvent(Event& e);
00044
00045         void PushLayer(Layer* layer);
00046         void PushOverlay(Layer* overlay);
00047
00048         void Close();
00049
00050         ImGuiLayer* GetImGuiLayer() { return m_ImGuiLayer; }
00051
00052         inline static Application& Get() { return *s_Instance; }
00053         inline Window& GetWindow() { return *m_Window; }
00054     private:
00055         bool OnWindowClose(WindowCloseEvent& e);
00056         bool OnWindowResize(WindowResizeEvent& e);
00057
00058         Scope<Window> m_Window;
00059         ImGuiLayer* m_ImGuiLayer;
00060         bool m_Running = true;
00061         bool m_Minimized = false;
00062         LayerStack m_LayerStack;
00063         float m_LastFrameTime = 0.0f;
00064
00065     private:
00066         static Application* s_Instance;
00067     };
00068
00069     // To be defined in CLIENT
00070     Application* CreateApplication();
00071 }
```

10.46 Vesper/src/Vesper/App/EntryPoint.h File Reference

10.47 EntryPoint.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #ifdef VZ_PLATFORM_WINDOWS
00004
00005 extern Vesper::Application* Vesper::CreateApplication();
00006
00007 int main(int argc, char** argv)
00008 {
00009
00010     Vesper::Log::Init();
00011
00012     VZ_PROFILE_BEGIN_SESSION("Startup", "VesperProfile-Startup.json");
00013     auto app = Vesper::CreateApplication();
00014     VZ_PROFILE_END_SESSION();
00015
00016     VZ_PROFILE_BEGIN_SESSION("Runtime", "VesperProfile-Runtime.json");
00017     app->Run();
00018     VZ_PROFILE_END_SESSION();
00019
00020     VZ_PROFILE_BEGIN_SESSION("Shutdown", "VesperProfile-Shutdown.json");
00021     delete app;
00022     VZ_PROFILE_END_SESSION();
00023 }
00024
00025 #endif
```

10.48 Vesper/src/Vesper/App/Layer.cpp File Reference

```
#include "vzpch.h"
#include "Layer.h"
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.49 Vesper/src/Vesper/App/Layer.h File Reference

```
#include "Vesper/Core/Timestep.h"
#include "Vesper/Events/Event.h"
```

Classes

- class Vesper::Layer

Namespaces

- namespace Vesper
  TEMPORARY.

10.50 Layer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Core/Timestep.h"
00004 #include "Vesper/Events/Event.h"
00005
00006 namespace Vesper {
00007
00008     class VESPER_API Layer
00009     {
00010     public:
00011         Layer(const std::string& name = "Layer");
00012         virtual ~Layer();
00013
00014         virtual void OnAttach() {};
00015         virtual void OnDetach() {};
00016         virtual void OnUpdate(Timestep ts) {};
00017         virtual void OnEvent(Event& event){}
00018         virtual void OnRender() {};
00019         virtual void OnImGuiRender() {};
00020
00021         inline const std::string& GetName() const { return m_DebugName; }
00022     protected:
00023         std::string m_DebugName;
00024     };
00025
00026 }
```

10.51 Vesper/src/Vesper/App/LayerStack.cpp File Reference

```
#include "vzpch.h"
#include "LayerStack.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.52 Vesper/src/Vesper/App/LayerStack.h File Reference

```
#include "Vesper/Core/Base.h"
#include "Layer.h"
```

Classes

- class Vesper::LayerStack

Namespaces

- namespace Vesper

  TEMPORARY.

10.53 LayerStack.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Core/Base.h"
00004 #include "Layer.h"
00005
00006
00007 namespace Vesper {
00008
00009     class VESPER_API LayerStack
00010     {
00011     public:
00012         LayerStack();
00013         ~LayerStack();
00014
00015         void PushLayer(Layer* layer);
00016         void PushOverlay(Layer* overlay);
00017         void PopLayer(Layer* layer);
00018         void PopOverlay(Layer* overlay);
00019
00020         std::vector<Layer*>::iterator begin() { return m_Layers.begin(); }
00021         std::vector<Layer*>::iterator end() { return m_Layers.end(); }
00022         std::vector<Layer*>::reverse_iterator rbegin() { return m_Layers.rbegin(); }
00023         std::vector<Layer*>::reverse_iterator rend() { return m_Layers.rend(); }
00024
00025
00026     private:
00027         std::vector<Layer*> m_Layers;
00028         unsigned int m_LayerInsertIndex = 0;
00029     };
00030
00031
00032 }
```

10.54 Vesper/src/Vesper/App/Window.h File Reference

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
```

Classes

- struct Vesper::WindowProps
- class Vesper::Window

Namespaces

- namespace Vesper

  TEMPORARY.

10.55 Window.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "vzpch.h"
00004
00005 #include "Vesper/Core/Base.h"
```

```
00006 #include "Vesper/Events/Event.h"
00007
00008 namespace Vesper {
00009     struct WindowProps {
00010         std::string Title;
00011         uint32_t Width;
00012         uint32_t Height;
00013         WindowProps(const std::string& title = "Vesper Engine",
00014             uint32_t width = 1600,
00015             uint32_t height = 900)
00016             : Title(title), Width(width), Height(height) {}
00017     };
00018
00019     class VESPER_API Window
00020     {
00021     public:
00022         using EventCallbackFn = std::function<void(Event&)>;
00023
00024         virtual ~Window() {}
00025
00026         virtual void OnUpdate() = 0;
00027
00028         virtual uint32_t GetWidth() const = 0;
00029         virtual uint32_t GetHeight() const = 0;
00030
00031         // Window attributes
00032         virtual void SetEventCallback(const EventCallbackFn& callback) = 0;
00033         virtual void SetVSync(bool enabled) = 0;
00034         virtual bool IsVSync() const = 0;
00035
00036         virtual void* GetNativeWindow() const = 0;
00037
00038         static Scope<Window> Create(const WindowProps& props = WindowProps());
00039
00040     };
00041 }
```

10.56 Vesper/src/Vesper/Core/Asserts.h File Reference

Macros

- #define VZ_ASSERT(x, ...)

  ASSERT MACROS.
- #define VZ_CORE_ASSERT(x, ...)

10.56.1 Macro Definition Documentation

10.56.1.1 VZ_ASSERT

```
#define VZ_ASSERT(
              x,
              ...)
```

ASSERT MACROS.

10.56.1.2 VZ_CORE_ASSERT

```
#define VZ_CORE_ASSERT(
              x,
              ...)
```

10.57 Asserts.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 /// ASSERT MACROS
00004 #ifdef VZ_DEBUG
00005 #define VZ_ENABLE_ASSERTS
00006 #endif
00007
00008 #ifdef VZ_ENABLE_ASSERTS
00009 #define VZ_ASSERT(x, ...) { if(!(x)) { VZ_CORE_ERROR("Assertion Failed: {0}", __VA_ARGS__);
      __debugbreak(); } }
00010 #define VZ_CORE_ASSERT(x, ...) { if(!(x)) { VZ_CORE_ERROR("Assertion Failed: {0}", __VA_ARGS__);
      __debugbreak(); } }
00011 #else
00012 #define VZ_ASSERT(x, ...)
00013 #define VZ_CORE_ASSERT(x, ...)
00014 #endif
```

10.58 Vesper/src/Vesper/Core/Base.h File Reference

```
#include "PlatformDetection.h"
#include "Config.h"
#include "Asserts.h"
#include "Defines_Macros.h"
```

10.59 Base.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "PlatformDetection.h"
00004 #include "Config.h"
00005 #include "Asserts.h"
00006 #include "Defines_Macros.h"
```

10.60 Vesper/src/Vesper/Core/Color.h File Reference

```
#include <glm/glm.hpp>
```

Namespaces

- namespace Vesper

  TEMPORARY.
- namespace Vesper::Color

Functions

- static glm::vec4 Vesper::Color::White ()
- static glm::vec4 Vesper::Color::Black ()
- static glm::vec4 Vesper::Color::Gray ()
- static glm::vec4 Vesper::Color::Red ()
- static glm::vec4 Vesper::Color::Orange ()
- static glm::vec4 Vesper::Color::Yellow ()
- static glm::vec4 Vesper::Color::Green ()
- static glm::vec4 Vesper::Color::Blue ()
- static glm::vec4 Vesper::Color::Indigo ()
- static glm::vec4 Vesper::Color::Purple ()
- static glm::vec4 Vesper::Color::Cyan ()
- static glm::vec4 Vesper::Color::Magenta ()
- static glm::vec4 Vesper::Color::Pink ()
- static glm::vec4 Vesper::Color::Brown ()
- static glm::vec4 Vesper::Color::Transparent ()
- static glm::vec4 Vesper::Color::StripAlpha (const glm::vec4 &color)
- static glm::vec4 Vesper::Color::SetAlpha (const glm::vec4 &color, float alpha=0.0f)

10.61 Color.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <glm/glm.hpp>
00004
00005 namespace Vesper {
00006
00007     namespace Color {
00008
00009         static glm::vec4 White() { return glm::vec4(1.0f, 1.0f, 1.0f, 1.0f); }
00010         static glm::vec4 Black() { return glm::vec4(0.0f, 0.0f, 0.0f, 1.0f); }
00011         static glm::vec4 Gray() { return glm::vec4(0.5f, 0.5f, 0.5f, 1.0f); }
00012
00013         static glm::vec4 Red() { return glm::vec4(1.0f, 0.0f, 0.0f, 1.0f); }
00014         static glm::vec4 Orange() { return glm::vec4(1.0f, 0.5f, 0.0f, 1.0f); }
00015         static glm::vec4 Yellow() { return glm::vec4(1.0f, 1.0f, 0.0f, 1.0f); }
00016         static glm::vec4 Green() { return glm::vec4(0.0f, 1.0f, 0.0f, 1.0f); }
00017         static glm::vec4 Blue() { return glm::vec4(0.0f, 0.0f, 1.0f, 1.0f); }
00018         static glm::vec4 Indigo() { return glm::vec4(0.29f, 0.0f, 0.51f, 1.0f); }
00019         static glm::vec4 Purple() { return glm::vec4(0.5f, 0.0f, 0.5f, 1.0f); }
00020
00021         static glm::vec4 Cyan() { return glm::vec4(0.0f, 1.0f, 1.0f, 1.0f); }
00022         static glm::vec4 Magenta() { return glm::vec4(1.0f, 0.0f, 1.0f, 1.0f); }
00023         static glm::vec4 Pink() { return glm::vec4(1.0f, 0.75f, 0.8f, 1.0f); }
00024         static glm::vec4 Brown() { return glm::vec4(0.6f, 0.4f, 0.2f, 1.0f); }
00025         static glm::vec4 Transparent() { return glm::vec4(0.0f, 0.0f, 0.0f, 0.0f); }
00026
00027         static glm::vec4 StripAlpha(const glm::vec4& color) {   return glm::vec4(color.x, color.y,
      color.z, 1.0f);  }
00028         static glm::vec4 SetAlpha(const glm::vec4& color, float alpha = 0.0f) { return
      glm::vec4(color.x, color.y, color.z, alpha);}
00029     }
00030
00031 }
```

10.62 Vesper/src/Vesper/Core/Config.h File Reference

Macros

- #define VZ_DEFAULT_TEXTURE Texture2D::Create("Resources/Textures/Checkerboard.png")

  DLL support.

10.62.1 Macro Definition Documentation

10.62.1.1 VZ_DEFAULT_TEXTURE

```
#define VZ_DEFAULT_TEXTURE Texture2D::Create("Resources/Textures/Checkerboard.png")
```

DLL support.

TODO: Implement Default Graphics API Editor Configurations

10.63 Config.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 /// DLL support
00004 #ifdef VZ_PLATFORM_WINDOWS
00005 #if VZ_DYNAMIC_LINK
00006 #ifdef VZ_BUILD_DLL
00007 #define VESPER_API __declspec(dllexport)
00008 #else
00009 #define VESPER_API __declspec(dllimport)
00010 #endif
00011 #else
00012 #define VESPER_API
00013 #endif
00014 #else
00015 #error Vesper only supports Windows!
00016 #endif // End of DLL support
00017
00018 /// TODO: Implement
00019 /// Default Graphics API
00020
00021
00022
00023 /// Editor Configurations
00024 #ifdef VZ_EDITOR_USE_DEFAULT_SCENE
00025 #define VZ_EDITOR_DEFAULT_SCENE "Resources/Scenes/TriColored3DCubeAndSpriteAnims.vesper"
00026 #endif
00027
00028 #define VZ_DEFAULT_TEXTURE Texture2D::Create("Resources/Textures/Checkerboard.png")
```

10.64 Vesper/src/Vesper/Core/Defines_Macros.h File Reference

Namespaces

- namespace Vesper

  TEMPORARY.

Macros

- #define BIT(x)

  DEFINES.
- #define VZ_BIND_EVENT_FN(fn)
- #define BIND_EVENT_FN(x)

Typedefs

- template<typename T>
  using Vesper::Scope = std::unique_ptr<T>
- template<typename T>
  using Vesper::Ref = std::shared_ptr<T>

Functions

- template<typename T, typename... Args>
  constexpr Scope< T > Vesper::CreateScope (Args &&... args)
- template<typename T, typename... Args>
  constexpr Ref< T > Vesper::CreateRef (Args &&... args)

10.64.1 Macro Definition Documentation

10.64.1.1 BIND_EVENT_FN

```
#define BIND_EVENT_FN(
                x)
```

Value:

```
std::bind(&Application::x, this, std::placeholders::_1)
```

10.64.1.2 BIT

```
#define BIT(
                x)
```

Value:

```
(1 « x)
```

DEFINES.

10.64.1.3 VZ_BIND_EVENT_FN

```
#define VZ_BIND_EVENT_FN(
                fn)
```

Value:

```
[this](auto&&... args) -> decltype(auto) { return this->fn(std::forward<decltype(args)>(args)...); }
```

10.65 Defines_Macros.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 /// DEFINES
00004 #define BIT(x) (1 « x)
00005 #define VZ_BIND_EVENT_FN(fn) [this](auto&&... args) -> decltype(auto) { return
      this->fn(std::forward<decltype(args)>(args)...); }
00006 #define BIND_EVENT_FN(x) std::bind(&Application::x, this, std::placeholders::_1)
00007
00008
00009 /// TYPE ALIASES
00010 namespace Vesper
00011 {
00012     template <typename T>
00013     using Scope = std::unique_ptr<T>;
00014     template <typename T, typename... Args>
00015     constexpr Scope<T> CreateScope(Args&&... args)
00016     {
00017         return std::make_unique<T>(std::forward<Args>(args)...);
00018     }
00019
00020     template <typename T>
00021     using Ref = std::shared_ptr<T>;
00022     template <typename T, typename... Args>
00023     constexpr Ref<T> CreateRef(Args&&... args)
00024     {
00025         return std::make_shared<T>(std::forward<Args>(args)...);
00026     }
00027
00028 }
```

10.66 Vesper/src/Vesper/Core/Log.cpp File Reference

```
#include "vzpch.h"
#include "Log.h"
#include "spdlog/sinks/stdout_color_sinks.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.67 Vesper/src/Vesper/Core/Log.h File Reference

```
#include "Base.h"
#include "spdlog/spdlog.h"
#include "spdlog/fmt/ostr.h"
```

Classes

- class Vesper::Log

Namespaces

- namespace Vesper

  TEMPORARY.

Macros

- #define VZ_CORE_TRACE(...)
- #define VZ_CORE_INFO(...)
- #define VZ_CORE_WARN(...)
- #define VZ_CORE_ERROR(...)
- #define VZ_CORE_FATAL(...)
- #define VZ_TRACE(...)
- #define VZ_INFO(...)
- #define VZ_WARN(...)
- #define VZ_ERROR(...)
- #define VZ_FATAL(...)

10.67.1 Macro Definition Documentation

10.67.1.1 VZ_CORE_ERROR

```
#define VZ_CORE_ERROR(
              ...)
```

Value:

```
::Vesper::Log::GetCoreLogger()->error(__VA_ARGS__)
```

10.67.1.2 VZ_CORE_FATAL

```
#define VZ_CORE_FATAL(
              ...)
```

Value:

```
::Vesper::Log::GetCoreLogger()->critical(__VA_ARGS__)
```

10.67.1.3 VZ_CORE_INFO

```
#define VZ_CORE_INFO(
              ...)
```

Value:

```
::Vesper::Log::GetCoreLogger()->info(__VA_ARGS__)
```

10.67.1.4 VZ_CORE_TRACE

```
#define VZ_CORE_TRACE(
              ...)
```

Value:

```
::Vesper::Log::GetCoreLogger()->trace(__VA_ARGS__)
```

10.67.1.5 VZ_CORE_WARN

```
#define VZ_CORE_WARN(
               ...)
```

Value:

```
::Vesper::Log::GetCoreLogger()->warn(__VA_ARGS__)
```

10.67.1.6 VZ_ERROR

```
#define VZ_ERROR(
               ...)
```

Value:

```
::Vesper::Log::GetClientLogger()->error(__VA_ARGS__)
```

10.67.1.7 VZ_FATAL

```
#define VZ_FATAL(
               ...)
```

Value:

```
::Vesper::Log::GetClientLogger()->critical(__VA_ARGS__)
```

10.67.1.8 VZ_INFO

```
#define VZ_INFO(
               ...)
```

Value:

```
::Vesper::Log::GetClientLogger()->info(__VA_ARGS__)
```

10.67.1.9 VZ_TRACE

```
#define VZ_TRACE(
               ...)
```

Value:

```
::Vesper::Log::GetClientLogger()->trace(__VA_ARGS__)
```

10.67.1.10 VZ_WARN

```
#define VZ_WARN(
               ...)
```

Value:

```
::Vesper::Log::GetClientLogger()->warn(__VA_ARGS__)
```

10.68 Log.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Base.h"
00003 #include "spdlog/spdlog.h"
00004 #include "spdlog/fmt/ostr.h"
00005
00006 namespace Vesper {
00007
00008     class VESPER_API Log
00009     {
00010     public:
00011
00012         static void Init();
00013         inline static std::shared_ptr<spdlog::logger>& GetCoreLogger() { return s_CoreLogger; }
00014         inline static std::shared_ptr<spdlog::logger>& GetClientLogger() { return s_ClientLogger; }
00015     private:
00016         static std::shared_ptr<spdlog::logger> s_CoreLogger;
00017         static std::shared_ptr<spdlog::logger> s_ClientLogger;
00018     };
00019
00020
00021 }
00022
00023 // Core log macros
00024 #define VZ_CORE_TRACE(...)    ::Vesper::Log::GetCoreLogger()->trace(__VA_ARGS__)
00025 #define VZ_CORE_INFO(...)     ::Vesper::Log::GetCoreLogger()->info(__VA_ARGS__)
00026 #define VZ_CORE_WARN(...)     ::Vesper::Log::GetCoreLogger()->warn(__VA_ARGS__)
00027 #define VZ_CORE_ERROR(...)    ::Vesper::Log::GetCoreLogger()->error(__VA_ARGS__)
00028 #define VZ_CORE_FATAL(...)    ::Vesper::Log::GetCoreLogger()->critical(__VA_ARGS__)
00029
00030 // Client log macros
00031 #define VZ_TRACE(...)         ::Vesper::Log::GetClientLogger()->trace(__VA_ARGS__)
00032 #define VZ_INFO(...)          ::Vesper::Log::GetClientLogger()->info(__VA_ARGS__)
00033 #define VZ_WARN(...)          ::Vesper::Log::GetClientLogger()->warn(__VA_ARGS__)
00034 #define VZ_ERROR(...)         ::Vesper::Log::GetClientLogger()->error(__VA_ARGS__)
00035 #define VZ_FATAL(...)         ::Vesper::Log::GetClientLogger()->critical(__VA_ARGS__)
```

10.69 Vesper/src/Vesper/Core/Math.cpp File Reference

```
#include "vzpch.h"
#include "Math.h"
#include <glm/gtx/matrix_decompose.hpp>
```

Namespaces

- namespace Vesper

  TEMPORARY.
- namespace Vesper::Math

Macros

- #define GLM_ENABLE_EXPERIMENTAL

Functions

- bool Vesper::Math::DecomposeTransform (const glm::mat4 &transform, glm::vec3 &translation, glm::vec3 &rotation, glm::vec3 &scale)

10.69.1 Macro Definition Documentation

10.69.1.1 GLM_ENABLE_EXPERIMENTAL

```
#define GLM_ENABLE_EXPERIMENTAL
```

10.70 Vesper/src/Vesper/Core/Math.h File Reference

```
#include <glm/glm.hpp>
```

Namespaces

- namespace Vesper

  TEMPORARY.
- namespace Vesper::Math

Functions

- bool Vesper::Math::DecomposeTransform (const glm::mat4 &transform, glm::vec3 &translation, glm::vec3 &rotation, glm::vec3 &scale)

10.71 Math.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <glm/glm.hpp>
00004
00005 namespace Vesper::Math {
00006
00007     bool DecomposeTransform(const glm::mat4& transform, glm::vec3& translation, glm::vec3& rotation,
      glm::vec3& scale);
00008
00009 }
```

10.72 Vesper/src/Vesper/Core/PlatformDetection.h File Reference

```
#include <memory>
```

10.73 PlatformDetection.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <memory>
00004
00005 // Platform detection
00006 #ifdef _WIN32
00007     /*Windows x64/x86*/
00008 #ifdef _WIN64
00009     /*Windows x64  */
00010 #ifndef VZ_PLATFORM_WINDOWS
00011 #define VZ_PLATFORM_WINDOWS
00012 #endif
00013 #else
00014         /*Windows x86*/
00015 #error "x86 Builds are not supported!"
00016 #endif
00017 #elif defined(__APPLE__) || defined(__MACH__)
00018 #include <TargetConditionals.h>
00019 /*Apple platforms */
00020 #if TARGET_IPHONE_SIMULATOR == 1
00021 #error "IOS Simulator is not supported!"
00022 #elif TARGET_OS_IPHONE == 1
00023 #define VZ_PLATFORM_IOS
00024 #error "IOS is not supported!"
00025 #elif TARGET_OS_MAC == 1
00026 #define VZ_PLATFORM_MACOS
00027 #error "MacOS is not supported!"
00028 #else
00029 #error "Unknown Apple platform!"
00030 #endif
00031 #elif defined(__ANDROID__)
00032 #define VZ_PLATFORM_ANDROID
00033 #error "Android is not supported!"
00034 #elif defined(__linux__)
00035 #define VZ_PLATFORM_LINUX
00036 #error "Linux is not supported!"
00037 #else
00038     /*Unknown compiler/platform*/
00039 #error "Unknown platform!"
00040 #endif // End of platform detection
```

10.74 Vesper/src/Vesper/Core/Random.h File Reference

```
#include <random>
#include <algorithm>
#include <cstdint>
#include <glm/glm.hpp>
#include "Vesper/Debug/Instrumentor.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.
- namespace Vesper::Random

Functions

- std::mt19937 & Vesper::Random::GetRNG ()
- void Vesper::Random::Seed (uint32_t seed)
- uint32_t Vesper::Random::UInt1 (uint32_t max)
- bool Vesper::Random::Bool1 (float trueChance)

- unsigned char Vesper::Random::Char ()
- std::string Vesper::Random::String (size_t length)
- std::string Vesper::Random::HexString (size_t length)
- std::string Vesper::Random::UUID ()
- float Vesper::Random::Float1 ()
- float Vesper::Random::RangeF1 (float min, float max)
- float Vesper::Random::RangeF1_Inclusive (float min, float max)
- glm::vec2 Vesper::Random::Float2 ()
- glm::vec2 Vesper::Random::RangeF2 (float min, float max)
- glm::vec2 Vesper::Random::RangeF2 (float min1, float max1, float min2, float max2)
- glm::vec2 Vesper::Random::RangeF2 (const glm::vec2 &minRange, const glm::vec2 &maxRange)
- glm::vec3 Vesper::Random::Float3 ()
- glm::vec3 Vesper::Random::RangeF3 (float min, float max)
- glm::vec3 Vesper::Random::RangeF3 (float min1, float max1, float min2, float max2, float min3, float max3)
- glm::vec3 Vesper::Random::RangeF3 (const glm::vec2 &range1, const glm::vec2 &range2, const glm::vec2 &range3)
- glm::vec4 Vesper::Random::Float4 ()
- glm::vec4 Vesper::Random::RangeF4 (float min, float max)

10.75 Random.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <random>
00003 #include <algorithm>
00004 #include <cstdint>
00005 #include <glm/glm.hpp>
00006 #include "Vesper/Debug/Instrumentor.h"
00007
00008 namespace Vesper {
00009
00010     namespace Random {
00011
00012         inline std::mt19937& GetRNG() {
00013             static thread_local std::mt19937 rng{ std::random_device{}() };
00014             return rng;
00015         }
00016
00017         inline void Seed(uint32_t seed) {
00018             GetRNG().seed(seed);
00019         }
00020
00021         inline uint32_t UInt1(uint32_t max) {
00022             VZ_PROFILE_FUNCTION();
00023             std::uniform_int_distribution<uint32_t> dist(0, max - 1);
00024             return dist(GetRNG());
00025         }
00026
00027         inline bool Bool1(float trueChance) {
00028             VZ_PROFILE_FUNCTION();
00029             std::uniform_real_distribution<float> dist(0.0f, 1.0f);
00030             return dist(GetRNG()) < trueChance;
00031         }
00032
00033         inline unsigned char Char() {
00034             VZ_PROFILE_FUNCTION();
00035             std::uniform_int_distribution<int> dist(0, 255);
00036             return static_cast<unsigned char>(dist(GetRNG()));
00037         }
00038
00039         inline std::string String(size_t length) {
00040             VZ_PROFILE_FUNCTION();
00041             const char charset[] =
00042                 "0123456789"
00043                 "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
00044                 "abcdefghijklmnopqrstuvwxyz";
00045             const size_t max_index = (sizeof(charset) - 1);
00046             std::string str(length, 0);
00047             for (size_t i = 0; i < length; ++i) {
00048                 str[i] = charset[UInt1(static_cast<uint32_t>(max_index))];
```

```
00049             }
00050             return str;
00051         }
00052 
00053         inline std::string HexString(size_t length) {
00054             VZ_PROFILE_FUNCTION();
00055             const char charset[] =
00056                 "0123456789"
00057                 "ABCDEF";
00058             const size_t max_index = (sizeof(charset) - 1);
00059             std::string str(length, 0);
00060             for (size_t i = 0; i < length; ++i) {
00061                 str[i] = charset[UInt1(static_cast<uint32_t>(max_index))];
00062             }
00063             return str;
00064         }
00065 
00066         inline std::string UUID() {
00067             VZ_PROFILE_FUNCTION();
00068             return HexString(8) + "-" + HexString(4) + "-" + HexString(4) + "-" + HexString(4) + "-" +
      HexString(12);
00069         }
00070 
00071         // Returns a float in the range [0.0, 1.0] = [inclusive, exclusive]
00072         inline float Float1() {
00073             VZ_PROFILE_FUNCTION();
00074             static thread_local std::uniform_real_distribution<float> dist(0.0f, 1.0f);
00075             return dist(GetRNG());
00076         }
00077 
00078         // Returns a float in the range [min, max]
00079         // If you need max to be possible, use the "_Inclusive" variant which advances the upper bound
00080         inline float RangeF1(float min, float max) {
00081             VZ_PROFILE_FUNCTION();
00082             if (min > max) std::swap(min, max);
00083             std::uniform_real_distribution<float> dist(min, max);
00084             return dist(GetRNG());
00085         }
00086 
00087         // Inclusive-upper variant: returns value in [min, max] (max possible)
00088         inline float RangeF1_Inclusive(float min, float max) {
00089             if (min > max) std::swap(min, max);
00090             float upper = std::nextafter(max, std::numeric_limits<float>::infinity());
00091             std::uniform_real_distribution<float> dist(min, upper);
00092             return dist(GetRNG());
00093         }
00094 
00095         // Returns a 2D float vector with each component in the range [0.0, 1.0] = [inclusive,
      exclusive]
00096         inline glm::vec2 Float2() {
00097             return glm::vec2{ Float1(), Float1() };
00098         }
00099 
00100         // Returns a 2D float vector with each component in the range [min, max]
00101         inline glm::vec2 RangeF2(float min, float max) {
00102             return glm::vec2{ RangeF1(min, max), RangeF1(min, max) };
00103         }
00104 
00105         // Returns a 2D float vector with each component in the range defined by the components of the
      given vector
00106         inline glm::vec2 RangeF2(float min1, float max1, float min2, float max2) {
00107             return glm::vec2{ RangeF1(min1, max1), RangeF1(min2, max2) };
00108         }
00109 
00110         // Returns a 2D float vector with each component in the specified ranges
00111         inline glm::vec2 RangeF2(const glm::vec2& minRange, const glm::vec2& maxRange) {
00112             return glm::vec2{ RangeF1(minRange.x, maxRange.x), RangeF1(minRange.y, maxRange.y) };
00113         }
00114 
00115 
00116         // Returns a 3D float vector with each component in the range [0.0, 1.0] = [inclusive,
      exclusive]
00117         inline glm::vec3 Float3() {
00118             return glm::vec3{ Float1(), Float1(), Float1() };
00119         }
00120 
00121         // Returns a 3D float vector with each component in the range [min, max]
00122         inline glm::vec3 RangeF3(float min, float max) {
00123             return glm::vec3{ RangeF1(min, max), RangeF1(min, max), RangeF1(min, max) };
00124         }
00125 
00126         // Returns a 3D float vector with each component in the specified ranges
00127         inline glm::vec3 RangeF3(float min1, float max1, float min2, float max2, float min3, float
      max3) {
00128             return glm::vec3{ RangeF1(min1, max1), RangeF1(min2, max2), RangeF1(min3, max3) };
00129         }
00130 
```

```
00131         // Returns a 3D float vector with each component in the specified ranges
00132         inline glm::vec3 RangeF3(const glm::vec2& range1, const glm::vec2& range2, const glm::vec2&
      range3) {
00133             return glm::vec3{ RangeF1(range1.x, range1.y), RangeF1(range2.x, range2.y),
      RangeF1(range3.x, range3.y) };
00134         }
00135
00136         // Returns a 4D float vector with each component in the range [0.0, 1.0] = [inclusive,
      exclusive]
00137         inline glm::vec4 Float4() {
00138             return glm::vec4{ Float1(), Float1(), Float1(), Float1() };
00139         }
00140
00141         // Returns a 4D float vector with each component in the range [min, max]
00142         inline glm::vec4 RangeF4(float min, float max) {
00143             return glm::vec4{ RangeF1(min, max), RangeF1(min, max), RangeF1(min, max), RangeF1(min,
      max) };
00144         }
00145
00146
00147
00148
00149     }
00150 }
```

10.76 Vesper/src/Vesper/Core/Timer.h File Reference

```
#include <chrono>
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.77 Timer.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <chrono>
00003
00004 namespace Vesper
00005 {
00006     /// TODO: Finish timer class
00007
00008
00009
00010 }
```

10.78 Vesper/src/Vesper/Core/Timestep.h File Reference

Classes

- class Vesper::Timestep

Namespaces

- namespace Vesper
  TEMPORARY.

10.79 Timestep.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 namespace Vesper {
00004
00005     class Timestep
00006     {
00007     public:
00008         Timestep(float time = 0.0f)
00009             : m_Time(time)
00010         {
00011         }
00012
00013         operator float() const { return m_Time; }
00014         float GetSeconds() const { return m_Time; }
00015         float GetMilliseconds() const { return m_Time * 1000.0f; }
00016     private:
00017         float m_Time;
00018     };
00019 }
```

10.80 Vesper/src/Vesper/Debug/Instrumentor.h File Reference

```
#include "Vesper/Core/Log.h"
#include <algorithm>
#include <chrono>
#include <fstream>
#include <iomanip>
#include <string>
#include <thread>
#include <mutex>
#include <sstream>
```

Classes

- struct Vesper::ProfileResult
- struct Vesper::InstrumentationSession
- class Vesper::Instrumentor
- class Vesper::InstrumentationTimer
- struct InstrumentorUtils::ChangeResult< N >

Namespaces

- namespace Vesper

  TEMPORARY.
- namespace InstrumentorUtils

Macros

- #define VZ_PROFILE 1
- #define VZ_FUNC_SIG "VZ_FUNC_SIG unknown!"
- #define VZ_PROFILE_BEGIN_SESSION(name, filepath)
- #define VZ_PROFILE_END_SESSION()
- #define VZ_PROFILE_SCOPE_LINE2(name, line)
- #define VZ_PROFILE_SCOPE_LINE(name, line)
- #define VZ_PROFILE_SCOPE(name)
- #define VZ_PROFILE_FUNCTION()

Typedefs

- using Vesper::FloatingPointMicroseconds = std::chrono::duration<double, std::micro>

Functions

- template<size_t N, size_t K>
  constexpr auto InstrumentorUtils::CleanupOutputString (const char(&expr)[N], const char(&remove)[K])

10.80.1 Class Documentation

10.80.1.1 struct Vesper::ProfileResult

Class Members

long long	End	
string	Name	
long long	Start	
uint32_t	ThreadID	

10.80.1.2 struct Vesper::InstrumentationSession

Class Members

string	Name	

10.80.1.3 struct InstrumentorUtils::ChangeResult

template<size_t N>
struct InstrumentorUtils::ChangeResult< N >

Class Members

char	Data↩ [N]	

10.80.2 Macro Definition Documentation

10.80.2.1 VZ_FUNC_SIG

```
#define VZ_FUNC_SIG "VZ_FUNC_SIG unknown!"
```

10.80.2.2 VZ_PROFILE

```
#define VZ_PROFILE 1
```

10.80.2.3 VZ_PROFILE_BEGIN_SESSION

```
#define VZ_PROFILE_BEGIN_SESSION(
              name,
              filepath)
```

Value:

```
::Vesper::Instrumentor::Get().BeginSession(name, filepath)
```

10.80.2.4 VZ_PROFILE_END_SESSION

```
#define VZ_PROFILE_END_SESSION()
```

Value:

```
::Vesper::Instrumentor::Get().EndSession()
```

10.80.2.5 VZ_PROFILE_FUNCTION

```
#define VZ_PROFILE_FUNCTION()
```

Value:

```
VZ_PROFILE_SCOPE(VZ_FUNC_SIG)
```

10.80.2.6 VZ_PROFILE_SCOPE

```
#define VZ_PROFILE_SCOPE(
              name)
```

Value:

```
VZ_PROFILE_SCOPE_LINE(name, __LINE__)
```

10.80.2.7 VZ_PROFILE_SCOPE_LINE

```
#define VZ_PROFILE_SCOPE_LINE(
              name,
              line)
```

Value:

```
VZ_PROFILE_SCOPE_LINE2(name, line)
```

10.80.2.8 VZ_PROFILE_SCOPE_LINE2

```
#define VZ_PROFILE_SCOPE_LINE2(
                name,
                line)
```

Value:

```
constexpr auto fixedName##line = InstrumentorUtils::CleanupOutputString(name, "__cdecl ");\
::Vesper::InstrumentationTimer timer##line(fixedName##line.Data)
```

```
00230     #define VZ_PROFILE_SCOPE_LINE2(name, line) constexpr auto fixedName##line =
      InstrumentorUtils::CleanupOutputString(name, "__cdecl ");\
00231                                                ::Vesper::InstrumentationTimer
      timer##line(fixedName##line.Data)
```

10.81 Instrumentor.h

Go to the documentation of this file.

```
00001 //Copyright  TheCherno
00002 //Modifications Copyright  2025 Damon S.Green II(nomad_ii_damon)
00003 //
00004 //Licensed under the Apache License, Version 2.0 (the "License");
00005 //you may not use this file except in compliance with the License.
00006 //You may obtain a copy of the License at
00007 //
00008 //[Apache](http://www.apache.org/licenses/LICENSE-2.0)
00009
00010 #pragma once
00011
00012 #include "Vesper/Core/Log.h"
00013
00014 #include <algorithm>
00015 #include <chrono>
00016 #include <fstream>
00017 #include <iomanip>
00018 #include <string>
00019 #include <thread>
00020 #include <mutex>
00021 #include <sstream>
00022
00023 namespace Vesper {
00024
00025     using FloatingPointMicroseconds = std::chrono::duration<double, std::micro>;
00026
00027
00028     struct ProfileResult
00029     {
00030         std::string Name;
00031         long long Start, End;
00032         uint32_t ThreadID;
00033     };
00034
00035     struct InstrumentationSession
00036     {
00037         std::string Name;
00038     };
00039
00040     class Instrumentor
00041     {
00042     private:
00043         InstrumentationSession* m_CurrentSession;
00044         std::ofstream m_OutputStream;
00045         std::mutex m_Mutex;
00046     public:
00047         Instrumentor()
00048             : m_CurrentSession(nullptr)
00049         {
00050         }
00051
00052         void BeginSession(const std::string& name, const std::string& filepath = "results.json")
00053         {
00054             std::lock_guard lock(m_Mutex);
00055             if (m_CurrentSession) {
00056                 // If there is already a current session, end it and start new one
00057                 if (Log::GetCoreLogger())
00058                 {
```

```
00059                   VZ_CORE_ERROR("Instrumentor::BeginSession('{0}') when session '{1}' already
      open.", name, m_CurrentSession->Name);
00060                   }
00061                   InternalEndSession();
00062               }
00063               m_OutputStream.open(filepath);
00064               if (m_OutputStream.is_open()) {
00065                   m_CurrentSession = new InstrumentationSession{ name };
00066                   WriteHeader();
00067               }
00068               else {
00069                   if (Log::GetCoreLogger())
00070                   {
00071                       VZ_CORE_ERROR("Instrumentor could not open results file '{0}'.", filepath);
00072                   }
00073               }
00074               WriteHeader();
00075               m_CurrentSession = new InstrumentationSession{ name };
00076           }
00077
00078           void EndSession()
00079           {
00080               std::lock_guard lock(m_Mutex);
00081               InternalEndSession();
00082           }
00083
00084           void WriteProfile(const ProfileResult& result)
00085           {
00086               m_OutputStream << ",";
00087
00088               std::string name = result.Name;
00089               std::replace(name.begin(), name.end(), '"', '\'');
00090
00091               m_OutputStream << "{";
00092               m_OutputStream << "\"cat\":\"function\",";
00093               m_OutputStream << "\"dur\":" << (result.End - result.Start) << ',';
00094               m_OutputStream << "\"name\":\"" << name << "\",";
00095               m_OutputStream << "\"ph\":\"X\",";
00096               m_OutputStream << "\"pid\":0,";
00097               m_OutputStream << "\"tid\":" << result.ThreadID << ",";
00098               m_OutputStream << "\"ts\":" << result.Start;
00099               m_OutputStream << "}";
00100
00101               std::lock_guard lock(m_Mutex);
00102               if (m_CurrentSession)
00103               {
00104                   //m_OutputStream << json.str();
00105                   m_OutputStream.flush();
00106               }
00107           }
00108
00109           void WriteHeader()
00110           {
00111               m_OutputStream << "{\"otherData\": {},\"traceEvents\":[";
00112               m_OutputStream.flush();
00113           }
00114
00115           void WriteFooter()
00116           {
00117               m_OutputStream << "]}";
00118               m_OutputStream.flush();
00119           }
00120
00121           void InternalEndSession()
00122           {
00123               if (m_CurrentSession)
00124               {
00125                   WriteFooter();
00126                   m_OutputStream.close();
00127                   delete m_CurrentSession;
00128                   m_CurrentSession = nullptr;
00129               }
00130           }
00131
00132           static Instrumentor& Get()
00133           {
00134               static Instrumentor instance;
00135               return instance;
00136           }
00137       };
00138
00139       class InstrumentationTimer
00140       {
00141       public:
00142           InstrumentationTimer(const char* name)
00143               : m_Name(name), m_Stopped(false)
00144           {
```

```
00145             m_StartTimepoint = std::chrono::high_resolution_clock::now();
00146         }
00147 
00148         ~InstrumentationTimer()
00149         {
00150             if (!m_Stopped)
00151                 Stop();
00152         }
00153 
00154         void Stop()
00155         {
00156             auto endTimepoint = std::chrono::high_resolution_clock::now();
00157 
00158             long long start =
      std::chrono::time_point_cast<std::chrono::microseconds>(m_StartTimepoint).time_since_epoch().count();
00159             long long end =
      std::chrono::time_point_cast<std::chrono::microseconds>(endTimepoint).time_since_epoch().count();
00160 
00161             uint32_t threadID = std::hash<std::thread::id>{}(std::this_thread::get_id());
00162             Instrumentor::Get().WriteProfile({ m_Name, start, end, threadID });
00163 
00164             m_Stopped = true;
00165         }
00166     private:
00167         const char* m_Name;
00168         std::chrono::time_point<std::chrono::high_resolution_clock> m_StartTimepoint;
00169         bool m_Stopped;
00170     };
00171 
00172 
00173 }
00174 
00175 namespace InstrumentorUtils {
00176 
00177     template <size_t N>
00178     struct ChangeResult
00179     {
00180         char Data[N];
00181     };
00182 
00183     template <size_t N, size_t K>
00184     constexpr auto CleanupOutputString(const char(&expr)[N], const char(&remove)[K])
00185     {
00186         ChangeResult<N> result = {};
00187 
00188         size_t srcIndex = 0;
00189         size_t dstIndex = 0;
00190         while (srcIndex < N)
00191         {
00192             size_t matchIndex = 0;
00193             while (matchIndex < K - 1 && srcIndex + matchIndex < N - 1 && expr[srcIndex + matchIndex]
      == remove[matchIndex])
00194                 matchIndex++;
00195             if (matchIndex == K - 1)
00196                 srcIndex += matchIndex;
00197             result.Data[dstIndex++] = expr[srcIndex] == '"' ? '\"' : expr[srcIndex];
00198             srcIndex++;
00199         }
00200         return result;
00201     }
00202 }
00203 
00204 
00205 #define VZ_PROFILE 1
00206 #if VZ_PROFILE
00207     // Resolve which function signature macro will be used. Note that this only
00208     // is resolved when the (pre)compiler starts, so the syntax highlighting
00209     // could mark the wrong one in your editor!
00210     #if defined(__GNUC__) || (defined(__MWERKS__) && (__MWERKS__ >= 0x3000)) || (defined(__ICC) &&
      (__ICC >= 600)) || defined(__ghs__)
00211     #define VZ_FUNC_SIG __PRETTY_FUNCTION__
00212     #elif defined(__DMC__) && (__DMC__ >= 0x810)
00213     #define VZ_FUNC_SIG __PRETTY_FUNCTION__
00214     #elif (defined(__FUNCSIG__) || (_MSC_VER))
00215     #define VZ_FUNC_SIG __FUNCSIG__
00216     #elif (defined(__INTEL_COMPILER) && (__INTEL_COMPILER >= 600)) || (defined(__IBMCPP__) &&
      (__IBMCPP__ >= 500))
00217     #define VZ_FUNC_SIG __FUNCTION__
00218     #elif defined(__BORLANDC__) && (__BORLANDC__ >= 0x550)
00219     #define VZ_FUNC_SIG __FUNC__
00220     #elif defined(__STDC_VERSION__) && (__STDC_VERSION__ >= 199901)
00221     #define VZ_FUNC_SIG __func__
00222     #elif defined(__cplusplus) && (__cplusplus >= 201103)
00223     #define VZ_FUNC_SIG __func__
00224     #else
00225     #define VZ_FUNC_SIG "VZ_FUNC_SIG unknown!"
00226     #endif
```

```
00227
00228     #define VZ_PROFILE_BEGIN_SESSION(name, filepath) ::Vesper::Instrumentor::Get().BeginSession(name,
      filepath)
00229     #define VZ_PROFILE_END_SESSION() ::Vesper::Instrumentor::Get().EndSession()
00230     #define VZ_PROFILE_SCOPE_LINE2(name, line) constexpr auto fixedName##line =
      InstrumentorUtils::CleanupOutputString(name, "__cdecl ");
00231                                                ::Vesper::InstrumentationTimer
      timer##line(fixedName##line.Data)
00232     #define VZ_PROFILE_SCOPE_LINE(name, line) VZ_PROFILE_SCOPE_LINE2(name, line)
00233     #define VZ_PROFILE_SCOPE(name) VZ_PROFILE_SCOPE_LINE(name, __LINE__)
00234     #define VZ_PROFILE_FUNCTION() VZ_PROFILE_SCOPE(VZ_FUNC_SIG)
00235 #else
00236     #define VZ_PROFILE_BEGIN_SESSION(name, filepath)
00237     #define VZ_PROFILE_END_SESSION()
00238     #define VZ_PROFILE_FUNCTION()
00239     #define VZ_PROFILE_SCOPE(name)
00240 #endif
```

10.82 Vesper/src/Vesper/Events/ApplicationEvent.h File Reference

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
```

Classes

- class Vesper::WindowResizeEvent
- class Vesper::WindowCloseEvent
- class Vesper::AppTickEvent
- class Vesper::AppUpdateEvent
- class Vesper::AppRenderEvent

Namespaces

- namespace Vesper

  TEMPORARY.

10.83 ApplicationEvent.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "vzpch.h"
00004 #include "Vesper/Core/Base.h"
00005 #include "Vesper/Events/Event.h"
00006
00007 namespace Vesper {
00008
00009     class VESPER_API WindowResizeEvent : public Event
00010     {
00011     public:
00012         WindowResizeEvent(unsigned int width, unsigned int height)
00013             : m_Width(width), m_Height(height) {}
00014
00015         unsigned int GetWidth() const { return m_Width; }
00016         unsigned int GetHeight() const { return m_Height; }
00017
00018         std::string ToString() const override
00019         {
00020             std::stringstream ss;
00021             ss « "WindowResizeEvent: " « m_Width « ", " « m_Height;
```

```
00022             return ss.str();
00023         }
00024
00025         EVENT_CLASS_TYPE(WindowResize)
00026         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00027
00028     private:
00029         unsigned int m_Width, m_Height;
00030     };
00031
00032     class VESPER_API WindowCloseEvent : public Event
00033     {
00034     public:
00035         WindowCloseEvent() {}
00036
00037         EVENT_CLASS_TYPE(WindowClose)
00038         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00039     };
00040
00041     class VESPER_API AppTickEvent : public Event
00042     {
00043     public:
00044         AppTickEvent() {}
00045
00046         EVENT_CLASS_TYPE(AppTick)
00047         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00048     };
00049
00050     class VESPER_API AppUpdateEvent : public Event
00051     {
00052     public:
00053         AppUpdateEvent() {}
00054
00055         EVENT_CLASS_TYPE(AppUpdate)
00056         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00057     };
00058
00059     class VESPER_API AppRenderEvent : public Event
00060     {
00061     public:
00062         AppRenderEvent() {}
00063
00064         EVENT_CLASS_TYPE(AppRender)
00065         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00066     };
00067 }
```

10.84 Vesper/src/Vesper/Events/Event.h File Reference

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
```

Classes

- class Vesper::Event
- class Vesper::EventDispatcher

Namespaces

- namespace Vesper

  TEMPORARY.

Macros

- #define EVENT_CLASS_TYPE(type)
- #define EVENT_CLASS_CATEGORY(category)

Enumerations

- enum class Vesper::EventType {
  Vesper::None = 0 , Vesper::WindowClose , Vesper::WindowResize , Vesper::WindowFocus ,
  Vesper::WindowLostFocus , Vesper::WindowMoved , Vesper::AppTick , Vesper::AppUpdate ,
  Vesper::AppRender , Vesper::KeyPressed , Vesper::KeyReleased , Vesper::KeyTyped ,
  Vesper::MouseButtonPressed , Vesper::MouseButtonReleased , Vesper::MouseMoved , Vesper::MouseScrolled
  }
- enum Vesper::EventCategory {
  Vesper::None = 0 , Vesper::EventCategoryApplication = BIT(0) , Vesper::EventCategoryInput = BIT(1) ,
  Vesper::EventCategoryKeyboard = BIT(2) ,
  Vesper::EventCategoryMouse = BIT(3) , Vesper::EventCategoryMouseButton = BIT(4) }

Functions

- std::string Vesper::format_as (const Event &e)

10.84.1 Macro Definition Documentation

10.84.1.1 EVENT_CLASS_CATEGORY

```
#define EVENT_CLASS_CATEGORY(
              category)
```

Value:

```
virtual int GetCategoryFlags() const override { return category; }
```

10.84.1.2 EVENT_CLASS_TYPE

```
#define EVENT_CLASS_TYPE(
              type)
```

Value:

```
static EventType GetStaticType() { return EventType::##type; }\
virtual EventType GetEventType() const override { return GetStaticType(); }\
virtual const char* GetName() const override { return #type; }
```

```
00027 #define EVENT_CLASS_TYPE(type) static EventType GetStaticType() { return EventType::##type; }\
00028                                virtual EventType GetEventType() const override { return
      GetStaticType(); }\
00029                                virtual const char* GetName() const override { return #type; }
```

10.85 Event.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "vzpch.h"
00004 #include "Vesper/Core/Base.h"
00005
00006 namespace Vesper
00007 {
00008
00009     enum class EventType
00010     {
00011         None = 0,
00012         WindowClose, WindowResize, WindowFocus, WindowLostFocus, WindowMoved,
00013         AppTick, AppUpdate, AppRender,
00014         KeyPressed, KeyReleased, KeyTyped,
00015         MouseButtonPressed, MouseButtonReleased, MouseMoved, MouseScrolled
00016     };
00017     enum EventCategory
00018     {
00019         None = 0,
00020         EventCategoryApplication    = BIT(0),
00021         EventCategoryInput          = BIT(1),
00022         EventCategoryKeyboard       = BIT(2),
00023         EventCategoryMouse          = BIT(3),
00024         EventCategoryMouseButton    = BIT(4)
00025     };
00026
00027 #define EVENT_CLASS_TYPE(type) static EventType GetStaticType() { return EventType::##type; }
00028                                virtual EventType GetEventType() const override { return
      GetStaticType(); }
00029                                virtual const char* GetName() const override { return #type; }
00030
00031 #define EVENT_CLASS_CATEGORY(category) virtual int GetCategoryFlags() const override { return
      category; }
00032
00033     class VESPER_API Event
00034     {
00035         friend class EventDispatcher;
00036     public:
00037         virtual ~Event() = default;
00038         virtual EventType GetEventType() const = 0;
00039         virtual const char* GetName() const = 0;
00040         virtual int GetCategoryFlags() const = 0;
00041         virtual std::string ToString() const { return GetName(); }
00042
00043         inline bool IsInCategory(EventCategory category)
00044         {
00045             return GetCategoryFlags() & category;
00046         }
00047         bool Handled = false;
00048     };
00049
00050     class EventDispatcher
00051     {
00052         template<typename T>
00053         using EventFn = std::function<bool(T&)>;
00054     public:
00055         EventDispatcher(Event& event)
00056             : m_Event(event)
00057         {
00058         }
00059
00060         template<typename T>
00061         bool Dispatch(EventFn<T> func)
00062         {
00063             if (m_Event.GetEventType() == T::GetStaticType())
00064             {
00065                 m_Event.Handled = func(*(T*)&m_Event);
00066                 return true;
00067             }
00068             return false;
00069         }
00070     private:
00071         Event& m_Event;
00072     };
00073
00074     inline std::string format_as(const Event& e)
00075     {
00076         return e.ToString();
00077     }
00078 }
```

10.86 Vesper/src/Vesper/Events/KeyEvent.h File Reference

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
```

Classes

- class Vesper::KeyEvent
- class Vesper::KeyPressedEvent
- class Vesper::KeyReleasedEvent
- class Vesper::KeyTypedEvent

Namespaces

- namespace Vesper

  TEMPORARY.

10.87 KeyEvent.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "vzpch.h"
00004 #include "Vesper/Core/Base.h"
00005 #include "Vesper/Events/Event.h"
00006
00007 namespace Vesper {
00008
00009     class VESPER_API KeyEvent : public Event
00010     {
00011     public:
00012         inline int GetKeyCode() const { return m_KeyCode; }
00013         EVENT_CLASS_CATEGORY(EventCategoryKeyboard | EventCategoryInput)
00014     protected:
00015         KeyEvent(int keycode)
00016             : m_KeyCode(keycode) {}
00017         int m_KeyCode;
00018     };
00019
00020     class VESPER_API KeyPressedEvent : public KeyEvent
00021     {
00022     public:
00023         KeyPressedEvent(int keycode, int repeatCount)
00024             : KeyEvent(keycode), m_RepeatCount(repeatCount) {}
00025
00026         inline int GetRepeatCount() const { return m_RepeatCount; }
00027
00028         std::string ToString() const override
00029         {
00030             std::stringstream ss;
00031             ss << "KeyPressedEvent: " << m_KeyCode << " (" << m_RepeatCount << " repeats)";
00032             return ss.str();
00033         }
00034
00035         EVENT_CLASS_TYPE(KeyPressed)
00036     private:
00037         int m_RepeatCount;
00038     };
00039
00040     class VESPER_API KeyReleasedEvent : public KeyEvent
00041     {
00042     public:
00043         KeyReleasedEvent(int keycode)
00044             : KeyEvent(keycode) {}
00045
```

```
00046         std::string ToString() const override
00047         {
00048             std::stringstream ss;
00049             ss « "KeyReleasedEvent: " « m_KeyCode;
00050             return ss.str();
00051         }
00052 
00053         EVENT_CLASS_TYPE(KeyReleased)
00054     };
00055 
00056     class VESPER_API KeyTypedEvent : public KeyEvent
00057     {
00058     public:
00059         KeyTypedEvent(int keycode)
00060             : KeyEvent(keycode) {
00061         }
00062 
00063 
00064         std::string ToString() const override
00065         {
00066             std::stringstream ss;
00067             ss « "KeyTypedEvent: " « m_KeyCode;
00068             return ss.str();
00069         }
00070 
00071         EVENT_CLASS_TYPE(KeyTyped)
00072     };
00073 
00074 }
```

10.88 Vesper/src/Vesper/Events/MouseEvent.h File Reference

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
```

Classes

- class Vesper::MouseMovedEvent
- class Vesper::MouseScrolledEvent
- class Vesper::MouseButtonEvent
- class Vesper::MouseButtonPressedEvent
- class Vesper::MouseButtonReleasedEvent

Namespaces

- namespace Vesper

  TEMPORARY.

10.89 MouseEvent.h

Go to the documentation of this file.

```
00001 #pragma once
00002 
00003 #include "vzpch.h"
00004 #include "Vesper/Core/Base.h"
00005 #include "Vesper/Events/Event.h"
00006 
00007 namespace Vesper {
00008 
00009     class VESPER_API MouseMovedEvent : public Event
```

```
00010     {
00011     public:
00012
00013         MouseMovedEvent(float x, float y)
00014             : m_MouseX(x), m_MouseY(y) {
00015         }
00016
00017         inline float GetX() const { return m_MouseX; }
00018         inline float GetY() const { return m_MouseY; }
00019
00020         std::string ToString() const override
00021         {
00022             std::stringstream ss;
00023             ss << "MouseMovedEvent: " << m_MouseX << ", " << m_MouseY;
00024             return ss.str();
00025         }
00026
00027         EVENT_CLASS_TYPE(MouseMoved)
00028             EVENT_CLASS_CATEGORY(EventCategoryMouse | EventCategoryInput)
00029     private:
00030         float m_MouseX, m_MouseY;
00031     };
00032
00033     class VESPER_API MouseScrolledEvent : public Event
00034     {
00035     public:
00036         MouseScrolledEvent(float xOffset, float yOffset)
00037             : m_XOffset(xOffset), m_YOffset(yOffset) {
00038         }
00039
00040         inline float GetXOffset() const { return m_XOffset; }
00041         inline float GetYOffset() const { return m_YOffset; }
00042
00043         std::string ToString() const override
00044         {
00045             std::stringstream ss;
00046             ss << "MouseScrolledEvent: " << GetXOffset() << ", " << GetYOffset();
00047             return ss.str();
00048         }
00049
00050         EVENT_CLASS_TYPE(MouseScrolled)
00051             EVENT_CLASS_CATEGORY(EventCategoryMouse | EventCategoryInput)
00052     private:
00053         float m_XOffset, m_YOffset;
00054     };
00055
00056     class VESPER_API MouseButtonEvent : public Event
00057     {
00058     public:
00059         inline int GetMouseButton() const { return m_Button; }
00060
00061         EVENT_CLASS_CATEGORY(EventCategoryMouse | EventCategoryInput)
00062     protected:
00063         MouseButtonEvent(int button)
00064             : m_Button(button) {
00065         }
00066         int m_Button;
00067     };
00068
00069     class VESPER_API MouseButtonPressedEvent : public MouseButtonEvent
00070     {
00071     public:
00072         MouseButtonPressedEvent(int button)
00073             : MouseButtonEvent(button) {
00074         }
00075
00076         std::string ToString() const override
00077         {
00078             std::stringstream ss;
00079             ss << "MouseButtonPressedEvent: " << GetMouseButton();
00080             return ss.str();
00081         }
00082
00083         EVENT_CLASS_TYPE(MouseButtonPressed)
00084     };
00085
00086     class VESPER_API MouseButtonReleasedEvent : public MouseButtonEvent
00087     {
00088     public:
00089
00090         MouseButtonReleasedEvent(int button)
00091             : MouseButtonEvent(button) {
00092         }
00093
00094         std::string ToString() const override
00095         {
00096             std::stringstream ss;
```

```
00097             ss « "MouseButtonReleasedEvent: " « GetMouseButton();
00098             return ss.str();
00099         }
00100
00101         EVENT_CLASS_TYPE(MouseButtonReleased)
00102     };
00103
00104 }
```

10.90 Vesper/src/Vesper/ImGui/ImGuiBuild.cpp File Reference

```
#include "vzpch.h"
#include "backends/imgui_impl_opengl3.cpp"
#include "backends/imgui_impl_glfw.cpp"
```

Macros

- #define IMGUI_IMPL_OPENGL_LOADER_GLAD

10.90.1 Macro Definition Documentation

10.90.1.1 IMGUI_IMPL_OPENGL_LOADER_GLAD

```
#define IMGUI_IMPL_OPENGL_LOADER_GLAD
```

10.91 Vesper/src/Vesper/ImGui/ImGuiLayer.cpp File Reference

```
#include "vzpch.h"
#include "ImGuiLayer.h"
#include "imgui.h"
#include "backends/imgui_impl_glfw.h"
#include "backends/imgui_impl_opengl3.h"
#include "Vesper/App/Application.h"
#include "Vesper/App/Layer.h"
#include <GLFW/glfw3.h>
#include <glad/glad.h>
#include "ImGuizmo.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.92 Vesper/src/Vesper/ImGui/ImGuiLayer.h File Reference

```
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
#include "Vesper/Events/ApplicationEvent.h"
#include "Vesper/Events/KeyEvent.h"
#include "Vesper/Events/MouseEvent.h"
#include "Vesper/App/Layer.h"
```

Classes

- class Vesper::ImGuiLayer

Namespaces

- namespace Vesper

  TEMPORARY.

10.93 ImGuiLayer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Core/Base.h"
00004 #include "Vesper/Events/Event.h"
00005 #include "Vesper/Events/ApplicationEvent.h"
00006 #include "Vesper/Events/KeyEvent.h"
00007 #include "Vesper/Events/MouseEvent.h"
00008 #include "Vesper/App/Layer.h"
00009
00010 namespace Vesper {
00011
00012     class VESPER_API ImGuiLayer : public Layer
00013     {
00014     public:
00015         ImGuiLayer();
00016         ~ImGuiLayer();
00017
00018         virtual void OnAttach() override;
00019         virtual void OnDetach() override;
00020         virtual void OnImGuiRender() override;
00021         virtual void OnEvent(Event& e) override;
00022
00023         virtual void Begin();
00024         virtual void End();
00025
00026         virtual void SetBlockEvents(bool block) { m_BlockEvents = block; }
00027         virtual void SetDarkThemeColors();
00028     protected:
00029         bool m_BlockEvents = true;
00030         float m_Time = 0.0f;
00031     };
00032
00033 }
```

10.94 Vesper/src/Vesper/ImGui/VesperImGui.h File Reference

```
#include "imgui/imgui.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

Functions

- static void Vesper::DisplayVesperInfo_ImGui ()

10.95 VesperImGui.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "imgui/imgui.h"
00004
00005 namespace Vesper {
00006
00007     static void DisplayVesperInfo_ImGui()
00008     {
00009         ImGui::Begin("Vesper Info");
00010
00011         if (ImGui::TreeNode("About Vesper"))
00012         {
00013             ImGui::Text("Vesper Engine");
00014             ImGui::Text("Version: 0.1.0");
00015             ImGui::Text("Author: Damon Green II");
00016             ImGui::Text("GitHub: https://github.com/nomadiidamon/Vesper");
00017             ImGui::Separator();
00018
00019             ImGui::Text("Status: ");
00020             ImGui::Text("\tEarly Development of API and 2D Renderer");
00021             ImGui::Separator();
00022
00023             ImGui::TextWrapped("Vesper is a cross-platform game engine currently in early development.
      The engine is being built from the ground up with a focus on modularity, performance, and ease of use.
      The goal of Vesper is to provide developers with a powerful and flexible toolset for creating games
      and interactive applications.");
00024             ImGui::Separator();
00025
00026             if (ImGui::TreeNode("Controls:"))
00027             {
00028                 ImGui::Text("\tWASD: Move Camera");
00029                 ImGui::Text("\tQ/E: Rotate Camera (if enabled {see settings})");
00030                 ImGui::Text("\tScroll Wheel: Zoom Camera");
00031                 ImGui::TreePop();
00032             }
00033             ImGui::Separator();
00034
00035             if (ImGui::TreeNode("RoadMap")) {
00036
00037                 if (ImGui::TreeNode("Current Features:"))
00038                 {
00039                     ImGui::Text("\t- Cross-Platform Design");
00040                     ImGui::Text("\t\t- Currently Windows only");
00041                     ImGui::Text("\t- OpenGL Renderer");
00042                     ImGui::Text("\t- Orthographic Camera");
00043                     ImGui::Text("\t- Shader System");
00044                     ImGui::Text("\t- Texture Loading");
00045                     ImGui::Text("\t- ImGui Integration");
00046                     ImGui::Text("\t\t- Current settings panel adjusts camera parameters!");
00047
00048                     ImGui::TreePop();
00049                 }
00050                 ImGui::Separator();
00051
00052                 if (ImGui::TreeNode("In Progress:"))
00053                 {
00054                     ImGui::Text("\t- 2D Rendering Features");
00055                     ImGui::Text("\t\t- Sprites");
00056                     ImGui::Text("\t\t- Sprite Sheets");
00057                     ImGui::Text("\t\t- Animation");
00058                     ImGui::TreePop();
00059                 }
00060                 ImGui::Separator();
```

```
00061
00062                 if (ImGui::TreeNode("Planned Features:"))
00063                 {
00064                     ImGui::Text("\t- Vulkan Renderer");
00065                     ImGui::Text("\t- 2D Editor");
00066                     ImGui::Text("\t- 2D Particles");
00067                     ImGui::Text("\t- Audio");
00068                     ImGui::Text("\t- Timelining");
00069                     ImGui::Text("\t- Video Playback");
00070                     ImGui::Text("\t- 3D Renderer");
00071                     ImGui::Text("\t- 3D Particles");
00072                     ImGui::TreePop();
00073                 }
00074                 ImGui::TreePop();
00075             }
00076
00077             ImGui::TreePop();
00078         }
00079         ImGui::End();
00080     }
00081
00082
00083 }
```

10.96 Vesper/src/Vesper/ImGuizmo/ImGuizmoBuild.cpp File Reference

```
#include "vzpch.h"
```

10.97 Vesper/src/Vesper/Input/Input.h File Reference

```
#include "Vesper/Core/Base.h"
#include <glm/glm.hpp>
```

Classes

- class Vesper::Input

Namespaces

- namespace Vesper

  TEMPORARY.

10.98 Input.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Core/Base.h"
00004 #include <glm/glm.hpp>
00005
00006
00007 namespace Vesper {
00008
00009     class VESPER_API Input
00010     {
00011     protected:
00012         Input() = default;
```

```
00013     public:
00014         Input(const Input&) = delete;
00015         Input& operator=(const Input&) = delete;
00016 
00017         static bool IsKeyPressed(int keycode);
00018 
00019         static bool IsMouseButtonPressed(int button);
00020         static float GetMouseX();
00021         static float GetMouseY();
00022         static glm::vec2 GetMousePosition();
00023     };
00024 }
```

10.99 Vesper/src/Vesper/Input/KeyCodes.h File Reference

Macros

- #define VZ_KEY_SPACE 32
- #define VZ_KEY_APOSTROPHE 39 /* ' */
- #define VZ_KEY_COMMA 44 /* , */
- #define VZ_KEY_MINUS 45 /* - */
- #define VZ_KEY_PERIOD 46 /* . */
- #define VZ_KEY_SLASH 47 /* / */
- #define VZ_KEY_0 48
- #define VZ_KEY_1 49
- #define VZ_KEY_2 50
- #define VZ_KEY_3 51
- #define VZ_KEY_4 52
- #define VZ_KEY_5 53
- #define VZ_KEY_6 54
- #define VZ_KEY_7 55
- #define VZ_KEY_8 56
- #define VZ_KEY_9 57
- #define VZ_KEY_SEMICOLON 59 /* ; */
- #define VZ_KEY_EQUAL 61 /* = */
- #define VZ_KEY_A 65
- #define VZ_KEY_B 66
- #define VZ_KEY_C 67
- #define VZ_KEY_D 68
- #define VZ_KEY_E 69
- #define VZ_KEY_F 70
- #define VZ_KEY_G 71
- #define VZ_KEY_H 72
- #define VZ_KEY_I 73
- #define VZ_KEY_J 74
- #define VZ_KEY_K 75
- #define VZ_KEY_L 76
- #define VZ_KEY_M 77
- #define VZ_KEY_N 78
- #define VZ_KEY_O 79
- #define VZ_KEY_P 80
- #define VZ_KEY_Q 81
- #define VZ_KEY_R 82
- #define VZ_KEY_S 83
- #define VZ_KEY_T 84
- #define VZ_KEY_U 85
- #define VZ_KEY_V 86

- #define VZ_KEY_W 87
- #define VZ_KEY_X 88
- #define VZ_KEY_Y 89
- #define VZ_KEY_Z 90
- #define VZ_KEY_LEFT_BRACKET 91 /* [*/
- #define VZ_KEY_BACKSLASH 92 /* \ */
- #define VZ_KEY_RIGHT_BRACKET 93 /*] */
- #define VZ_KEY_GRAVE_ACCENT 96 /* ` */
- #define VZ_KEY_WORLD_1 161 /* non-US #1 */
- #define VZ_KEY_WORLD_2 162 /* non-US #2 */
- #define VZ_KEY_ESCAPE 256
- #define VZ_KEY_ENTER 257
- #define VZ_KEY_TAB 258
- #define VZ_KEY_BACKSPACE 259
- #define VZ_KEY_INSERT 260
- #define VZ_KEY_DELETE 261
- #define VZ_KEY_RIGHT 262
- #define VZ_KEY_LEFT 263
- #define VZ_KEY_DOWN 264
- #define VZ_KEY_UP 265
- #define VZ_KEY_PAGE_UP 266
- #define VZ_KEY_PAGE_DOWN 267
- #define VZ_KEY_HOME 268
- #define VZ_KEY_END 269
- #define VZ_KEY_CAPS_LOCK 280
- #define VZ_KEY_SCROLL_LOCK 281
- #define VZ_KEY_NUM_LOCK 282
- #define VZ_KEY_PRINT_SCREEN 283
- #define VZ_KEY_PAUSE 284
- #define VZ_KEY_F1 290
- #define VZ_KEY_F2 291
- #define VZ_KEY_F3 292
- #define VZ_KEY_F4 293
- #define VZ_KEY_F5 294
- #define VZ_KEY_F6 295
- #define VZ_KEY_F7 296
- #define VZ_KEY_F8 297
- #define VZ_KEY_F9 298
- #define VZ_KEY_F10 299
- #define VZ_KEY_F11 300
- #define VZ_KEY_F12 301
- #define VZ_KEY_F13 302
- #define VZ_KEY_F14 303
- #define VZ_KEY_F15 304
- #define VZ_KEY_F16 305
- #define VZ_KEY_F17 306
- #define VZ_KEY_F18 307
- #define VZ_KEY_F19 308
- #define VZ_KEY_F20 309
- #define VZ_KEY_F21 310
- #define VZ_KEY_F22 311
- #define VZ_KEY_F23 312
- #define VZ_KEY_F24 313
- #define VZ_KEY_F25 314
- #define VZ_KEY_KP_0 320

- #define VZ_KEY_KP_1 321
- #define VZ_KEY_KP_2 322
- #define VZ_KEY_KP_3 323
- #define VZ_KEY_KP_4 324
- #define VZ_KEY_KP_5 325
- #define VZ_KEY_KP_6 326
- #define VZ_KEY_KP_7 327
- #define VZ_KEY_KP_8 328
- #define VZ_KEY_KP_9 329
- #define VZ_KEY_KP_DECIMAL 330
- #define VZ_KEY_KP_DIVIDE 331
- #define VZ_KEY_KP_MULTIPLY 332
- #define VZ_KEY_KP_SUBTRACT 333
- #define VZ_KEY_KP_ADD 334
- #define VZ_KEY_KP_ENTER 335
- #define VZ_KEY_KP_EQUAL 336
- #define VZ_KEY_LEFT_SHIFT 340
- #define VZ_KEY_LEFT_CONTROL 341
- #define VZ_KEY_LEFT_ALT 342
- #define VZ_KEY_LEFT_SUPER 343
- #define VZ_KEY_RIGHT_SHIFT 344
- #define VZ_KEY_RIGHT_CONTROL 345
- #define VZ_KEY_RIGHT_ALT 346
- #define VZ_KEY_RIGHT_SUPER 347
- #define VZ_KEY_MENU 348

10.99.1 Macro Definition Documentation

10.99.1.1 VZ_KEY_0

```
#define VZ_KEY_0 48
```

10.99.1.2 VZ_KEY_1

```
#define VZ_KEY_1 49
```

10.99.1.3 VZ_KEY_2

```
#define VZ_KEY_2 50
```

10.99.1.4 VZ_KEY_3

```
#define VZ_KEY_3 51
```

10.99.1.5 VZ_KEY_4

```
#define VZ_KEY_4 52
```

10.99.1.6 VZ_KEY_5

```
#define VZ_KEY_5 53
```

10.99.1.7 VZ_KEY_6

```
#define VZ_KEY_6 54
```

10.99.1.8 VZ_KEY_7

```
#define VZ_KEY_7 55
```

10.99.1.9 VZ_KEY_8

```
#define VZ_KEY_8 56
```

10.99.1.10 VZ_KEY_9

```
#define VZ_KEY_9 57
```

10.99.1.11 VZ_KEY_A

```
#define VZ_KEY_A 65
```

10.99.1.12 VZ_KEY_APOSTROPHE

```
#define VZ_KEY_APOSTROPHE 39 /* ' */
```

10.99.1.13 VZ_KEY_B

```
#define VZ_KEY_B 66
```

10.99.1.14 VZ_KEY_BACKSLASH

```
#define VZ_KEY_BACKSLASH 92 /* \ */
```

10.99.1.15 VZ_KEY_BACKSPACE

```
#define VZ_KEY_BACKSPACE 259
```

10.99.1.16 VZ_KEY_C

```
#define VZ_KEY_C 67
```

10.99.1.17 VZ_KEY_CAPS_LOCK

```
#define VZ_KEY_CAPS_LOCK 280
```

10.99.1.18 VZ_KEY_COMMA

```
#define VZ_KEY_COMMA 44 /* , */
```

10.99.1.19 VZ_KEY_D

```
#define VZ_KEY_D 68
```

10.99.1.20 VZ_KEY_DELETE

```
#define VZ_KEY_DELETE 261
```

10.99.1.21 VZ_KEY_DOWN

```
#define VZ_KEY_DOWN 264
```

10.99.1.22 VZ_KEY_E

```
#define VZ_KEY_E 69
```

10.99.1.23 VZ_KEY_END

```
#define VZ_KEY_END 269
```

10.99.1.24 VZ_KEY_ENTER

```
#define VZ_KEY_ENTER 257
```

10.99.1.25 VZ_KEY_EQUAL

```
#define VZ_KEY_EQUAL 61 /* = */
```

10.99.1.26 VZ_KEY_ESCAPE

```
#define VZ_KEY_ESCAPE 256
```

10.99.1.27 VZ_KEY_F

```
#define VZ_KEY_F 70
```

10.99.1.28 VZ_KEY_F1

```
#define VZ_KEY_F1 290
```

10.99.1.29 VZ_KEY_F10

```
#define VZ_KEY_F10 299
```

10.99.1.30 VZ_KEY_F11

```
#define VZ_KEY_F11 300
```

10.99.1.31 VZ_KEY_F12

```
#define VZ_KEY_F12 301
```

10.99.1.32 VZ_KEY_F13

```
#define VZ_KEY_F13 302
```

10.99.1.33 VZ_KEY_F14

```
#define VZ_KEY_F14 303
```

10.99.1.34 VZ_KEY_F15

```
#define VZ_KEY_F15 304
```

10.99.1.35 VZ_KEY_F16

```
#define VZ_KEY_F16 305
```

10.99.1.36 VZ_KEY_F17

```
#define VZ_KEY_F17 306
```

10.99.1.37 VZ_KEY_F18

```
#define VZ_KEY_F18 307
```

10.99.1.38 VZ_KEY_F19

```
#define VZ_KEY_F19 308
```

10.99.1.39 VZ_KEY_F2

```
#define VZ_KEY_F2 291
```

10.99.1.40 VZ_KEY_F20

```
#define VZ_KEY_F20 309
```

10.99.1.41 VZ_KEY_F21

```
#define VZ_KEY_F21 310
```

10.99.1.42 VZ_KEY_F22

```
#define VZ_KEY_F22 311
```

10.99.1.43 VZ_KEY_F23

```
#define VZ_KEY_F23 312
```

10.99.1.44 VZ_KEY_F24

```
#define VZ_KEY_F24 313
```

10.99.1.45 VZ_KEY_F25

```
#define VZ_KEY_F25 314
```

10.99.1.46 VZ_KEY_F3

```
#define VZ_KEY_F3 292
```

10.99.1.47 VZ_KEY_F4

```
#define VZ_KEY_F4 293
```

10.99.1.48 VZ_KEY_F5

```
#define VZ_KEY_F5 294
```

10.99.1.49 VZ_KEY_F6

```
#define VZ_KEY_F6 295
```

10.99.1.50 VZ_KEY_F7

```
#define VZ_KEY_F7 296
```

10.99.1.51 VZ_KEY_F8

```
#define VZ_KEY_F8 297
```

10.99.1.52 VZ_KEY_F9

```
#define VZ_KEY_F9 298
```

10.99.1.53 VZ_KEY_G

```
#define VZ_KEY_G 71
```

10.99.1.54 VZ_KEY_GRAVE_ACCENT

```
#define VZ_KEY_GRAVE_ACCENT 96 /* ` */
```

10.99.1.55 VZ_KEY_H

```
#define VZ_KEY_H 72
```

10.99.1.56 VZ_KEY_HOME

```
#define VZ_KEY_HOME 268
```

10.99.1.57 VZ_KEY_I

```
#define VZ_KEY_I 73
```

10.99.1.58 VZ_KEY_INSERT

```
#define VZ_KEY_INSERT 260
```

10.99.1.59 VZ_KEY_J

```
#define VZ_KEY_J 74
```

10.99.1.60 VZ_KEY_K

```
#define VZ_KEY_K 75
```

10.99.1.61 VZ_KEY_KP_0

```
#define VZ_KEY_KP_0 320
```

10.99.1.62 VZ_KEY_KP_1

```
#define VZ_KEY_KP_1 321
```

10.99.1.63 VZ_KEY_KP_2

```
#define VZ_KEY_KP_2 322
```

10.99.1.64 VZ_KEY_KP_3

```
#define VZ_KEY_KP_3 323
```

10.99.1.65 VZ_KEY_KP_4

```
#define VZ_KEY_KP_4 324
```

10.99.1.66 VZ_KEY_KP_5

```
#define VZ_KEY_KP_5 325
```

10.99.1.67 VZ_KEY_KP_6

```
#define VZ_KEY_KP_6 326
```

10.99.1.68 VZ_KEY_KP_7

```
#define VZ_KEY_KP_7 327
```

10.99.1.69 VZ_KEY_KP_8

```
#define VZ_KEY_KP_8 328
```

10.99.1.70 VZ_KEY_KP_9

```
#define VZ_KEY_KP_9 329
```

10.99.1.71 VZ_KEY_KP_ADD

```
#define VZ_KEY_KP_ADD 334
```

10.99.1.72 VZ_KEY_KP_DECIMAL

```
#define VZ_KEY_KP_DECIMAL 330
```

10.99.1.73 VZ_KEY_KP_DIVIDE

```
#define VZ_KEY_KP_DIVIDE 331
```

10.99.1.74 VZ_KEY_KP_ENTER

```
#define VZ_KEY_KP_ENTER 335
```

10.99.1.75 VZ_KEY_KP_EQUAL

```
#define VZ_KEY_KP_EQUAL 336
```

10.99.1.76 VZ_KEY_KP_MULTIPLY

```
#define VZ_KEY_KP_MULTIPLY 332
```

10.99.1.77 VZ_KEY_KP_SUBTRACT

```
#define VZ_KEY_KP_SUBTRACT 333
```

10.99.1.78 VZ_KEY_L

```
#define VZ_KEY_L 76
```

10.99.1.79 VZ_KEY_LEFT

```
#define VZ_KEY_LEFT 263
```

10.99.1.80 VZ_KEY_LEFT_ALT

```
#define VZ_KEY_LEFT_ALT 342
```

10.99.1.81 VZ_KEY_LEFT_BRACKET

```
#define VZ_KEY_LEFT_BRACKET 91 /* [ */
```

10.99.1.82 VZ_KEY_LEFT_CONTROL

```
#define VZ_KEY_LEFT_CONTROL 341
```

10.99.1.83 VZ_KEY_LEFT_SHIFT

```
#define VZ_KEY_LEFT_SHIFT 340
```

10.99.1.84 VZ_KEY_LEFT_SUPER

```
#define VZ_KEY_LEFT_SUPER 343
```

10.99.1.85 VZ_KEY_M

```
#define VZ_KEY_M 77
```

10.99.1.86 VZ_KEY_MENU

```
#define VZ_KEY_MENU 348
```

10.99.1.87 VZ_KEY_MINUS

```
#define VZ_KEY_MINUS 45 /* - */
```

10.99.1.88 VZ_KEY_N

```
#define VZ_KEY_N 78
```

10.99.1.89 VZ_KEY_NUM_LOCK

```
#define VZ_KEY_NUM_LOCK 282
```

10.99.1.90 VZ_KEY_O

```
#define VZ_KEY_O 79
```

10.99.1.91 VZ_KEY_P

```
#define VZ_KEY_P 80
```

10.99.1.92 VZ_KEY_PAGE_DOWN

```
#define VZ_KEY_PAGE_DOWN 267
```

10.99.1.93 VZ_KEY_PAGE_UP

```
#define VZ_KEY_PAGE_UP 266
```

10.99.1.94 VZ_KEY_PAUSE

```
#define VZ_KEY_PAUSE 284
```

10.99.1.95 VZ_KEY_PERIOD

```
#define VZ_KEY_PERIOD 46 /* .  */
```

10.99.1.96 VZ_KEY_PRINT_SCREEN

```
#define VZ_KEY_PRINT_SCREEN 283
```

10.99.1.97 VZ_KEY_Q

```
#define VZ_KEY_Q 81
```

10.99.1.98 VZ_KEY_R

```
#define VZ_KEY_R 82
```

10.99.1.99 VZ_KEY_RIGHT

```
#define VZ_KEY_RIGHT 262
```

10.99.1.100 VZ_KEY_RIGHT_ALT

```
#define VZ_KEY_RIGHT_ALT 346
```

10.99.1.101 VZ_KEY_RIGHT_BRACKET

```
#define VZ_KEY_RIGHT_BRACKET 93 /* ] */
```

10.99.1.102 VZ_KEY_RIGHT_CONTROL

```
#define VZ_KEY_RIGHT_CONTROL 345
```

10.99.1.103 VZ_KEY_RIGHT_SHIFT

```
#define VZ_KEY_RIGHT_SHIFT 344
```

10.99.1.104 VZ_KEY_RIGHT_SUPER

```
#define VZ_KEY_RIGHT_SUPER 347
```

10.99.1.105 VZ_KEY_S

```
#define VZ_KEY_S 83
```

10.99.1.106 VZ_KEY_SCROLL_LOCK

```
#define VZ_KEY_SCROLL_LOCK 281
```

10.99.1.107 VZ_KEY_SEMICOLON

```
#define VZ_KEY_SEMICOLON 59 /* ; */
```

10.99.1.108 VZ_KEY_SLASH

```
#define VZ_KEY_SLASH 47 /* / */
```

10.99.1.109 VZ_KEY_SPACE

```
#define VZ_KEY_SPACE 32
```

10.99.1.110 VZ_KEY_T

```
#define VZ_KEY_T 84
```

10.99.1.111 VZ_KEY_TAB

```
#define VZ_KEY_TAB 258
```

10.99.1.112 VZ_KEY_U

```
#define VZ_KEY_U 85
```

10.99.1.113 VZ_KEY_UP

```
#define VZ_KEY_UP 265
```

10.99.1.114 VZ_KEY_V

```
#define VZ_KEY_V 86
```

10.99.1.115 VZ_KEY_W

```
#define VZ_KEY_W 87
```

10.99.1.116 VZ_KEY_WORLD_1

```
#define VZ_KEY_WORLD_1 161 /* non-US #1 */
```

10.99.1.117 VZ_KEY_WORLD_2

```
#define VZ_KEY_WORLD_2 162 /* non-US #2 */
```

10.99.1.118 VZ_KEY_X

```
#define VZ_KEY_X 88
```

10.99.1.119 VZ_KEY_Y

```
#define VZ_KEY_Y 89
```

10.99.1.120 VZ_KEY_Z

```
#define VZ_KEY_Z 90
```

10.100 KeyCodes.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003
00004 // From glfw3.h
00005 #define VZ_KEY_SPACE              32
00006 #define VZ_KEY_APOSTROPHE         39  /* ' */
00007 #define VZ_KEY_COMMA              44  /* , */
00008 #define VZ_KEY_MINUS              45  /* - */
00009 #define VZ_KEY_PERIOD             46  /* . */
00010 #define VZ_KEY_SLASH              47  /* / */
00011 #define VZ_KEY_0                  48
00012 #define VZ_KEY_1                  49
00013 #define VZ_KEY_2                  50
00014 #define VZ_KEY_3                  51
00015 #define VZ_KEY_4                  52
00016 #define VZ_KEY_5                  53
00017 #define VZ_KEY_6                  54
00018 #define VZ_KEY_7                  55
00019 #define VZ_KEY_8                  56
00020 #define VZ_KEY_9                  57
00021 #define VZ_KEY_SEMICOLON          59  /* ; */
00022 #define VZ_KEY_EQUAL              61  /* = */
00023 #define VZ_KEY_A                  65
00024 #define VZ_KEY_B                  66
00025 #define VZ_KEY_C                  67
00026 #define VZ_KEY_D                  68
00027 #define VZ_KEY_E                  69
00028 #define VZ_KEY_F                  70
00029 #define VZ_KEY_G                  71
00030 #define VZ_KEY_H                  72
00031 #define VZ_KEY_I                  73
00032 #define VZ_KEY_J                  74
00033 #define VZ_KEY_K                  75
00034 #define VZ_KEY_L                  76
00035 #define VZ_KEY_M                  77
00036 #define VZ_KEY_N                  78
00037 #define VZ_KEY_O                  79
```

```
00038 #define VZ_KEY_P                80
00039 #define VZ_KEY_Q                81
00040 #define VZ_KEY_R                82
00041 #define VZ_KEY_S                83
00042 #define VZ_KEY_T                84
00043 #define VZ_KEY_U                85
00044 #define VZ_KEY_V                86
00045 #define VZ_KEY_W                87
00046 #define VZ_KEY_X                88
00047 #define VZ_KEY_Y                89
00048 #define VZ_KEY_Z                90
00049 #define VZ_KEY_LEFT_BRACKET     91  /* [ */
00050 #define VZ_KEY_BACKSLASH        92  /* \ */
00051 #define VZ_KEY_RIGHT_BRACKET    93  /* ] */
00052 #define VZ_KEY_GRAVE_ACCENT     96  /* ` */
00053 #define VZ_KEY_WORLD_1          161 /* non-US #1 */
00054 #define VZ_KEY_WORLD_2          162 /* non-US #2 */
00055
00056 /* Function keys */
00057 #define VZ_KEY_ESCAPE           256
00058 #define VZ_KEY_ENTER            257
00059 #define VZ_KEY_TAB              258
00060 #define VZ_KEY_BACKSPACE        259
00061 #define VZ_KEY_INSERT           260
00062 #define VZ_KEY_DELETE           261
00063 #define VZ_KEY_RIGHT            262
00064 #define VZ_KEY_LEFT             263
00065 #define VZ_KEY_DOWN             264
00066 #define VZ_KEY_UP               265
00067 #define VZ_KEY_PAGE_UP          266
00068 #define VZ_KEY_PAGE_DOWN        267
00069 #define VZ_KEY_HOME             268
00070 #define VZ_KEY_END              269
00071 #define VZ_KEY_CAPS_LOCK        280
00072 #define VZ_KEY_SCROLL_LOCK      281
00073 #define VZ_KEY_NUM_LOCK         282
00074 #define VZ_KEY_PRINT_SCREEN     283
00075 #define VZ_KEY_PAUSE            284
00076 #define VZ_KEY_F1               290
00077 #define VZ_KEY_F2               291
00078 #define VZ_KEY_F3               292
00079 #define VZ_KEY_F4               293
00080 #define VZ_KEY_F5               294
00081 #define VZ_KEY_F6               295
00082 #define VZ_KEY_F7               296
00083 #define VZ_KEY_F8               297
00084 #define VZ_KEY_F9               298
00085 #define VZ_KEY_F10              299
00086 #define VZ_KEY_F11              300
00087 #define VZ_KEY_F12              301
00088 #define VZ_KEY_F13              302
00089 #define VZ_KEY_F14              303
00090 #define VZ_KEY_F15              304
00091 #define VZ_KEY_F16              305
00092 #define VZ_KEY_F17              306
00093 #define VZ_KEY_F18              307
00094 #define VZ_KEY_F19              308
00095 #define VZ_KEY_F20              309
00096 #define VZ_KEY_F21              310
00097 #define VZ_KEY_F22              311
00098 #define VZ_KEY_F23              312
00099 #define VZ_KEY_F24              313
00100 #define VZ_KEY_F25              314
00101 #define VZ_KEY_KP_0             320
00102 #define VZ_KEY_KP_1             321
00103 #define VZ_KEY_KP_2             322
00104 #define VZ_KEY_KP_3             323
00105 #define VZ_KEY_KP_4             324
00106 #define VZ_KEY_KP_5             325
00107 #define VZ_KEY_KP_6             326
00108 #define VZ_KEY_KP_7             327
00109 #define VZ_KEY_KP_8             328
00110 #define VZ_KEY_KP_9             329
00111 #define VZ_KEY_KP_DECIMAL       330
00112 #define VZ_KEY_KP_DIVIDE        331
00113 #define VZ_KEY_KP_MULTIPLY      332
00114 #define VZ_KEY_KP_SUBTRACT      333
00115 #define VZ_KEY_KP_ADD           334
00116 #define VZ_KEY_KP_ENTER         335
00117 #define VZ_KEY_KP_EQUAL         336
00118 #define VZ_KEY_LEFT_SHIFT       340
00119 #define VZ_KEY_LEFT_CONTROL     341
00120 #define VZ_KEY_LEFT_ALT         342
00121 #define VZ_KEY_LEFT_SUPER       343
00122 #define VZ_KEY_RIGHT_SHIFT      344
00123 #define VZ_KEY_RIGHT_CONTROL    345
00124 #define VZ_KEY_RIGHT_ALT        346
```

```
00125 #define VZ_KEY_RIGHT_SUPER        347
00126 #define VZ_KEY_MENU               348
```

10.101 Vesper/src/Vesper/Input/MouseButtonCodes.h File Reference

Macros

- #define VZ_MOUSE_BUTTON_1 0
- #define VZ_MOUSE_BUTTON_2 1
- #define VZ_MOUSE_BUTTON_3 2
- #define VZ_MOUSE_BUTTON_4 3
- #define VZ_MOUSE_BUTTON_5 4
- #define VZ_MOUSE_BUTTON_6 5
- #define VZ_MOUSE_BUTTON_7 6
- #define VZ_MOUSE_BUTTON_8 7
- #define VZ_MOUSE_BUTTON_LAST VZ_MOUSE_BUTTON_8
- #define VZ_MOUSE_BUTTON_LEFT VZ_MOUSE_BUTTON_1
- #define VZ_MOUSE_BUTTON_RIGHT VZ_MOUSE_BUTTON_2
- #define VZ_MOUSE_BUTTON_MIDDLE VZ_MOUSE_BUTTON_3

10.101.1 Macro Definition Documentation

10.101.1.1 VZ_MOUSE_BUTTON_1

```
#define VZ_MOUSE_BUTTON_1 0
```

10.101.1.2 VZ_MOUSE_BUTTON_2

```
#define VZ_MOUSE_BUTTON_2 1
```

10.101.1.3 VZ_MOUSE_BUTTON_3

```
#define VZ_MOUSE_BUTTON_3 2
```

10.101.1.4 VZ_MOUSE_BUTTON_4

```
#define VZ_MOUSE_BUTTON_4 3
```

10.101.1.5 VZ_MOUSE_BUTTON_5

```
#define VZ_MOUSE_BUTTON_5 4
```

10.101.1.6 VZ_MOUSE_BUTTON_6

```
#define VZ_MOUSE_BUTTON_6 5
```

10.101.1.7 VZ_MOUSE_BUTTON_7

```
#define VZ_MOUSE_BUTTON_7 6
```

10.101.1.8 VZ_MOUSE_BUTTON_8

```
#define VZ_MOUSE_BUTTON_8 7
```

10.101.1.9 VZ_MOUSE_BUTTON_LAST

```
#define VZ_MOUSE_BUTTON_LAST VZ_MOUSE_BUTTON_8
```

10.101.1.10 VZ_MOUSE_BUTTON_LEFT

```
#define VZ_MOUSE_BUTTON_LEFT VZ_MOUSE_BUTTON_1
```

10.101.1.11 VZ_MOUSE_BUTTON_MIDDLE

```
#define VZ_MOUSE_BUTTON_MIDDLE VZ_MOUSE_BUTTON_3
```

10.101.1.12 VZ_MOUSE_BUTTON_RIGHT

```
#define VZ_MOUSE_BUTTON_RIGHT VZ_MOUSE_BUTTON_2
```

10.102 MouseButtonCodes.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 // From glfw3.h
00004 #define VZ_MOUSE_BUTTON_1         0
00005 #define VZ_MOUSE_BUTTON_2         1
00006 #define VZ_MOUSE_BUTTON_3         2
00007 #define VZ_MOUSE_BUTTON_4         3
00008 #define VZ_MOUSE_BUTTON_5         4
00009 #define VZ_MOUSE_BUTTON_6         5
00010 #define VZ_MOUSE_BUTTON_7         6
00011 #define VZ_MOUSE_BUTTON_8         7
00012 #define VZ_MOUSE_BUTTON_LAST      VZ_MOUSE_BUTTON_8
00013 #define VZ_MOUSE_BUTTON_LEFT      VZ_MOUSE_BUTTON_1
00014 #define VZ_MOUSE_BUTTON_RIGHT     VZ_MOUSE_BUTTON_2
00015 #define VZ_MOUSE_BUTTON_MIDDLE    VZ_MOUSE_BUTTON_3
```

10.103 Vesper/src/Vesper/ParticleSystem/ParticleSystem.cpp File Reference

```
#include "vzpch.h"
#include "ParticleSystem.h"
#include "Vesper/Renderer/Renderer2D.h"
#include "Vesper/Renderer/OrthographicCamera.h"
#include <glm/gtc/constants.hpp>
#include <glm/gtx/compatibility.hpp>
```

Namespaces

- namespace Vesper

  TEMPORARY.

Macros

- #define GLM_ENABLE_EXPERIMENTAL

10.103.1 Macro Definition Documentation

10.103.1.1 GLM_ENABLE_EXPERIMENTAL

```
#define GLM_ENABLE_EXPERIMENTAL
```

10.104 Vesper/src/Vesper/ParticleSystem/ParticleSystem.h File Reference

```
#include "Vesper.h"
#include "Vesper/Renderer/OrthographicCamera.h"
```

Classes

- struct Vesper::ParticleProps
- class Vesper::ParticleSystem
- struct Vesper::ParticleSystem::Particle

Namespaces

- namespace Vesper

  TEMPORARY.

10.104.1 Class Documentation

10.104.1.1 struct Vesper::ParticleProps

Class Members

vec4	ColorBegin = { 1.0f, 1.0f, 1.0f, 1.0f }	
vec4	ColorEnd = { 1.0f, 1.0f, 1.0f, 1.0f }	
float	LifeTime = 1.0f	
float	LifetimeVariation = 0.0f	
vec3	Position = { 0.0f, 0.0f, 0.0f }	
float	Rotation = 0.0f	
float	RotationVariation = 0.0f	
float	SizeBegin = 1.0f	
float	SizeEnd = 0.0f	
float	SizeVariation = 0.0f	
vec3	Velocity = { 0.0f, 0.0f, 0.0f }	
vec3	VelocityVariation = { 0.0f, 0.0f, 0.0f }	

10.104.1.2 struct Vesper::ParticleSystem::Particle

Class Members

bool	Active = false	
vec4	ColorBegin	
vec4	ColorEnd	
float	LifeRemaining = 0.0f	
float	LifeTime = 0.0f	
vec3	Position	
float	Rotation	
float	SizeBegin	
float	SizeEnd	
vec3	Velocity	

10.105 ParticleSystem.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper.h"
00004 #include "Vesper/Renderer/OrthographicCamera.h"
00005
00006 namespace Vesper {
00007
00008     struct ParticleProps
00009     {
00010         glm::vec3 Position = { 0.0f, 0.0f, 0.0f };
00011         glm::vec3 Velocity = { 0.0f, 0.0f, 0.0f };
```

```
00012         glm::vec3 VelocityVariation = { 0.0f, 0.0f, 0.0f };
00013         glm::vec4 ColorBegin = { 1.0f, 1.0f, 1.0f, 1.0f };
00014         glm::vec4 ColorEnd = { 1.0f, 1.0f, 1.0f, 1.0f };
00015         float SizeBegin = 1.0f;
00016         float SizeEnd = 0.0f;
00017         float SizeVariation = 0.0f;
00018         float Rotation = 0.0f;
00019         float RotationVariation = 0.0f;
00020         float LifeTime = 1.0f;
00021         float LifetimeVariation = 0.0f;
00022     };
00023
00024     class ParticleSystem
00025     {
00026     public:
00027         ParticleSystem();
00028         ParticleSystem(uint32_t maxParticles);
00029
00030         void OnUpdate(Timestep ts);
00031         void OnRender(OrthographicCamera& camera);
00032         void Emit(const ParticleProps& particleProps);
00033         void SetParticleProps(const ParticleProps& particleProps) { m_Props = particleProps; }
00034
00035     private:
00036         struct Particle
00037         {
00038             glm::vec3 Position;
00039             glm::vec3 Velocity;
00040             glm::vec4 ColorBegin, ColorEnd;
00041             float SizeBegin, SizeEnd;
00042             float Rotation;
00043             float LifeTime = 0.0f;
00044             float LifeRemaining = 0.0f;
00045             bool Active = false;
00046         };
00047         std::vector<Particle> m_ParticlePool;
00048         uint32_t m_PoolIndex = 999;
00049         ParticleProps m_Props;
00050     };
00051
00052
00053 }
```

10.106 Vesper/src/Vesper/Renderer/Buffer.cpp File Reference

```
#include "vzpch.h"
#include "Buffer.h"
#include "Renderer.h"
#include "RenderAPI/OpenGL/OpenGLBuffer.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.107 Vesper/src/Vesper/Renderer/Buffer.h File Reference

Classes

- struct Vesper::BufferElement
- class Vesper::BufferLayout
- class Vesper::VertexBuffer
- class Vesper::IndexBuffer

Namespaces

- namespace Vesper

  TEMPORARY.

Enumerations

- enum class Vesper::ShaderDataType {
  Vesper::None = 0 , Vesper::Float , Vesper::Float2 , Vesper::Float3 ,
  Vesper::Float4 , Vesper::Mat3 , Vesper::Mat4 , Vesper::Int ,
  Vesper::Int2 , Vesper::Int3 , Vesper::Int4 , Vesper::Bool }

Functions

- static uint32_t Vesper::ShaderDataTypeSize (ShaderDataType type)

10.108 Buffer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 namespace Vesper {
00004
00005     enum class ShaderDataType {
00006         None = 0,
00007         Float, Float2, Float3, Float4,
00008         Mat3, Mat4,
00009         Int, Int2, Int3, Int4,
00010         Bool
00011     };
00012
00013     static uint32_t ShaderDataTypeSize(ShaderDataType type) {
00014         switch (type) {
00015         case ShaderDataType::Float:     return 4;
00016         case ShaderDataType::Float2:    return 4 * 2;
00017         case ShaderDataType::Float3:    return 4 * 3;
00018         case ShaderDataType::Float4:    return 4 * 4;
00019         case ShaderDataType::Mat3:      return 4 * 3 * 3;
00020         case ShaderDataType::Mat4:      return 4 * 4 * 4;
00021         case ShaderDataType::Int:       return 4;
00022         case ShaderDataType::Int2:      return 4 * 2;
00023         case ShaderDataType::Int3:      return 4 * 3;
00024         case ShaderDataType::Int4:      return 4 * 4;
00025         case ShaderDataType::Bool:      return 1;
00026         }
00027         VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00028         return 0;
00029     }
00030
00031     struct BufferElement {
00032
00033         std::string Name;
00034         ShaderDataType Type;
00035         uint32_t Size;
00036         uint32_t Offset;
00037         bool Normalized;
00038
00039         BufferElement() {}
00040         BufferElement(ShaderDataType type, const std::string& name, bool normalized = false)
00041             : Name(name), Type(type), Size(ShaderDataTypeSize(type)), Offset(0),
    Normalized(normalized)
00042         {
00043         }
00044
00045         uint32_t GetComponentCount() const {
00046             switch (Type) {
00047             case ShaderDataType::Float:     return 1;
00048             case ShaderDataType::Float2:    return 2;
00049             case ShaderDataType::Float3:    return 3;
00050             case ShaderDataType::Float4:    return 4;
```

```
00051             case ShaderDataType::Mat3:      return 3 * 3;
00052             case ShaderDataType::Mat4:      return 4 * 4;
00053             case ShaderDataType::Int:       return 1;
00054             case ShaderDataType::Int2:      return 2;
00055             case ShaderDataType::Int3:      return 3;
00056             case ShaderDataType::Int4:      return 4;
00057             case ShaderDataType::Bool:      return 1;
00058             }
00059             VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00060             return 0;
00061         }
00062     };
00063
00064
00065     class BufferLayout {
00066     public:
00067         BufferLayout(){}
00068         BufferLayout(const std::initializer_list<BufferElement>& elements)
00069             : m_Elements(elements), m_Stride(0)
00070         {
00071             CalculateOffsetsAndStride();
00072         }
00073
00074
00075         inline const std::vector<BufferElement>& GetElements() const { return m_Elements; }
00076         inline uint32_t GetStride() const { return m_Stride; }
00077
00078         std::vector<BufferElement>::iterator begin() { return m_Elements.begin(); }
00079         std::vector<BufferElement>::const_iterator begin() const { return m_Elements.begin(); }
00080         std::vector<BufferElement>::iterator end() { return m_Elements.end(); }
00081         std::vector<BufferElement>::const_iterator end() const { return m_Elements.end(); }
00082
00083
00084
00085     private:
00086         void CalculateOffsetsAndStride() {
00087             uint32_t offset = 0;
00088             m_Stride = 0;
00089             for (auto& element : m_Elements) {
00090                 element.Offset = offset;
00091                 offset += element.Size;
00092                 m_Stride += element.Size;
00093             }
00094         }
00095
00096     private:
00097         std::vector<BufferElement> m_Elements;
00098         uint32_t m_Stride = 0;
00099     };
00100
00101
00102     class VertexBuffer
00103     {
00104     public:
00105         virtual ~VertexBuffer() {}
00106
00107         virtual void Bind() const = 0;
00108         virtual void Unbind() const = 0;
00109
00110         virtual void SetLayout(const BufferLayout& layout) = 0;
00111         virtual const BufferLayout& GetLayout() const = 0;
00112
00113         virtual void SetData(const void* data, uint32_t size) = 0;
00114
00115
00116         static Ref<VertexBuffer> Create(uint32_t size);
00117         static Ref<VertexBuffer> Create(float* vertices, uint32_t size);
00118     };
00119
00120     // Currently only supports uint32_t indices
00121     class IndexBuffer
00122     {
00123     public:
00124         virtual ~IndexBuffer() {}
00125         virtual void Bind() const = 0;
00126         virtual void Unbind() const = 0;
00127
00128         virtual uint32_t GetCount() const = 0;
00129
00130         static Ref<IndexBuffer> Create(uint32_t* indices, uint32_t count);
00131     };
00132 }
```

10.109 Vesper/src/Vesper/Renderer/Camera.h File Reference

```
#include <glm/glm.hpp>
```

Classes

- class Vesper::Camera

Namespaces

- namespace Vesper
  TEMPORARY.

10.110 Camera.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <glm/glm.hpp>
00003
00004 namespace Vesper {
00005
00006     class Camera
00007     {
00008     public:
00009         Camera() = default;
00010         Camera(const glm::mat4& projection)
00011             : m_Projection(projection) {
00012         }
00013         ~Camera() = default;
00014
00015         const glm::mat4& GetProjection() const { return m_Projection; }
00016     protected:
00017         glm::mat4 m_Projection = glm::mat4(1.0f);
00018
00019     };
00020
00021 }
```

10.111 Vesper/src/Vesper/Renderer/EditorCamera.cpp File Reference

```
#include "vzpch.h"
#include "EditorCamera.h"
#include <glfw/glfw3.h>
#include <Vesper.h>
#include <glm/gtx/quaternion.hpp>
```

Namespaces

- namespace Vesper
  TEMPORARY.

Macros

- #define GLM_ENABLE_EXPERIMENTAL

10.111.1 Macro Definition Documentation

10.111.1.1 GLM_ENABLE_EXPERIMENTAL

```
#define GLM_ENABLE_EXPERIMENTAL
```

10.112 Vesper/src/Vesper/Renderer/EditorCamera.h File Reference

```
#include "Camera.h"
#include "Vesper/Core/Timestep.h"
#include "Vesper/Events/Event.h"
#include "Vesper/Events/MouseEvent.h"
#include <glm/glm.hpp>
```

Classes

- class Vesper::EditorCamera

Namespaces

- namespace Vesper

  TEMPORARY.

10.113 EditorCamera.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Camera.h"
00004 #include "Vesper/Core/Timestep.h"
00005 #include "Vesper/Events/Event.h"
00006 #include "Vesper/Events/MouseEvent.h"
00007
00008 #include <glm/glm.hpp>
00009
00010
00011 namespace Vesper {
00012     class EditorCamera : public Camera
00013     {
00014     public:
00015         EditorCamera();
00016         EditorCamera(float fov, float aspectRation, float nearClip, float farClip);
00017
00018         void OnUpdate(Timestep ts);
00019         void OnEvent(Event& e);
00020         inline float GetDistance() const { return m_Distance; }
00021         inline void SetDistance(float distance) { m_Distance = distance; }
00022
00023         inline void SetViewportSize(float width, float height) { m_ViewportWidth = width,
      m_ViewportHeight = height; UpdateProjection(); }
```

```
00024
00025         const glm::mat4& GetViewMatrix() const { return m_ViewMatrix; }
00026         const glm::mat4& GetViewProjection() const { return m_Projection * m_ViewMatrix; }
00027
00028         glm::vec3 GetUpDirection() const;
00029         glm::vec3 GetRightDirection() const;
00030         glm::vec3 GetForwardDirection() const;
00031         glm::quat GetOrientation() const;
00032
00033         const glm::vec3& GetPosition() const { return m_Position; }
00034         void SetPosition(const glm::vec3& position);
00035
00036         float GetPitch() const { return m_Pitch; }
00037         void SetPitch(float pitch) { m_Pitch = pitch; }
00038
00039         float GetYaw() const { return m_Yaw; }
00040         void SetYaw(float yaw) { m_Yaw = yaw; }
00041     private:
00042         void UpdateProjection();
00043         void UpdateView();
00044
00045         bool OnMouseScroll(MouseScrolledEvent& e);
00046
00047         void MousePan(const glm::vec2& delta);
00048         void MouseRotate(const glm::vec2& delta);
00049         void MouseZoom(float delta);
00050
00051         glm::vec3 CalculatePosition() const;
00052
00053         std::pair<float, float> PanSpeed() const;
00054         float RotationSpeed() const;
00055         float ZoomSpeed() const;
00056
00057
00058     private:
00059         float m_FOV = 45.0f, m_AspectRatio = 1.778f, m_NearClip = 0.1f, m_FarClip = 1000.0f;
00060
00061         glm::mat4 m_ViewMatrix;
00062         glm::vec3 m_Position = { 0.0f, 0.0f, 0.0f };
00063         glm::vec3 m_FocalPoint = glm::vec3(1.0f);
00064
00065         glm::vec2 m_InitialMousePosition = { 0.0f, 0.0f };
00066
00067         float m_Distance = 10.0f;
00068         float m_Pitch = 0.0f, m_Yaw = 0.0f;
00069
00070         float m_ViewportWidth = 1280, m_ViewportHeight = 720;
00071
00072     };
00073 }
```

10.114 Vesper/src/Vesper/Renderer/Framebuffer.cpp File Reference

```
#include "vzpch.h"
#include "Framebuffer.h"
#include "Vesper/Renderer/Renderer.h"
#include "RenderAPI/OpenGL/OpenGLFramebuffer.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.115 Vesper/src/Vesper/Renderer/Framebuffer.h File Reference

```
#include "Vesper/Core/Base.h"
```

Classes

- struct Vesper::FramebufferSpecification
- class Vesper::Framebuffer

Namespaces

- namespace Vesper

  TEMPORARY.

10.115.1 Class Documentation

10.115.1.1 struct Vesper::FramebufferSpecification

Class Members

uint32_t	Height	
uint32_t	Samples = 1	
bool	SwapChainTarget = false	
uint32_t	Width	

10.116 Framebuffer.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Vesper/Core/Base.h"
00003
00004 namespace Vesper {
00005
00006
00007     struct FramebufferSpecification
00008     {
00009         uint32_t Width, Height;
00010         // FramebufferFormat Format = FramebufferFormat::RGBA8;
00011         uint32_t Samples = 1;
00012
00013         bool SwapChainTarget = false;
00014     };
00015
00016
00017     class Framebuffer
00018     {
00019     public:
00020         ~Framebuffer() = default;
00021         virtual void Bind() = 0;
00022         virtual void Unbind() = 0;
00023
00024         virtual void Resize(uint32_t width, uint32_t height) = 0;
00025
00026         virtual uint32_t GetColorAttachmentRendererID() const = 0;
00027
00028         virtual const FramebufferSpecification& GetSpecification() const = 0;
00029
00030         static Ref<Framebuffer> Create(const FramebufferSpecification& spec);
00031     };
00032
00033 }
```

10.117 Vesper/src/Vesper/Renderer/GraphicsContext.h File Reference

```
#include "Vesper/Core/Base.h"
```

Classes

- class Vesper::GraphicsContext

Namespaces

- namespace Vesper

  TEMPORARY.

10.118 GraphicsContext.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Core/Base.h"
00004
00005 namespace Vesper {
00006
00007     class VESPER_API GraphicsContext
00008     {
00009     public:
00010         virtual ~GraphicsContext() {}
00011         virtual void Init() = 0;
00012         virtual void SwapBuffers() = 0;
00013
00014     };
00015
00016 }
```

10.119 Vesper/src/Vesper/Renderer/OrthographicCamera.cpp File Reference

```
#include "vzpch.h"
#include "OrthographicCamera.h"
#include <glm/gtc/matrix_transform.hpp>
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.120 Vesper/src/Vesper/Renderer/OrthographicCamera.h File Reference

```
#include <glm/glm.hpp>
```

Classes

- class Vesper::OrthographicCamera

Namespaces

- namespace Vesper

  TEMPORARY.

10.121 OrthographicCamera.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <glm/glm.hpp>
00003
00004 namespace Vesper {
00005
00006     class OrthographicCamera
00007     {
00008     public:
00009         OrthographicCamera(float left, float right, float bottom, float top);
00010
00011         void SetProjection(float left, float right, float bottom, float top);
00012
00013         void SetPosition(const glm::vec3& position) { m_Position = position; RecalculateViewMatrix();
      }
00014         const glm::vec3& GetPosition() const { return m_Position; }
00015
00016         void SetRotation(float rotation) { m_Rotation = rotation; RecalculateViewMatrix(); }
00017         const float GetRotation() const { return m_Rotation; }
00018
00019
00020         const glm::mat4& GetProjectionMatrix() const { return m_ProjectionMatrix; }
00021         const glm::mat4& GetViewMatrix() const { return m_ViewMatrix; }
00022         const glm::mat4& GetViewProjectionMatrix() const { return m_ViewProjectionMatrix; }
00023     private:
00024         void RecalculateViewMatrix();
00025     private:
00026         glm::mat4 m_ProjectionMatrix;
00027         glm::mat4 m_ViewMatrix;
00028         glm::mat4 m_ViewProjectionMatrix;
00029
00030         glm::vec3 m_Position = { 0.0f, 0.0f, 0.0f };
00031         float m_Rotation = 0.0f;
00032     };
00033
00034 }
```

10.122 Vesper/src/Vesper/Renderer/OrthographicCameraController.cpp File Reference

```
#include "vzpch.h"
#include "OrthographicCameraController.h"
#include "Vesper/Input/Input.h"
#include "Vesper/Input/KeyCodes.h"
#include <imgui.h>
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.123 Vesper/src/Vesper/Renderer/OrthographicCameraController.h File Reference

```
#include "Vesper/Renderer/OrthographicCamera.h"
#include "Vesper/Core/Timestep.h"
#include "Vesper/Events/ApplicationEvent.h"
#include "Vesper/Events/MouseEvent.h"
```

Classes

- struct Vesper::OrthographicCameraBounds
- class Vesper::OrthographicCameraController

Namespaces

- namespace Vesper

  TEMPORARY.

10.124 OrthographicCameraController.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Renderer/OrthographicCamera.h"
00004 #include "Vesper/Core/Timestep.h"
00005
00006 #include "Vesper/Events/ApplicationEvent.h"
00007 #include "Vesper/Events/MouseEvent.h"
00008
00009
00010
00011 namespace Vesper {
00012
00013
00014     struct OrthographicCameraBounds
00015     {
00016         float Left, Right;
00017         float Bottom, Top;
00018
00019         float GetWidth() const { return Right - Left; }
00020         float GetHeight() const { return Top - Bottom; }
00021
00022     };
00023
00024
00025     class OrthographicCameraController
00026     {
00027     public:
00028         OrthographicCameraController(float aspectRatio, bool rotation = false);
00029
00030         void OnUpdate(Timestep ts);
00031         void OnEvent(Event& e);
00032         void OnResize(float width, float height);
00033
00034         OrthographicCamera& GetCamera() { return camera; }
00035         const OrthographicCamera& GetCamera() const { return camera; }
00036         OrthographicCameraBounds GetBounds() const { return m_Bounds; }
00037
00038         glm::vec3 GetPosition() const { return m_CameraPosition; }
00039         void SetPosition(float x, float y);
00040
00041         void SetMoveSpeed(float speed);
00042         float GetMoveSpeed() const { return m_CameraMoveSpeed; }
00043
00044         void SetRotation(float rotation);
```

```
00045         float GetRotation() const { return m_CameraRotation; }
00046
00047         void SetRotationSpeed(float speed);
00048         float GetRotationSpeed() const { return m_CameraRotationSpeed; }
00049
00050         float GetAspectRatio() const;
00051         void SetAspectRatio(float aspectRatio);
00052
00053         bool CanRotate() const { return m_Rotation; }
00054         void SetCanRotate(bool canRotate) { m_Rotation = canRotate; }
00055
00056         void SetZoomLevel(float level) { m_ZoomLevel = level; CalculateView(); }
00057         float GetZoomLevel() const { return m_ZoomLevel; }
00058
00059         void OnImGuiRender();
00060     private:
00061         bool OnMouseScrolled(MouseScrolledEvent& e);
00062         bool OnWindowResized(WindowResizeEvent& e);
00063         void UpdateCameraBounds();
00064         void OnUpdateBounds();
00065         void CalculateView();
00066
00067     private:
00068         float m_AspectRatio;
00069         float m_ZoomLevel = 1.0f;
00070         OrthographicCamera camera;
00071         OrthographicCameraBounds m_Bounds;
00072
00073         bool m_Rotation = true;
00074         glm::vec3 m_CameraPosition = { 0.0f, 0.0f, 0.0f };
00075         float m_CameraRotation = 0.0f;
00076         float m_CameraMoveSpeed = 5.0f, m_CameraRotationSpeed = 180.0f;
00077
00078     };
00079
00080 }
```

10.125 Vesper/src/Vesper/Renderer/RenderCommand.cpp File Reference

```
#include "vzpch.h"
#include "RenderCommand.h"
#include "RenderAPI/OpenGL/OpenGLRendererAPI.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.126 Vesper/src/Vesper/Renderer/RenderCommand.h File Reference

```
#include "RendererAPI.h"
```

Classes

- class Vesper::RenderCommand

Namespaces

- namespace Vesper

  TEMPORARY.

10.127 RenderCommand.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "RendererAPI.h"
00004
00005 namespace Vesper {
00006
00007     class RenderCommand
00008     {
00009     public:
00010
00011         inline static void Init()
00012         {
00013             s_RendererAPI->Init();
00014         }
00015
00016         inline static void SetViewport(uint32_t x, uint32_t y, uint32_t width, uint32_t height)
00017         {
00018             s_RendererAPI->SetViewport(x, y, width, height);
00019         }
00020
00021         inline static void SetClearColor(const glm::vec4& color)
00022         {
00023             s_RendererAPI->SetClearColor(color);
00024         }
00025
00026         inline static void Clear()
00027         {
00028             s_RendererAPI->Clear();
00029         }
00030
00031         inline static void DrawIndexed(const Ref<VertexArray>& vertexArray, uint32_t indexCount = 0)
00032         {
00033             s_RendererAPI->DrawIndexed(vertexArray, indexCount);
00034         }
00035
00036     private:
00037         static RendererAPI* s_RendererAPI;
00038     };
00039
00040 }
```

10.128 Vesper/src/Vesper/Renderer/Renderer.cpp File Reference

```
#include "vzpch.h"
#include "Renderer.h"
#include "RenderCommand.h"
#include "RenderAPI/OpenGL/OpenGLShader.h"
#include "Renderer2D.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.129 Vesper/src/Vesper/Renderer/Renderer.h File Reference

```
#include "Vesper/Renderer/RenderCommand.h"
#include "Vesper/Renderer/OrthographicCamera.h"
#include "Vesper/Renderer/Shader.h"
```

Classes

- class Vesper::Renderer
- struct Vesper::Renderer::SceneData

Namespaces

- namespace Vesper

  TEMPORARY.

10.129.1 Class Documentation

10.129.1.1 struct Vesper::Renderer::SceneData

Class Members

mat4	ViewProjectionMatrix	

10.130 Renderer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Renderer/RenderCommand.h"
00004 #include "Vesper/Renderer/OrthographicCamera.h"
00005 #include "Vesper/Renderer/Shader.h"
00006
00007 namespace Vesper {
00008
00009     class Renderer {
00010
00011     public:
00012
00013         static void Init();
00014         static void OnWindowResize(uint32_t width, uint32_t height);
00015
00016
00017         static void BeginScene(OrthographicCamera& camera);
00018         static void EndScene();
00019
00020         static void Submit(const Ref<Shader>& shader, const Ref<VertexArray>& vertexArray, const
      glm::mat4& transform = glm::mat4(1.0f));
00021
00022
00023         inline static RendererAPI::API GetAPI() { return RendererAPI::GetAPI(); }
00024
00025     private:
00026         struct SceneData {
00027             glm::mat4 ViewProjectionMatrix;
00028         };
00029
00030         static SceneData* s_SceneData;
00031
00032     };
00033 }
```

10.131 Vesper/src/Vesper/Renderer/Renderer2D.cpp File Reference

```
#include "vzpch.h"
#include "Renderer2D.h"
#include "UniformBuffer.h"
#include "VertexArray.h"
#include "Shader.h"
#include "RenderCommand.h"
#include <glm/gtc/matrix_transform.hpp>
```

Classes

- struct Vesper::QuadVertex
- struct Vesper::Renderer2DData
- struct Vesper::Renderer2DData::CameraData

Namespaces

- namespace Vesper

  TEMPORARY.

Variables

- static Renderer2DData Vesper::s_Data

10.131.1 Class Documentation

10.131.1.1 struct Vesper::QuadVertex

Class Members

vec4	Color	
vec3	Position	
vec2	TexCoord	
float	TexIndex	
float	TilingFactor	

10.131.1.2 struct Vesper::Renderer2DData::CameraData

Class Members

mat4	ViewProjection	

10.132 Vesper/src/Vesper/Renderer/Renderer2D.h File Reference

```
#include "Vesper/Renderer/OrthographicCamera.h"
#include "Vesper/Renderer/Texture.h"
#include "Vesper/Renderer/SubTexture2D.h"
#include "Vesper/Renderer/Camera.h"
#include "Vesper/Renderer/EditorCamera.h"
#include "Vesper/Scene/Components.h"
```

Classes

- class Vesper::Renderer2D
- struct Vesper::Renderer2D::Statistics

Namespaces

- namespace Vesper

  TEMPORARY.

10.133 Renderer2D.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Renderer/OrthographicCamera.h"
00004
00005 #include "Vesper/Renderer/Texture.h"
00006 #include "Vesper/Renderer/SubTexture2D.h"
00007
00008 #include "Vesper/Renderer/Camera.h"
00009 #include "Vesper/Renderer/EditorCamera.h"
00010 #include "Vesper/Scene/Components.h"
00011
00012 namespace Vesper {
00013
00014
00015     class Renderer2D
00016     {
00017     public:
00018         static void Init();
00019         static void Shutdown();
00020
00021         static void BeginScene(const Camera& camera, const glm::mat4& transform);
00022         static void BeginScene(const EditorCamera& camera);
00023         static void BeginScene(const OrthographicCamera& camera); // TODO: Remove once we have a
      proper scene system
00024         static void EndScene();
00025         static void Flush();
00026
00027         // Primitives
00028         static void DrawQuad(const glm::mat4& transform, const glm::vec4& color);
00029         static void DrawQuad(const glm::vec2& position, const glm::vec2& size, const glm::vec4&
      color);
00030         static void DrawQuad(const glm::vec3& position, const glm::vec2& size, const glm::vec4&
      color);
00031
00032         static void DrawQuadWithTexture(const glm::mat4& transform, const Ref<Texture2D>& texture,
      float tilingFactor, const glm::vec4 tintColor);
00033         static void DrawQuadWithTexture(const glm::vec2& position, const glm::vec2& size, const
      Ref<Texture2D>& texture, float tilingFactor, const glm::vec4 tintColor);
00034         static void DrawQuadWithTexture(const glm::vec3& position, const glm::vec2& size, const
      Ref<Texture2D>& texture, float tilingFactor, const glm::vec4 tintColor);
00035
00036         static void DrawQuadWithTexture(const glm::mat4& transform, const Ref<SubTexture2D>&
      subtexture, float tilingFactor, const glm::vec4 tintColor);
00037         static void DrawQuadWithTexture(const glm::vec2& position, const glm::vec2& size, const
      Ref<SubTexture2D>& subtexture, float tilingFactor, const glm::vec4 tintColor);
00038         static void DrawQuadWithTexture(const glm::vec3& position, const glm::vec2& size, const
      Ref<SubTexture2D>& subtexture, float tilingFactor, const glm::vec4 tintColor);
00039
00040         static void DrawQuadRotated(const glm::mat4& transform, const glm::vec4& color);
00041         static void DrawQuadRotated(const glm::vec2& position, const glm::vec2& size, float
      rotationRads, const glm::vec4& color);
00042         static void DrawQuadRotated(const glm::vec3& position, const glm::vec2& size, float
      rotationRads, const glm::vec4& color);
00043
00044         static void DrawQuadRotatedWithTexture(const glm::mat4& transform, const Ref<Texture2D>&
      texture, float tilingFactor, const glm::vec4 tintColor);
00045         static void DrawQuadRotatedWithTexture(const glm::vec2& position, const glm::vec2& size, const
      Ref<Texture2D>& texture, float rotationRads, float tilingFactor, const glm::vec4 tintColor);
00046         static void DrawQuadRotatedWithTexture(const glm::vec3& position, const glm::vec2& size, const
      Ref<Texture2D>& texture, float rotationRads, float tilingFactor, const glm::vec4 tintColor);
00047
```

```
00048         static void DrawQuadRotatedWithTexture(const glm::mat4& transform, const Ref<SubTexture2D>&
      subtexture, float tilingFactor, const glm::vec4 tintColor);
00049         static void DrawQuadRotatedWithTexture(const glm::vec2& position, const glm::vec2& size, const
      Ref<SubTexture2D>& subtexture, float rotationRads, float tilingFactor, const glm::vec4 tintColor);
00050         static void DrawQuadRotatedWithTexture(const glm::vec3& position, const glm::vec2& size, const
      Ref<SubTexture2D>& subtexture, float rotationRads, float tilingFactor, const glm::vec4 tintColor);
00051
00052         //static void DrawSprite(const glm::mat4& transform, const SpriteRendererComponent& src, int
      entityID);
00053         //static void DrawSprite(const glm::mat4& transform, const SubTextureComponent& stc, int
      entityID);
00054
00055         static Ref<Texture2D> GetWhiteTexture();
00056
00057         struct Statistics {
00058             uint32_t DrawCalls = 0;
00059             uint32_t QuadCount = 0;
00060             uint32_t GetTotalVertexCount() { return QuadCount * 4; }
00061             uint32_t GetTotalIndexCount() { return QuadCount * 6; }
00062         };
00063         static void ResetStats();
00064         static Statistics GetStats();
00065
00066     private:
00067         static void FlushAndReset();
00068         static void StartBatch();
00069     };
00070
00071 }
```

10.134 Vesper/src/Vesper/Renderer/RendererAPI.cpp File Reference

```
#include "vzpch.h"
#include "RendererAPI.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.135 Vesper/src/Vesper/Renderer/RendererAPI.h File Reference

```
#include <glm/glm.hpp>
#include "VertexArray.h"
```

Classes

- class Vesper::RendererAPI

Namespaces

- namespace Vesper

  TEMPORARY.

10.136 RendererAPI.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <glm/glm.hpp>
00004
00005 #include "VertexArray.h"
00006
00007 namespace Vesper {
00008
00009     class RendererAPI {
00010     public:
00011         enum class API {
00012             None = 0,
00013             OpenGL = 1,
00014         };
00015
00016     public:
00017         virtual ~RendererAPI() = default;
00018         virtual void Init() = 0;
00019         virtual void SetViewport(uint32_t x, uint32_t y, uint32_t width, uint32_t height) = 0;
00020         virtual void SetClearColor(const glm::vec4& color) = 0;
00021         virtual void Clear() = 0;
00022
00023         virtual void DrawIndexed(const Ref<VertexArray>& vertexArray, uint32_t indexCount = 0) = 0;
00024
00025         inline static API GetAPI() { return s_API; }
00026     private:
00027         static API s_API;
00028     };
00029
00030
00031 }
```

10.137 Vesper/src/Vesper/Renderer/Shader.cpp File Reference

```
#include "vzpch.h"
#include "Shader.h"
#include "Renderer.h"
#include "RenderAPI/OpenGL/OpenGLShader.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.138 Vesper/src/Vesper/Renderer/Shader.h File Reference

```
#include <string>
#include <unordered_map>
#include <glm/glm.hpp>
```

Classes

- class Vesper::Shader
- class Vesper::ShaderLibrary

Namespaces

- namespace Vesper

  TEMPORARY.

10.139 Shader.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <string>
00004 #include <unordered_map>
00005 #include <glm/glm.hpp>
00006
00007 namespace Vesper {
00008
00009     class Shader
00010     {
00011     public:
00012
00013         virtual ~Shader() = default;
00014
00015         virtual void Bind() const = 0;
00016         virtual void Unbind() const = 0;
00017
00018         virtual void SetMat4(const std::string& name, const glm::mat4& value) = 0;
00019
00020         virtual void SetFloat4(const std::string& name, const glm::vec4& value) = 0;
00021         virtual void SetFloat3(const std::string& name, const glm::vec3& value) = 0;
00022         virtual void SetFloat(const std::string& name, float value) = 0;
00023
00024         virtual void SetInt(const std::string& name, int value) = 0;
00025         virtual void SetIntArray(const std::string& name, int* values, uint32_t count) = 0;
00026
00027
00028
00029         static Ref<Shader> Create(const std::string& name, const std::string& vertexSrc, const
      std::string& fragmentSrc);
00030         static Ref<Shader> Create(const std::string& filepath);
00031
00032         virtual const std::string& GetName() const = 0;
00033
00034     };
00035
00036     class ShaderLibrary
00037     {
00038     public:
00039         void Add(const std::string& name, const Ref<Shader>& shader);
00040         void Add(const Ref<Shader>& shader);
00041         Ref<Shader> Load(const std::string& filepath);
00042         Ref<Shader> Load(const std::string& name, const std::string& filepath);
00043
00044         Ref<Shader> Get(const std::string& name);
00045         bool Exists(const std::string& name) const;
00046
00047     private:
00048         std::unordered_map<std::string, Ref<Shader>> m_Shaders;
00049
00050     };
00051
00052 }
```

10.140 Vesper/src/Vesper/Renderer/SubTexture2D.cpp File Reference

```
#include "vzpch.h"
#include "SubTexture2D.h"
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.141 Vesper/src/Vesper/Renderer/SubTexture2D.h File Reference

```
#include <glm/glm.hpp>
#include "Texture.h"
```

Classes

- class Vesper::SubTexture2D

Namespaces

- namespace Vesper
  TEMPORARY.

10.142 SubTexture2D.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <glm/glm.hpp>
00004
00005 #include "Texture.h"
00006
00007
00008 namespace Vesper {
00009
00010     class SubTexture2D
00011     {
00012     public:
00013         SubTexture2D(const Ref<Texture2D>& texture, const glm::vec2& min, const glm::vec2& max);
00014
00015         const Ref<Texture2D> GetTexture() { return m_Texture; }
00016         glm::vec2* GetTexCoords() { return m_TexCoords; }
00017
00018         static Ref<SubTexture2D> CreateFromCoords(const Ref<Texture2D>& texture, const glm::vec2&
      coords, const glm::vec2& cellSize, const glm::vec2& spriteSize = {1, 1});
00019     private:
00020         Ref<Texture2D> m_Texture;
00021         glm::vec2 m_TexCoords[4];
00022     };
00023
00024 }
```

10.143 Vesper/src/Vesper/Renderer/Texture.cpp File Reference

```
#include "vzpch.h"
#include "Texture.h"
#include "Renderer.h"
#include <string>
#include "RenderAPI/OpenGL/OpenGLTexture.h"
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.144 Vesper/src/Vesper/Renderer/Texture.h File Reference

Classes

- class Vesper::Texture
- class Vesper::Texture2D
- class Vesper::TextureLibrary

Namespaces

- namespace Vesper
  TEMPORARY.

10.145 Texture.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 namespace Vesper {
00004     class Texture2D;
00005     class Texture
00006     {
00007     public:
00008         virtual ~Texture() = default;
00009
00010         virtual uint32_t GetWidth() const = 0;
00011         virtual uint32_t GetHeight() const = 0;
00012         virtual uint32_t GetRendererID() const = 0;
00013
00014         virtual void Bind(uint32_t slot = 0) const = 0;
00015         virtual void SetData(void* data, uint32_t size) = 0;
00016
00017         virtual bool operator==(const Texture2D& other) const = 0;
00018         virtual std::string GetName() const = 0;
00019     };
00020
00021     class Texture2D : public Texture
00022     {
00023     public:
00024         static Ref<Texture2D> Create(uint32_t width, uint32_t height);
00025         static Ref<Texture2D> Create(const std::string& path);
00026     };
00027
00028
00029     class TextureLibrary
00030     {
00031     public:
00032         void Add(const std::string& name, const Ref<Texture2D>& texture);
00033         void Add(const Ref<Texture2D>& texture);
00034         Ref<Texture2D> Load(const std::string& filepath);
00035         Ref<Texture2D> Load(const std::string& name, const std::string& filepath);
00036         Ref<Texture2D> Get(const std::string& name) const;
00037         bool Exists(const std::string& name) const;
00038     private:
00039         std::unordered_map<std::string, Ref<Texture2D>> m_Textures;
00040
00041     };
00042 }
```

10.146 Vesper/src/Vesper/Renderer/UniformBuffer.cpp File Reference

```
#include "vzpch.h"
#include "UniformBuffer.h"
#include "Vesper/Renderer/Renderer.h"
#include "RenderAPI/OpenGL/OpenGLUniformBuffer.h"
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.147 Vesper/src/Vesper/Renderer/UniformBuffer.h File Reference

```
#include "Vesper/Core/Base.h"
```

Classes

- class Vesper::UniformBuffer

Namespaces

- namespace Vesper
  TEMPORARY.

10.148 UniformBuffer.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Vesper/Core/Base.h"
00004
00005 namespace Vesper {
00006
00007     class UniformBuffer
00008     {
00009     public:
00010         virtual ~UniformBuffer() {}
00011         virtual void SetData(const void* data, uint32_t size, uint32_t offset = 0) = 0;
00012
00013         static Ref<UniformBuffer> Create(uint32_t size, uint32_t binding);
00014     };
00015
00016 }
```

10.149 Vesper/src/Vesper/Renderer/VertexArray.cpp File Reference

```
#include "vzpch.h"
#include "VertexArray.h"
#include "Renderer.h"
#include "RenderAPI/OpenGL/OpenGLVertexArray.h"
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.150 Vesper/src/Vesper/Renderer/VertexArray.h File Reference

```
#include <memory>
#include "Vesper/Renderer/Buffer.h"
```

Classes

- class Vesper::VertexArray

Namespaces

- namespace Vesper
  TEMPORARY.

10.151 VertexArray.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <memory>
00004 #include "Vesper/Renderer/Buffer.h"
00005
00006 namespace Vesper {
00007
00008
00009     class VertexArray
00010     {
00011     public:
00012         virtual ~VertexArray() {}
00013
00014         virtual void Bind() const = 0;
00015         virtual void Unbind() const = 0;
00016
00017         virtual void AddVertexBuffer(const Ref<VertexBuffer>& vertexBuffer) = 0;
00018         virtual void SetIndexBuffer(const Ref<IndexBuffer>& indexBuffer) = 0;
00019
00020         virtual const std::vector<Ref<VertexBuffer»& GetVertexBuffers() = 0;
00021         virtual const Ref<IndexBuffer>& GetIndexBuffer() const = 0;
00022
00023         static Ref<VertexArray> Create();
00024     };
00025 }
```

10.152 Vesper/src/Vesper/Scene/Components.h File Reference

```
#include "Vesper/Renderer/Texture.h"
#include "Vesper/Renderer/SubTexture2D.h"
#include "SceneCamera.h"
#include "Vesper/Core/Random.h"
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/quaternion.hpp>
```

Classes

- struct Vesper::UUID
- struct Vesper::UUIDComponent
- struct Vesper::NameComponent
- struct Vesper::TransformComponent
- struct Vesper::SpriteRendererComponent
- struct Vesper::SubTextureComponent
- struct Vesper::TextureAnimationComponent
- struct Vesper::CameraComponent
- struct Vesper::NativeScriptComponent

Namespaces

- namespace Vesper

  TEMPORARY.

Macros

- #define GLM_ENABLE_EXPERIMENTAL

10.152.1 Macro Definition Documentation

10.152.1.1 GLM_ENABLE_EXPERIMENTAL

```
#define GLM_ENABLE_EXPERIMENTAL
```

10.153 Components.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Vesper/Renderer/Texture.h"
00003 #include "Vesper/Renderer/SubTexture2D.h"
00004 #include "SceneCamera.h"
00005 #include "Vesper/Core/Random.h"
00006
00007 #include <glm/glm.hpp>
00008 #include <glm/gtc/matrix_transform.hpp>
00009 #define GLM_ENABLE_EXPERIMENTAL
00010 #include <glm/gtx/quaternion.hpp>
00011
00012 namespace Vesper
00013 {
00014     struct UUID {
00015         std::string ID;
00016         UUID() { ID = Random::UUID(); }
00017         UUID(const std::string& id)
00018             : ID{ id } {
00019         }
00020         operator std::string& () { return ID; }
00021         operator const std::string& () const { return ID; }
00022     };
00023
00024     struct UUIDComponent {
00025         UUID ID;
00026         UUIDComponent()
00027             : ID() {
00028         }
00029         UUIDComponent(const UUIDComponent&) = default;
```

```
00030         UUIDComponent(const std::string& id)
00031             : ID{ id } {
00032         }
00033
00034     };
00035
00036     struct NameComponent
00037     {
00038         std::string Name;
00039         NameComponent() = default;
00040         NameComponent(const NameComponent&) = default;
00041         NameComponent(const std::string& name)
00042             : Name(name) {
00043         }
00044         operator std::string& () { return Name; }
00045         operator const std::string& () const { return Name; }
00046         std::string& GetName() { return Name; }
00047     };
00048
00049     struct TransformComponent
00050     {
00051         glm::vec3 Translation = { 0.0f, 0.0f, 0.0f };
00052         glm::vec3 Rotation = { 0.0f, 0.0f, 0.0f };
00053         glm::vec3 Scale = { 1.0f, 1.0f, 1.0f };
00054
00055         TransformComponent() = default;
00056         TransformComponent(const TransformComponent&) = default;
00057         TransformComponent(const glm::vec3& translation)
00058             : Translation(translation) {
00059         }
00060
00061         glm::mat4 GetTransform() const
00062         {
00063             glm::mat4 rotation = glm::toMat4(glm::quat(Rotation));
00064
00065             return glm::translate(glm::mat4(1.0f), Translation)
00066                 * rotation
00067                 * glm::scale(glm::mat4(1.0f), Scale);
00068         }
00069     };
00070
00071     struct SpriteRendererComponent
00072     {
00073         glm::vec4 Color{ 1.0f, 1.0f, 1.0f, 1.0f };
00074         Ref<Texture2D> Texture = nullptr;
00075         float TilingFactor = 1.0f;
00076
00077         SpriteRendererComponent() = default;
00078         SpriteRendererComponent(const SpriteRendererComponent&) = default;
00079         SpriteRendererComponent(const glm::vec4& color)
00080             : Color(color) {
00081         }
00082
00083         operator glm::vec4& () { return Color; }
00084         operator const glm::vec4& () const { return Color; }
00085
00086         glm::vec4& GetColor() { return Color; }
00087
00088         bool TextureEnabled = false;
00089         bool Billboard = false;
00090     };
00091
00092     struct SubTextureComponent
00093     {
00094         Ref<SubTexture2D> SubTexture;
00095         glm::vec2 TilingFactor = { 1.0f, 1.0f };
00096         glm::vec2 Offset = { 0.0f, 0.0f };
00097         SubTextureComponent() = default;
00098         SubTextureComponent(const Ref<Texture2D>& texture)
00099             : SubTexture(SubTexture2D::CreateFromCoords(texture, { 0, 0 }, { texture->GetWidth(),
      texture->GetHeight() })) {
00100         }
00101         SubTextureComponent(const Ref<SubTexture2D>& subTexture)
00102             : SubTexture(subTexture) {
00103         }
00104
00105         void SetTexture(const Ref<Texture2D>& texture) {
00106             SubTexture = SubTexture2D::CreateFromCoords(texture, { 0, 0 }, { texture->GetWidth(),
      texture->GetHeight() });
00107         }
00108
00109         void SetTilingFactor(const glm::vec2& tiling) {
00110             TilingFactor = tiling;
00111         }
00112
00113         void SetOffset(const glm::vec2& offset) {
00114             Offset = offset;
```

```
00115         }
00116
00117         operator Ref<SubTexture2D>& () { return SubTexture; }
00118         operator const Ref<SubTexture2D>& () const { return SubTexture; }
00119         Ref<SubTexture2D>& GetSubTexture() { return SubTexture; }
00120     };
00121
00122     // Animates through a series of sub textures (can be used with full textures)
00123     struct TextureAnimationComponent
00124     {
00125         std::vector<Ref<SubTexture2D» SubTextures;
00126         uint32_t CurrentFrame = 0;
00127         float FrameTime = 0.6f; // Time per frame in seconds
00128         float TimeAccumulator = 0.0f; // per-instance accumulator
00129
00130         TextureAnimationComponent() = default;
00131         TextureAnimationComponent(const TextureAnimationComponent&) = default;
00132         TextureAnimationComponent(const std::vector<Ref<SubTexture2D»& subTextures, float frameTime)
00133             : SubTextures(subTextures), FrameTime(frameTime) {
00134         }
00135         operator std::vector<Ref<SubTexture2D»& () { return SubTextures; }
00136         operator const std::vector<Ref<SubTexture2D»& () const { return SubTextures; }
00137
00138         std::vector<Ref<SubTexture2D»& GetSubTextures() { return SubTextures; }
00139         uint32_t GetCurrentFrame() const { return CurrentFrame; }
00140
00141         void Update(float deltaTime) {
00142             if (SubTextures.empty() || FrameTime <= 0.0f)
00143                 return;
00144
00145             TimeAccumulator += deltaTime;
00146             while (TimeAccumulator >= FrameTime) {
00147                 CurrentFrame = (CurrentFrame + 1) % static_cast<uint32_t>(SubTextures.size());
00148                 TimeAccumulator -= FrameTime;
00149             }
00150         }
00151
00152
00153     };
00154
00155     struct CameraComponent
00156     {
00157         SceneCamera Camera;
00158         bool Primary = true;
00159         bool FixedAspectRatio = false;
00160
00161         CameraComponent() = default;
00162         CameraComponent(const CameraComponent&) = default;
00163
00164     };
00165
00166     class ScriptableEntity;
00167     class Timestep;
00168
00169     struct NativeScriptComponent
00170     {
00171         ScriptableEntity* Instance = nullptr;
00172
00173         ScriptableEntity* (*InstantiateScript)();
00174         void (*DestroyScript)(NativeScriptComponent*);
00175
00176         template<typename T>
00177         void Bind()
00178         {
00179             InstantiateScript = []() { return static_cast<ScriptableEntity*> (new T()); };
00180             DestroyScript = [](NativeScriptComponent* nsc) { delete nsc->Instance; nsc->Instance =
      nullptr; };
00181         }
00182     };
00183
00184 }
```

10.154 Vesper/src/Vesper/Scene/Entity.cpp File Reference

```
#include "vzpch.h"
#include "Entity.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.155 Vesper/src/Vesper/Scene/Entity.h File Reference

```
#include "entt.hpp"
#include "Scene.h"
```

Classes

- class Vesper::Entity

Namespaces

- namespace Vesper

  TEMPORARY.

10.156 Entity.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "entt.hpp"
00004
00005 #include "Scene.h"
00006
00007 namespace Vesper {
00008
00009     class Entity
00010     {
00011     public:
00012         Entity() = default;
00013         Entity(entt::entity handle, Scene* scene);
00014         Entity(const Entity& other) = default;
00015
00016         template<typename T>
00017         bool HasComponent() const
00018         {
00019             return m_Scene->m_Registry.all_of<T>(m_EntityID);
00020         }
00021
00022         template<typename T, typename... Args>
00023         T& AddComponent(Args&&... args)
00024         {
00025             VZ_CORE_ASSERT(!HasComponent<T>(), "Entity already has component!");
00026             T& component = m_Scene->m_Registry.emplace<T>(m_EntityID, std::forward<Args>(args)...);
00027             m_Scene->OnComponentAdded<T>(*this, component);
00028             return component;
00029         }
00030
00031
00032         template<typename T, typename... Args>
00033         T& AddOrReplaceComponent(Args&&... args)
00034         {
00035             return m_Scene->m_Registry.emplace_or_replace<T>(m_EntityID, std::forward<Args>(args)...);
00036         }
00037
00038         template<typename T>
00039         T& GetComponent()
00040         {
```

```
00041             VZ_CORE_ASSERT(HasComponent<T>(), "Entity does not have component!");
00042             return m_Scene->m_Registry.get<T>(m_EntityID);
00043         }
00044 
00045         template<typename T, typename... Args>
00046         T& GetOrAddComponent(Args&&... args)
00047         {
00048             if (HasComponent<T>())
00049                 return GetComponent<T>();
00050             else
00051                 return AddComponent<T>(std::forward<Args>(args)...);
00052         }
00053 
00054         template<typename T>
00055         void RemoveComponent()
00056         {
00057             VZ_CORE_ASSERT(HasComponent<T>(), "Entity does not have component!");
00058             m_Scene->m_Registry.remove<T>(m_EntityID);
00059         }
00060 
00061 
00062         const UUID& GetID()
00063         {
00064             return GetComponent<UUIDComponent>().ID;
00065         }
00066 
00067         const std::string& GetName()
00068         {
00069             return GetComponent<NameComponent>().Name;
00070         }
00071 
00072 
00073         operator bool() const { return m_EntityID != entt::null; }
00074         operator entt::entity() const { return m_EntityID; }
00075         operator uint32_t() const { return (uint32_t)m_EntityID; }
00076         bool operator==(const Entity& other) const {    return m_EntityID == other.m_EntityID &&
      m_Scene == other.m_Scene;  }
00077         bool operator!=(const Entity& other) const {    return !(*this == other);   }
00078 
00079     private:
00080         entt::entity m_EntityID {entt::null};
00081         Scene* m_Scene = nullptr;
00082     };
00083 
00084 
00085 }
```

10.157 Vesper/src/Vesper/Scene/Scene.cpp File Reference

```
#include "vzpch.h"
#include "Scene.h"
#include "Vesper/Core/Log.h"
#include "Vesper/Renderer/Renderer2D.h"
#include "Vesper/Scene/Entity.h"
#include "Vesper/Scene/ScriptableEntity.h"
#include "Vesper/Renderer/EditorCamera.h"
```

Namespaces

- namespace Vesper

  TEMPORARY.

10.158 Vesper/src/Vesper/Scene/Scene.h File Reference

```
#include <entt.hpp>
#include "Components.h"
#include "Vesper/Core/Timestep.h"
```

Classes

- class Vesper::Scene

Namespaces

- namespace Vesper

  TEMPORARY.

10.159 Scene.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include <entt.hpp>
00003 #include "Components.h"
00004 #include "Vesper/Core/Timestep.h"
00005
00006
00007 namespace Vesper {
00008
00009     class Entity;
00010     class EditorCamera;
00011
00012     class Scene
00013     {
00014     public:
00015         Scene();
00016         Scene(const std::string& name);
00017         ~Scene();
00018
00019         // Temp-> add entity wrapper later
00020         Entity CreateEntity(const std::string& name = std::string());
00021         Entity CreateEntity(const std::string& name, const std::string& uuid);
00022         void DestroyEntity(Entity entity);
00023
00024         void OnUpdateRuntime(Timestep ts);
00025         void OnUpdateEditor(Timestep ts, EditorCamera& camera);
00026         void OnViewportResize(uint32_t width, uint32_t height);
00027         Entity GetPrimaryCameraEntity();
00028     private:
00029         template<typename T>
00030         void OnComponentAdded(Entity entity, T& component);
00031
00032     private:
00033         std::string m_Name;
00034         entt::registry m_Registry;
00035         uint32_t m_ViewportWidth = 160, m_ViewportHeight = 90;
00036         friend class Entity;
00037         friend class SceneSerializer;
00038         friend class SceneHierarchyPanel;
00039         /// TODO: friend class SceneCamera;
00040         /// TODO: friend class SceneRenderer;
00041
00042         void SetName(const std::string& name) { m_Name = name; }
00043         const std::string& GetName() const { return m_Name; }
00044     };
00045
00046
00047 }
```

10.160 Vesper/src/Vesper/Scene/SceneCamera.cpp File Reference

```
#include "vzpch.h"
#include "SceneCamera.h"
#include <glm/gtc/matrix_transform.hpp>
```

Namespaces

- namespace Vesper
  TEMPORARY.

10.161 Vesper/src/Vesper/Scene/SceneCamera.h File Reference

```
#include "Vesper/Renderer/Camera.h"
```

Classes

- class Vesper::SceneCamera

Namespaces

- namespace Vesper
  TEMPORARY.

10.162 SceneCamera.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Vesper/Renderer/Camera.h"
00003
00004 namespace Vesper {
00005
00006     class SceneCamera : public Camera
00007     {
00008     public:
00009         enum class ProjectionType { Perspective = 0, Orthographic = 1 };
00010     public:
00011         SceneCamera();
00012         virtual ~SceneCamera() = default;
00013
00014         void SetOrthographic(float size, float nearClip, float farClip);
00015         void SetPerspective(float verticalFOV, float nearClip, float farClip);
00016
00017         void SetViewportSize(uint32_t width, uint32_t height);
00018
00019         float GetPerspectiveVerticalFOV() const { return m_PerspectiveFOV; }
00020         void SetPerspectiveVerticalFOV(float verticalFov) { m_PerspectiveFOV = verticalFov;
      RecalculateProjection(); }
00021         float GetPerspectiveNearClip() const { return m_PerspectiveNear; }
00022         void SetPerspectiveNearClip(float nearClip) { m_PerspectiveNear = nearClip;
      RecalculateProjection(); }
00023         float GetPerspectiveFarClip() const { return m_PerspectiveFar; }
00024         void SetPerspectiveFarClip(float farClip) { m_PerspectiveFar = farClip;
      RecalculateProjection(); }
00025
00026         float GetOrthographicSize() const { return m_OrthographicSize; }
00027         void SetOrthographicSize(float size) { m_OrthographicSize = size; RecalculateProjection(); }
00028         float GetOrthographicNearClip() const { return m_OrthographicNear; }
00029         void SetOrthographicNearClip(float nearClip) { m_OrthographicNear = nearClip;
      RecalculateProjection(); }
00030         float GetOrthographicFarClip() const { return m_OrthographicFar; }
00031         void SetOrthographicFarClip(float farClip) { m_OrthographicFar = farClip;
      RecalculateProjection(); }
00032
00033         ProjectionType GetProjectionType() const { return m_ProjectionType; }
00034         void SetProjectionType(ProjectionType type) { m_ProjectionType = type;
      RecalculateProjection(); }
00035     private:
00036         void RecalculateProjection();
```

```
00037     private:
00038         ProjectionType m_ProjectionType = ProjectionType::Orthographic;
00039
00040         float m_PerspectiveFOV = glm::radians(45.0f);
00041         float m_PerspectiveNear = 0.01f, m_PerspectiveFar = 1000.0f;
00042
00043         float m_OrthographicSize = 10.0f;
00044         float m_OrthographicNear = -1.0f, m_OrthographicFar = 1.0f;
00045
00046         float m_AspectRatio = 0.0f;
00047     };
00048
00049
00050 }
```

10.163 Vesper/src/Vesper/Scene/SceneSerializer.cpp File Reference

```
#include "vzpch.h"
#include "SceneSerializer.h"
#include "Entity.h"
#include "Components.h"
#include <fstream>
#include <yaml-cpp/yaml.h>
```

Classes

- struct YAML::convert< glm::vec2 >
- struct YAML::convert< glm::vec3 >
- struct YAML::convert< glm::vec4 >

Namespaces

- namespace YAML
- namespace Vesper

  TEMPORARY.

Functions

- YAML::Emitter & YAML::operator<< (YAML::Emitter &out, const glm::vec2 &v)
- YAML::Emitter & YAML::operator<< (YAML::Emitter &out, const glm::vec3 &v)
- YAML::Emitter & YAML::operator<< (YAML::Emitter &out, const glm::vec4 &v)
- static void Vesper::SerializeEntity (YAML::Emitter &out, Entity entity)

10.164 Vesper/src/Vesper/Scene/SceneSerializer.h File Reference

```
#include "Vesper/Core/Base.h"
#include "Scene.h"
```

Classes

- class Vesper::SceneSerializer

Namespaces

- namespace Vesper

  TEMPORARY.

10.165 SceneSerializer.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #include "Vesper/Core/Base.h"
00003 #include "Scene.h"
00004
00005 namespace Vesper {
00006     class SceneSerializer
00007     {
00008     public:
00009         SceneSerializer(const Ref<Scene>& scene);
00010
00011         void Serialize(const std::string& filepath);
00012         void SerializeRuntime(const std::string& filepath);
00013
00014         bool Deserialize(const std::string& filepath);
00015         bool DeserializeRuntime(const std::string& filepath);
00016     private:
00017         Ref<Scene> m_Scene;
00018     };
00019 }
```

10.166 Vesper/src/Vesper/Scene/ScriptableEntity.h File Reference

```
#include "Entity.h"
```

Classes

- class Vesper::ScriptableEntity

Namespaces

- namespace Vesper

  TEMPORARY.

10.167 ScriptableEntity.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "Entity.h"
00004
00005 namespace Vesper {
00006
00007     class ScriptableEntity
00008     {
00009     public:
00010         virtual ~ScriptableEntity() {};
00011
00012         template<typename T>
```

```
00013         T& GetComponent()
00014         {
00015             return m_Entity.GetComponent<T>();
00016         }
00017
00018     protected:
00019         virtual void OnCreate() {}
00020         virtual void OnDestroy() {}
00021         virtual void OnUpdate(Timestep ts) {}
00022     private:
00023         Entity m_Entity;
00024         friend class Scene;
00025     };
00026 }
```

10.168 Vesper/src/Vesper/Utils/PlatformUtils.h File Reference

```
#include <string>
```

Classes

- class Vesper::FileDialogs
- class Vesper::FileSystem

Namespaces

- namespace Vesper

  TEMPORARY.

10.169 PlatformUtils.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <string>
00004
00005 namespace Vesper {
00006
00007     class FileDialogs
00008     {
00009     public:
00010         // These return empty string if cancelled
00011         static std::string OpenFile(const char* filter);
00012         static std::string SaveFile(const char* filter);
00013     };
00014
00015     class FileSystem
00016     {
00017     public:
00018         static void Initialize();
00019         static std::string GetCurrentWorkingDirectory();
00020         static std::string GetAbsolutePath(const std::string& relativePath);
00021         static std::string GetTravelingUpPath(const std::string& path);
00022         static bool IsInitialized() { return m_Initialized; }
00023
00024         static bool m_Initialized;
00025         static std::string m_RootEngineDirectory;
00026         static std::string m_RootEditorDirectory;
00027         static std::string m_ResourcesDirectory;
00028         static std::string m_AssetsDirectory;
00029         static std::string m_ProjectsDirectory;
00030         static std::string m_CurrentProjectDirectory;
00031     };
00032
00033
00034 };
```

10.170 Vesper/src/vzpch.cpp File Reference

```
#include "vzpch.h"
```

10.171 Vesper/src/vzpch.h File Reference

```
#include <iostream>
#include <memory>
#include <utility>
#include <algorithm>
#include <random>
#include <functional>
#include <string>
#include <sstream>
#include <array>
#include <vector>
#include <unordered_map>
#include <unordered_set>
#include "Vesper/Core/Log.h"
#include "Vesper/Debug/Instrumentor.h"
```

10.172 vzpch.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include <iostream>
00004 #include <memory>
00005 #include <utility>
00006 #include <algorithm>
00007 #include <random>
00008 #include <functional>
00009
00010 #include <string>
00011 #include <sstream>
00012 #include <array>
00013 #include <vector>
00014 #include <unordered_map>
00015 #include <unordered_set>
00016
00017 #include "Vesper/Core/Log.h"
00018
00019
00020 #include "Vesper/Debug/Instrumentor.h"
00021
00022
00023 #ifdef VZ_PLATFORM_WINDOWS
00024     #include <windows.h>
00025 #endif
```