

Vesper

0.5.1

Generated by Doxygen 1.16.1

| | |
|---|-----------|
| 1 Vesper | 1 |
| 1.1 Quick overview | 1 |
| 1.2 New Project example code | 2 |
| 2 Todo List | 3 |
| 3 Directory Hierarchy | 5 |
| 3.1 Directories | 5 |
| 4 Namespace Index | 15 |
| 4.1 Namespace List | 15 |
| 5 Hierarchical Index | 17 |
| 5.1 Class Hierarchy | 17 |
| 6 Class Index | 21 |
| 6.1 Class List | 21 |
| 7 File Index | 25 |
| 7.1 File List | 25 |
| 8 Directory Documentation | 29 |
| 8.1 Vesper/src/Vesper/App Directory Reference | 29 |
| 8.2 Vesper/src/Vesper/Core Directory Reference | 29 |
| 8.3 Vesper/src/Vesper/Debug Directory Reference | 30 |
| 8.4 docs Directory Reference | 30 |
| 8.5 Vesper/src/Vesper/Events Directory Reference | 30 |
| 8.6 Vesper/src/Vesper/ImGui Directory Reference | 30 |
| 8.7 Vesper/src/Vesper/ImGuiDirectory Reference | 31 |
| 8.8 Vesper/src/Vesper/Input Directory Reference | 31 |
| 8.9 Vesper/src/RenderAPI/OpenGL Directory Reference | 31 |
| 8.10 Vesper-Editor/src/Panels Directory Reference | 31 |
| 8.11 Vesper/src/Vesper/ParticleSystem Directory Reference | 32 |
| 8.12 Vesper/src/Platform Directory Reference | 32 |
| 8.13 Vesper/src/RenderAPI Directory Reference | 32 |
| 8.14 Vesper/src/Vesper/Renderer Directory Reference | 32 |
| 8.15 Vesper/src/Vesper/Scene Directory Reference | 33 |
| 8.16 Vesper-Editor/src Directory Reference | 33 |
| 8.17 Vesper/src Directory Reference | 33 |
| 8.18 Vesper/src/Vesper/Utils Directory Reference | 33 |
| 8.19 Vesper Directory Reference | 34 |
| 8.20 Vesper/src/Vesper Directory Reference | 34 |
| 8.21 Vesper-Editor Directory Reference | 34 |
| 8.22 Vesper/src/Platform/Windows Directory Reference | 34 |

| | |
|---|-----------|
| 9 Namespace Documentation | 35 |
| 9.1 InstrumentorUtils Namespace Reference | 35 |
| 9.1.1 Class Documentation | 35 |
| 9.1.1.1 struct InstrumentorUtils::ChangeResult | 35 |
| 9.1.2 Function Documentation | 36 |
| 9.1.2.1 CleanupOutputString() | 36 |
| 9.2 Vesper Namespace Reference | 36 |
| 9.2.1 Detailed Description | 41 |
| 9.2.2 Class Documentation | 41 |
| 9.2.2.1 struct Vesper::ApplicationSettings | 41 |
| 9.2.2.2 struct Vesper::ParticleProps | 41 |
| 9.2.2.3 struct Vesper::FramebufferSpecification | 42 |
| 9.2.2.4 struct Vesper::ProfileResult | 42 |
| 9.2.2.5 struct Vesper::InstrumentationSession | 42 |
| 9.2.2.6 struct Vesper::QuadVertex | 42 |
| 9.2.3 Typedef Documentation | 43 |
| 9.2.3.1 FloatingPointMicroseconds | 43 |
| 9.2.3.2 KeyCode | 43 |
| 9.2.3.3 MouseCode | 43 |
| 9.2.3.4 Ref | 43 |
| 9.2.3.5 Scope | 43 |
| 9.2.4 Enumeration Type Documentation | 44 |
| 9.2.4.1 EventCategory | 44 |
| 9.2.4.2 EventType | 44 |
| 9.2.4.3 ShaderDataType | 45 |
| 9.2.4.4 WindowMode | 45 |
| 9.2.5 Function Documentation | 46 |
| 9.2.5.1 CreateApplication() | 46 |
| 9.2.5.2 CreateRef() | 46 |
| 9.2.5.3 CreateScope() | 46 |
| 9.2.5.4 DisplayVesperInfo_ImGui() | 47 |
| 9.2.5.5 DrawComponent() | 48 |
| 9.2.5.6 DrawVec2Control() | 48 |
| 9.2.5.7 DrawVec3Control() | 49 |
| 9.2.5.8 format_as() | 50 |
| 9.2.5.9 GLFWErrorCallback() | 51 |
| 9.2.5.10 SerializeEntity() | 51 |
| 9.2.5.11 ShaderDataTypeSize() | 52 |
| 9.2.5.12 ShaderDataTypeToOpenGLBaseType() | 52 |
| 9.2.5.13 ShaderTypeFromString() | 52 |
| 9.2.5.14 SubTextureEdit() | 53 |
| 9.2.6 Variable Documentation | 53 |

| | |
|--|----|
| 9.2.6.1 s_Data | 53 |
| 9.2.6.2 s_GlfwInitialized | 53 |
| 9.2.6.3 s_MaxFramebufferSize | 53 |
| 9.3 Vesper::Color Namespace Reference | 54 |
| 9.3.1 Detailed Description | 54 |
| 9.3.2 Function Documentation | 55 |
| 9.3.2.1 Black() | 55 |
| 9.3.2.2 Blue() | 55 |
| 9.3.2.3 Brown() | 55 |
| 9.3.2.4 Cyan() | 55 |
| 9.3.2.5 Gray() | 55 |
| 9.3.2.6 Green() | 55 |
| 9.3.2.7 Indigo() | 56 |
| 9.3.2.8 Magenta() | 56 |
| 9.3.2.9 Orange() | 56 |
| 9.3.2.10 Pink() | 56 |
| 9.3.2.11 Purple() | 56 |
| 9.3.2.12 Red() | 56 |
| 9.3.2.13 SetAlpha() | 56 |
| 9.3.2.14 StripAlpha() | 57 |
| 9.3.2.15 Transparent() | 57 |
| 9.3.2.16 White() | 57 |
| 9.3.2.17 Yellow() | 57 |
| 9.4 Vesper::Key Namespace Reference | 57 |
| 9.4.1 Detailed Description | 58 |
| 9.4.2 Enumeration Type Documentation | 58 |
| 9.4.2.1 anonymous enum | 58 |
| 9.5 Vesper::Math Namespace Reference | 63 |
| 9.5.1 Function Documentation | 63 |
| 9.5.1.1 DecomposeTransform() | 63 |
| 9.6 Vesper::Mouse Namespace Reference | 64 |
| 9.6.1 Detailed Description | 64 |
| 9.6.2 Enumeration Type Documentation | 65 |
| 9.6.2.1 anonymous enum | 65 |
| 9.7 Vesper::Random Namespace Reference | 65 |
| 9.7.1 Function Documentation | 66 |
| 9.7.1.1 Bool1() | 66 |
| 9.7.1.2 Char() | 66 |
| 9.7.1.3 Float1() | 66 |
| 9.7.1.4 Float2() | 66 |
| 9.7.1.5 Float3() | 67 |
| 9.7.1.6 Float4() | 67 |

| | |
|---|-----------|
| 9.7.1.7 GetRNG() | 67 |
| 9.7.1.8 HexString() | 67 |
| 9.7.1.9 RangeF1() | 67 |
| 9.7.1.10 RangeF1_Inclusive() | 68 |
| 9.7.1.11 RangeF2() [1/3] | 68 |
| 9.7.1.12 RangeF2() [2/3] | 68 |
| 9.7.1.13 RangeF2() [3/3] | 68 |
| 9.7.1.14 RangeF3() [1/3] | 68 |
| 9.7.1.15 RangeF3() [2/3] | 69 |
| 9.7.1.16 RangeF3() [3/3] | 69 |
| 9.7.1.17 RangeF4() | 69 |
| 9.7.1.18 Seed() | 69 |
| 9.7.1.19 String() | 69 |
| 9.7.1.20 UInt1() | 70 |
| 9.7.1.21 UUID() | 70 |
| 9.8 YAML Namespace Reference | 70 |
| 9.8.1 Function Documentation | 70 |
| 9.8.1.1 operator<<() [1/3] | 70 |
| 9.8.1.2 operator<<() [2/3] | 71 |
| 9.8.1.3 operator<<() [3/3] | 71 |
| 10 Class Documentation | 73 |
| 10.1 Vesper::Application Class Reference | 73 |
| 10.1.1 Detailed Description | 74 |
| 10.1.2 Constructor & Destructor Documentation | 75 |
| 10.1.2.1 Application() | 75 |
| 10.1.2.2 ~Application() | 75 |
| 10.1.3 Member Function Documentation | 75 |
| 10.1.3.1 Close() | 75 |
| 10.1.3.2 Get() | 76 |
| 10.1.3.3 GetImGuiLayer() | 76 |
| 10.1.3.4 GetWindow() | 76 |
| 10.1.3.5 OnEvent() | 76 |
| 10.1.3.6 OnWindowClose() | 77 |
| 10.1.3.7 OnWindowResize() | 77 |
| 10.1.3.8 PushLayer() | 77 |
| 10.1.3.9 PushOverlay() | 78 |
| 10.1.3.10 Run() | 78 |
| 10.1.4 Member Data Documentation | 78 |
| 10.1.4.1 m_ImGuiLayer | 78 |
| 10.1.4.2 m_LastFrameTime | 79 |
| 10.1.4.3 m_LayerStack | 79 |

| | |
|---|----|
| 10.1.4.4 m_Minimized | 79 |
| 10.1.4.5 m_Running | 79 |
| 10.1.4.6 m_Window | 79 |
| 10.1.4.7 s_Instance | 79 |
| 10.2 Vesper::AppRenderEvent Class Reference | 80 |
| 10.2.1 Detailed Description | 80 |
| 10.2.2 Constructor & Destructor Documentation | 81 |
| 10.2.2.1 AppRenderEvent() | 81 |
| 10.3 Vesper::AppTickEvent Class Reference | 81 |
| 10.3.1 Detailed Description | 82 |
| 10.3.2 Constructor & Destructor Documentation | 82 |
| 10.3.2.1 AppTickEvent() | 82 |
| 10.4 Vesper::AppUpdateEvent Class Reference | 82 |
| 10.4.1 Detailed Description | 83 |
| 10.4.2 Constructor & Destructor Documentation | 83 |
| 10.4.2.1 AppUpdateEvent() | 83 |
| 10.5 Vesper::BufferElement Struct Reference | 83 |
| 10.5.1 Detailed Description | 84 |
| 10.5.2 Constructor & Destructor Documentation | 84 |
| 10.5.2.1 BufferElement() [1/2] | 84 |
| 10.5.2.2 BufferElement() [2/2] | 84 |
| 10.5.3 Member Function Documentation | 84 |
| 10.5.3.1 GetComponentCount() | 84 |
| 10.5.4 Member Data Documentation | 85 |
| 10.5.4.1 Name | 85 |
| 10.5.4.2 Normalized | 85 |
| 10.5.4.3 Offset | 85 |
| 10.5.4.4 Size | 85 |
| 10.5.4.5 Type | 85 |
| 10.6 Vesper::BufferLayout Class Reference | 85 |
| 10.6.1 Detailed Description | 86 |
| 10.6.2 Constructor & Destructor Documentation | 86 |
| 10.6.2.1 BufferLayout() [1/2] | 86 |
| 10.6.2.2 BufferLayout() [2/2] | 86 |
| 10.6.3 Member Function Documentation | 87 |
| 10.6.3.1 begin() [1/2] | 87 |
| 10.6.3.2 begin() [2/2] | 87 |
| 10.6.3.3 CalculateOffsetsAndStride() | 87 |
| 10.6.3.4 end() [1/2] | 87 |
| 10.6.3.5 end() [2/2] | 87 |
| 10.6.3.6 GetElements() | 87 |
| 10.6.3.7 GetStride() | 87 |

| | |
|---|----|
| 10.6.4 Member Data Documentation | 88 |
| 10.6.4.1 m_Elements | 88 |
| 10.6.4.2 m_Stride | 88 |
| 10.7 Vesper::Camera Class Reference | 88 |
| 10.7.1 Constructor & Destructor Documentation | 88 |
| 10.7.1.1 Camera() [1/2] | 88 |
| 10.7.1.2 Camera() [2/2] | 89 |
| 10.7.1.3 ~Camera() | 89 |
| 10.7.2 Member Function Documentation | 89 |
| 10.7.2.1 GetProjection() | 89 |
| 10.7.3 Member Data Documentation | 89 |
| 10.7.3.1 m_Projection | 89 |
| 10.8 Vesper::CameraComponent Struct Reference | 89 |
| 10.8.1 Detailed Description | 90 |
| 10.8.2 Constructor & Destructor Documentation | 90 |
| 10.8.2.1 CameraComponent() [1/2] | 90 |
| 10.8.2.2 CameraComponent() [2/2] | 90 |
| 10.8.3 Member Data Documentation | 90 |
| 10.8.3.1 Camera | 90 |
| 10.8.3.2 FixedAspectRatio | 90 |
| 10.8.3.3 Primary | 90 |
| 10.9 YAML::convert< glm::vec2 > Struct Reference | 91 |
| 10.9.1 Member Function Documentation | 91 |
| 10.9.1.1 decode() [1/2] | 91 |
| 10.9.1.2 decode() [2/2] | 91 |
| 10.9.1.3 encode() [1/2] | 91 |
| 10.9.1.4 encode() [2/2] | 92 |
| 10.10 YAML::convert< glm::vec3 > Struct Reference | 92 |
| 10.10.1 Member Function Documentation | 92 |
| 10.10.1.1 decode() [1/2] | 92 |
| 10.10.1.2 decode() [2/2] | 92 |
| 10.10.1.3 encode() [1/2] | 93 |
| 10.10.1.4 encode() [2/2] | 93 |
| 10.11 YAML::convert< glm::vec4 > Struct Reference | 93 |
| 10.11.1 Member Function Documentation | 93 |
| 10.11.1.1 decode() [1/2] | 93 |
| 10.11.1.2 decode() [2/2] | 94 |
| 10.11.1.3 encode() [1/2] | 94 |
| 10.11.1.4 encode() [2/2] | 94 |
| 10.12 Vesper::EditorCamera Class Reference | 94 |
| 10.12.1 Constructor & Destructor Documentation | 96 |
| 10.12.1.1 EditorCamera() [1/2] | 96 |

| | |
|---|-----|
| 10.12.1.2 EditorCamera() [2/2] | 96 |
| 10.12.2 Member Function Documentation | 96 |
| 10.12.2.1 CalculatePosition() | 96 |
| 10.12.2.2 GetDistance() | 96 |
| 10.12.2.3 GetForwardDirection() | 97 |
| 10.12.2.4 GetOrientation() | 97 |
| 10.12.2.5 GetPitch() | 97 |
| 10.12.2.6 GetPosition() | 97 |
| 10.12.2.7 GetRightDirection() | 97 |
| 10.12.2.8 GetUpDirection() | 97 |
| 10.12.2.9 GetViewMatrix() | 97 |
| 10.12.2.10 GetViewProjection() | 97 |
| 10.12.2.11 GetYaw() | 98 |
| 10.12.2.12 MousePan() | 98 |
| 10.12.2.13 MouseRotate() | 98 |
| 10.12.2.14 MouseZoom() | 98 |
| 10.12.2.15 OnEvent() | 98 |
| 10.12.2.16 OnMouseScroll() | 99 |
| 10.12.2.17 OnUpdate() | 99 |
| 10.12.2.18 PanSpeed() | 99 |
| 10.12.2.19 RotationSpeed() | 99 |
| 10.12.2.20 SetDistance() | 100 |
| 10.12.2.21 SetPitch() | 100 |
| 10.12.2.22 SetPosition() | 100 |
| 10.12.2.23 SetViewportSize() | 100 |
| 10.12.2.24 SetYaw() | 100 |
| 10.12.2.25 UpdateProjection() | 100 |
| 10.12.2.26 UpdateView() | 101 |
| 10.12.2.27 ZoomSpeed() | 101 |
| 10.12.3 Member Data Documentation | 101 |
| 10.12.3.1 m_AspectRatio | 101 |
| 10.12.3.2 m_Distance | 101 |
| 10.12.3.3 m_FarClip | 101 |
| 10.12.3.4 m_FocalPoint | 101 |
| 10.12.3.5 m_FOV | 102 |
| 10.12.3.6 m_Initial.mousePosition | 102 |
| 10.12.3.7 m_NearClip | 102 |
| 10.12.3.8 m_Pitch | 102 |
| 10.12.3.9 m_Position | 102 |
| 10.12.3.10 m_ViewMatrix | 102 |
| 10.12.3.11 m_ViewportHeight | 102 |
| 10.12.3.12 m_ViewportWidth | 102 |

| | |
|--|-----|
| 10.12.3.13 m_Yaw | 103 |
| 10.13 Vesper::EditorLayer Class Reference | 103 |
| 10.13.1 Member Enumeration Documentation | 105 |
| 10.13.1.1 SceneState | 105 |
| 10.13.2 Constructor & Destructor Documentation | 105 |
| 10.13.2.1 EditorLayer() | 105 |
| 10.13.2.2 ~EditorLayer() | 106 |
| 10.13.3 Member Function Documentation | 106 |
| 10.13.3.1 NewScene() | 106 |
| 10.13.3.2 OnAttach() | 106 |
| 10.13.3.3 OnDetach() | 108 |
| 10.13.3.4 OnEvent() | 109 |
| 10.13.3.5 OnImGuiRender() | 109 |
| 10.13.3.6 OnKeyPressed() | 112 |
| 10.13.3.7 OnUpdate() | 113 |
| 10.13.3.8 OpenScene() | 116 |
| 10.13.3.9 ResetScene() | 116 |
| 10.13.3.10 SaveSceneAs() | 116 |
| 10.13.4 Member Data Documentation | 117 |
| 10.13.4.1 lastFrameTime | 117 |
| 10.13.4.2 m_ActiveScene | 117 |
| 10.13.4.3 m_BackgroundColor | 117 |
| 10.13.4.4 m_CameraController | 117 |
| 10.13.4.5 m_CameraEntity | 117 |
| 10.13.4.6 m_CheckerboardTexture | 117 |
| 10.13.4.7 m_ClearColor | 117 |
| 10.13.4.8 m_EditorCamera | 117 |
| 10.13.4.9 m_EditorScene | 117 |
| 10.13.4.10 m_FireEntity | 118 |
| 10.13.4.11 m_FlatColorShader | 118 |
| 10.13.4.12 m_Framebuffer | 118 |
| 10.13.4.13 m_GizmoType | 118 |
| 10.13.4.14 m_ParticleProps | 118 |
| 10.13.4.15 m_ParticleSystem | 118 |
| 10.13.4.16 m_PrimaryCamera | 118 |
| 10.13.4.17 m_RotationSnap | 118 |
| 10.13.4.18 m_ScaleSnap | 118 |
| 10.13.4.19 m_SceneHierarchyPanel | 119 |
| 10.13.4.20 m_SceneState | 119 |
| 10.13.4.21 m_SmokeEntity | 119 |
| 10.13.4.22 m_SpecialQuadColor | 119 |
| 10.13.4.23 m_specialQuadRotation | 119 |

| | |
|---|------------|
| 10.13.4.24 m_SpriteSheetCrystals | 119 |
| 10.13.4.25 m_SpriteSheetCursedLands | 119 |
| 10.13.4.26 m_SpriteSheetFire | 119 |
| 10.13.4.27 m_SpriteSheetRocks | 119 |
| 10.13.4.28 m_SpriteSheetSmoke | 119 |
| 10.13.4.29 m_SpriteSheetTown | 120 |
| 10.13.4.30 m_SquareColor | 120 |
| 10.13.4.31 m_squareRotation | 120 |
| 10.13.4.32 m_SquareVA | 120 |
| 10.13.4.33 m_SubTextureFire | 120 |
| 10.13.4.34 m_SubTextureSmoke | 120 |
| 10.13.4.35 m_SubTextureTown | 120 |
| 10.13.4.36 m_textureScale | 120 |
| 10.13.4.37 m_TextureTintColor1 | 120 |
| 10.13.4.38 m_TextureTintColor2 | 120 |
| 10.13.4.39 m_TranslationSnap | 121 |
| 10.13.4.40 m_UseSpecialQuadColor | 121 |
| 10.13.4.41 m_ViewportBounds | 121 |
| 10.13.4.42 m_ViewportFocused | 121 |
| 10.13.4.43 m_ViewportHovered | 121 |
| 10.13.4.44 m_ViewportSize | 121 |
| 10.13.4.45 ParticleEmitCount | 121 |
| 10.13.4.46 s_TextureMap | 121 |
| 10.13.4.47 scene1 | 122 |
| 10.13.4.48 scene2 | 122 |
| 10.13.4.49 scene3 | 122 |
| 10.13.4.50 scene4 | 122 |
| 10.13.4.51 useEntityScene | 122 |
| 10.14 Vesper::Entity Class Reference | 122 |
| 10.14.1 Detailed Description | 123 |
| 10.14.2 Constructor & Destructor Documentation | 123 |
| 10.14.2.1 Entity() [1/3] | 123 |
| 10.14.2.2 Entity() [2/3] | 124 |
| 10.14.2.3 Entity() [3/3] | 124 |
| 10.14.3 Member Function Documentation | 124 |
| 10.14.3.1 AddComponent() | 124 |
| 10.14.3.2 AddOrReplaceComponent() | 125 |
| 10.14.3.3 GetComponent() | 125 |
| 10.14.3.4 GetID() | 125 |
| 10.14.3.5 GetName() | 126 |
| 10.14.3.6 GetOrAddComponent() | 126 |
| 10.14.3.7 HasComponent() | 126 |

| | |
|--|-----|
| 10.14.3.8 operator bool() | 127 |
| 10.14.3.9 operator entt::entity() | 127 |
| 10.14.3.10 operator uint32_t() | 127 |
| 10.14.3.11 operator"!="() | 127 |
| 10.14.3.12 operator==() | 127 |
| 10.14.3.13 RemoveComponent() | 128 |
| 10.14.4 Member Data Documentation | 128 |
| 10.14.4.1 m_EntityID | 128 |
| 10.14.4.2 m_Scene | 128 |
| 10.15 Vesper::Event Class Reference | 128 |
| 10.15.1 Detailed Description | 129 |
| 10.15.2 Constructor & Destructor Documentation | 130 |
| 10.15.2.1 ~Event() | 130 |
| 10.15.3 Member Function Documentation | 130 |
| 10.15.3.1 GetCategoryFlags() | 130 |
| 10.15.3.2 GetEventType() | 130 |
| 10.15.3.3 GetName() | 130 |
| 10.15.3.4 IsInCategory() | 130 |
| 10.15.3.5 ToString() | 131 |
| 10.15.4 Friends And Related Symbol Documentation | 131 |
| 10.15.4.1 EventDispatcher | 131 |
| 10.15.5 Member Data Documentation | 131 |
| 10.15.5.1 Handled | 131 |
| 10.16 Vesper::EventDispatcher Class Reference | 131 |
| 10.16.1 Detailed Description | 132 |
| 10.16.2 Member Typedef Documentation | 132 |
| 10.16.2.1 EventFn | 132 |
| 10.16.3 Constructor & Destructor Documentation | 132 |
| 10.16.3.1 EventDispatcher() | 132 |
| 10.16.4 Member Function Documentation | 133 |
| 10.16.4.1 Dispatch() | 133 |
| 10.16.5 Member Data Documentation | 133 |
| 10.16.5.1 m_Event | 133 |
| 10.17 Vesper::FileDialogs Class Reference | 133 |
| 10.17.1 Detailed Description | 134 |
| 10.17.2 Member Function Documentation | 134 |
| 10.17.2.1 OpenFile() | 134 |
| 10.17.2.2 SaveFile() | 134 |
| 10.18 Vesper::FileSystem Class Reference | 135 |
| 10.18.1 Member Function Documentation | 136 |
| 10.18.1.1 GetAbsolutePath() | 136 |
| 10.18.1.2 GetCurrentWorkingDirectory() | 136 |

| | |
|--|-----|
| 10.18.1.3 GetTravelingUpPath() | 136 |
| 10.18.1.4 Initialize() | 136 |
| 10.18.1.5 IsInitialized() | 137 |
| 10.18.2 Member Data Documentation | 137 |
| 10.18.2.1 m_AssetsDirectory | 137 |
| 10.18.2.2 m_CurrentProjectDirectory | 137 |
| 10.18.2.3 m_Initialized | 137 |
| 10.18.2.4 m_ProjectsDirectory | 137 |
| 10.18.2.5 m_ResourcesDirectory | 137 |
| 10.18.2.6 m_RootEditorDirectory | 137 |
| 10.18.2.7 m_RootEngineDirectory | 138 |
| 10.19 Vesper::Framebuffer Class Reference | 138 |
| 10.19.1 Detailed Description | 138 |
| 10.19.2 Constructor & Destructor Documentation | 139 |
| 10.19.2.1 ~Framebuffer() | 139 |
| 10.19.3 Member Function Documentation | 139 |
| 10.19.3.1 Bind() | 139 |
| 10.19.3.2 Create() | 139 |
| 10.19.3.3 GetColorAttachmentRendererID() | 139 |
| 10.19.3.4 GetSpecification() | 140 |
| 10.19.3.5 Resize() | 140 |
| 10.19.3.6 Unbind() | 140 |
| 10.20 Vesper::GraphicsContext Class Reference | 140 |
| 10.20.1 Detailed Description | 141 |
| 10.20.2 Constructor & Destructor Documentation | 141 |
| 10.20.2.1 ~GraphicsContext() | 141 |
| 10.20.3 Member Function Documentation | 141 |
| 10.20.3.1 Init() | 141 |
| 10.20.3.2 SwapBuffers() | 141 |
| 10.21 Vesper::ImGuiLayer Class Reference | 141 |
| 10.21.1 Constructor & Destructor Documentation | 142 |
| 10.21.1.1 ImGuiLayer() | 142 |
| 10.21.1.2 ~ImGuiLayer() | 143 |
| 10.21.2 Member Function Documentation | 143 |
| 10.21.2.1 Begin() | 143 |
| 10.21.2.2 End() | 143 |
| 10.21.2.3 OnAttach() | 144 |
| 10.21.2.4 OnDetach() | 144 |
| 10.21.2.5 OnEvent() | 145 |
| 10.21.2.6 OnImGuiRender() | 145 |
| 10.21.2.7 SetBlockEvents() | 146 |
| 10.21.2.8 SetDarkThemeColors() | 146 |

| | |
|--|-----|
| 10.21.3 Member Data Documentation | 146 |
| 10.21.3.1 m_BlockEvents | 146 |
| 10.21.3.2 m_Time | 147 |
| 10.22 Vesper::IndexBuffer Class Reference | 147 |
| 10.22.1 Detailed Description | 147 |
| 10.22.2 Constructor & Destructor Documentation | 147 |
| 10.22.2.1 ~IndexBuffer() | 147 |
| 10.22.3 Member Function Documentation | 148 |
| 10.22.3.1 Bind() | 148 |
| 10.22.3.2 Create() | 148 |
| 10.22.3.3 GetCount() | 148 |
| 10.22.3.4 Unbind() | 148 |
| 10.23 Vesper::Input Class Reference | 148 |
| 10.23.1 Detailed Description | 149 |
| 10.23.2 Constructor & Destructor Documentation | 149 |
| 10.23.2.1 Input() [1/2] | 149 |
| 10.23.2.2 Input() [2/2] | 149 |
| 10.23.3 Member Function Documentation | 149 |
| 10.23.3.1 GetMousePosition() | 149 |
| 10.23.3.2 GetMouseX() | 150 |
| 10.23.3.3 GetMouseY() | 150 |
| 10.23.3.4 IsKeyPressed() | 150 |
| 10.23.3.5 IsMouseButtonPressed() | 151 |
| 10.23.3.6 operator=() | 151 |
| 10.24 Vesper::InstrumentationTimer Class Reference | 151 |
| 10.24.1 Constructor & Destructor Documentation | 152 |
| 10.24.1.1 InstrumentationTimer() | 152 |
| 10.24.1.2 ~InstrumentationTimer() | 152 |
| 10.24.2 Member Function Documentation | 152 |
| 10.24.2.1 Stop() | 152 |
| 10.24.3 Member Data Documentation | 152 |
| 10.24.3.1 m_Name | 152 |
| 10.24.3.2 m_StartTimepoint | 152 |
| 10.24.3.3 m_Stopped | 152 |
| 10.25 Vesper::Instrumentor Class Reference | 153 |
| 10.25.1 Constructor & Destructor Documentation | 153 |
| 10.25.1.1 Instrumentor() | 153 |
| 10.25.2 Member Function Documentation | 153 |
| 10.25.2.1 BeginSession() | 153 |
| 10.25.2.2 EndSession() | 154 |
| 10.25.2.3 Get() | 154 |
| 10.25.2.4 InternalEndSession() | 154 |

| | |
|--|-----|
| 10.25.2.5 WriteFooter() | 154 |
| 10.25.2.6 WriteHeader() | 154 |
| 10.25.2.7 WriteProfile() | 154 |
| 10.25.3 Member Data Documentation | 155 |
| 10.25.3.1 m_CurrentSession | 155 |
| 10.25.3.2 m_Mutex | 155 |
| 10.25.3.3 m_OutputStream | 155 |
| 10.26 Vesper::KeyEvent Class Reference | 155 |
| 10.26.1 Detailed Description | 156 |
| 10.26.2 Constructor & Destructor Documentation | 156 |
| 10.26.2.1 KeyEvent() | 156 |
| 10.26.3 Member Function Documentation | 157 |
| 10.26.3.1 GetKeyCode() | 157 |
| 10.26.4 Member Data Documentation | 157 |
| 10.26.4.1 m_KeyCode | 157 |
| 10.27 Vesper::KeyPressedEvent Class Reference | 157 |
| 10.27.1 Detailed Description | 159 |
| 10.27.2 Constructor & Destructor Documentation | 159 |
| 10.27.2.1 KeyPressedEvent() | 159 |
| 10.27.3 Member Function Documentation | 159 |
| 10.27.3.1 GetRepeatCount() | 159 |
| 10.27.3.2 ToString() | 159 |
| 10.27.4 Member Data Documentation | 160 |
| 10.27.4.1 m_RepeatCount | 160 |
| 10.28 Vesper::KeyReleasedEvent Class Reference | 160 |
| 10.28.1 Detailed Description | 161 |
| 10.28.2 Constructor & Destructor Documentation | 161 |
| 10.28.2.1 KeyReleasedEvent() | 161 |
| 10.28.3 Member Function Documentation | 161 |
| 10.28.3.1 ToString() | 161 |
| 10.29 Vesper::KeyTypedEvent Class Reference | 162 |
| 10.29.1 Detailed Description | 163 |
| 10.29.2 Constructor & Destructor Documentation | 163 |
| 10.29.2.1 KeyTypedEvent() | 163 |
| 10.29.3 Member Function Documentation | 163 |
| 10.29.3.1 ToString() | 163 |
| 10.30 Vesper::Layer Class Reference | 164 |
| 10.30.1 Detailed Description | 164 |
| 10.30.2 Constructor & Destructor Documentation | 165 |
| 10.30.2.1 Layer() | 165 |
| 10.30.2.2 ~Layer() | 165 |
| 10.30.3 Member Function Documentation | 165 |

| | |
|--|-----|
| 10.30.3.1 GetName() | 165 |
| 10.30.3.2 OnAttach() | 165 |
| 10.30.3.3 OnDetach() | 166 |
| 10.30.3.4 OnEvent() | 166 |
| 10.30.3.5 OnImGuiRender() | 166 |
| 10.30.3.6 OnRender() | 166 |
| 10.30.3.7 OnUpdate() | 167 |
| 10.30.4 Member Data Documentation | 167 |
| 10.30.4.1 m_DebugName | 167 |
| 10.31 Vesper::LayerStack Class Reference | 167 |
| 10.31.1 Detailed Description | 168 |
| 10.31.2 Constructor & Destructor Documentation | 168 |
| 10.31.2.1 LayerStack() | 168 |
| 10.31.2.2 ~LayerStack() | 168 |
| 10.31.3 Member Function Documentation | 168 |
| 10.31.3.1 begin() | 168 |
| 10.31.3.2 end() | 168 |
| 10.31.3.3 PopLayer() | 169 |
| 10.31.3.4 PopOverlay() | 169 |
| 10.31.3.5 PushLayer() | 169 |
| 10.31.3.6 PushOverlay() | 170 |
| 10.31.3.7 rbegin() | 170 |
| 10.31.3.8 rend() | 170 |
| 10.31.4 Member Data Documentation | 170 |
| 10.31.4.1 m_LayerInsertIndex | 170 |
| 10.31.4.2 m_Layers | 170 |
| 10.32 Vesper::Log Class Reference | 171 |
| 10.32.1 Detailed Description | 171 |
| 10.32.2 Member Function Documentation | 171 |
| 10.32.2.1 GetClientLogger() | 171 |
| 10.32.2.2 GetCoreLogger() | 171 |
| 10.32.2.3 Init() | 172 |
| 10.32.3 Member Data Documentation | 172 |
| 10.32.3.1 s_ClientLogger | 172 |
| 10.32.3.2 s_CoreLogger | 172 |
| 10.33 Vesper::MouseEvent Class Reference | 172 |
| 10.33.1 Detailed Description | 173 |
| 10.33.2 Constructor & Destructor Documentation | 173 |
| 10.33.2.1 MouseEvent() | 173 |
| 10.33.3 Member Function Documentation | 174 |
| 10.33.3.1 GetMouseButton() | 174 |
| 10.33.4 Member Data Documentation | 174 |

| | |
|---|-----|
| 10.33.4.1 <code>m_Button</code> | 174 |
| 10.34 <code>Vesper::MouseButtonPressedEvent</code> Class Reference | 174 |
| 10.34.1 Detailed Description | 175 |
| 10.34.2 Constructor & Destructor Documentation | 176 |
| 10.34.2.1 <code>MouseButtonPressedEvent()</code> | 176 |
| 10.34.3 Member Function Documentation | 176 |
| 10.34.3.1 <code>ToString()</code> | 176 |
| 10.35 <code>Vesper::MouseButtonReleasedEvent</code> Class Reference | 176 |
| 10.35.1 Detailed Description | 177 |
| 10.35.2 Constructor & Destructor Documentation | 178 |
| 10.35.2.1 <code>MouseButtonReleasedEvent()</code> | 178 |
| 10.35.3 Member Function Documentation | 178 |
| 10.35.3.1 <code>ToString()</code> | 178 |
| 10.36 <code>Vesper::MouseMovedEvent</code> Class Reference | 178 |
| 10.36.1 Detailed Description | 179 |
| 10.36.2 Constructor & Destructor Documentation | 180 |
| 10.36.2.1 <code>MouseMovedEvent()</code> | 180 |
| 10.36.3 Member Function Documentation | 180 |
| 10.36.3.1 <code>GetX()</code> | 180 |
| 10.36.3.2 <code>GetY()</code> | 180 |
| 10.36.3.3 <code>ToString()</code> | 180 |
| 10.36.4 Member Data Documentation | 181 |
| 10.36.4.1 <code>m_MouseX</code> | 181 |
| 10.36.4.2 <code>m_MouseY</code> | 181 |
| 10.37 <code>Vesper::MouseScrolledEvent</code> Class Reference | 181 |
| 10.37.1 Detailed Description | 182 |
| 10.37.2 Constructor & Destructor Documentation | 182 |
| 10.37.2.1 <code>MouseScrolledEvent()</code> | 182 |
| 10.37.3 Member Function Documentation | 183 |
| 10.37.3.1 <code>GetXOffset()</code> | 183 |
| 10.37.3.2 <code>GetYOffset()</code> | 183 |
| 10.37.3.3 <code>ToString()</code> | 183 |
| 10.37.4 Member Data Documentation | 183 |
| 10.37.4.1 <code>m_XOffset</code> | 183 |
| 10.37.4.2 <code>m_YOffset</code> | 184 |
| 10.38 <code>Vesper::NameComponent</code> Struct Reference | 184 |
| 10.38.1 Detailed Description | 184 |
| 10.38.2 Constructor & Destructor Documentation | 184 |
| 10.38.2.1 <code>NameComponent()</code> [1/3] | 184 |
| 10.38.2.2 <code>NameComponent()</code> [2/3] | 184 |
| 10.38.2.3 <code>NameComponent()</code> [3/3] | 185 |
| 10.38.3 Member Function Documentation | 185 |

| | |
|--|-----|
| 10.38.3.1 GetName() | 185 |
| 10.38.3.2 operator const std::string &() | 185 |
| 10.38.3.3 operator std::string &() | 185 |
| 10.38.4 Member Data Documentation | 185 |
| 10.38.4.1 Name | 185 |
| 10.39 Vesper::NativeScriptComponent Struct Reference | 185 |
| 10.39.1 Detailed Description | 186 |
| 10.39.2 Member Function Documentation | 186 |
| 10.39.2.1 Bind() | 186 |
| 10.39.3 Member Data Documentation | 186 |
| 10.39.3.1 DestroyScript | 186 |
| 10.39.3.2 Instance | 186 |
| 10.39.3.3 InstantiateScript | 187 |
| 10.40 Vesper::OpenGLContext Class Reference | 187 |
| 10.40.1 Constructor & Destructor Documentation | 187 |
| 10.40.1.1 OpenGLContext() | 187 |
| 10.40.1.2 ~OpenGLContext() | 188 |
| 10.40.2 Member Function Documentation | 188 |
| 10.40.2.1 Init() | 188 |
| 10.40.2.2 SwapBuffers() | 188 |
| 10.40.3 Member Data Documentation | 188 |
| 10.40.3.1 m_WindowHandle | 188 |
| 10.41 Vesper::OpenGLFramebuffer Class Reference | 189 |
| 10.41.1 Constructor & Destructor Documentation | 190 |
| 10.41.1.1 OpenGLFramebuffer() | 190 |
| 10.41.1.2 ~OpenGLFramebuffer() | 190 |
| 10.41.2 Member Function Documentation | 190 |
| 10.41.2.1 Bind() | 190 |
| 10.41.2.2 GetColorAttachmentRendererID() | 190 |
| 10.41.2.3 GetSpecification() | 191 |
| 10.41.2.4 Invalidate() | 191 |
| 10.41.2.5 Resize() | 191 |
| 10.41.2.6 Unbind() | 192 |
| 10.41.3 Member Data Documentation | 192 |
| 10.41.3.1 m_ColorAttachment | 192 |
| 10.41.3.2 m_DepthAttachment | 192 |
| 10.41.3.3 m_RendererID | 192 |
| 10.41.3.4 m_Specification | 192 |
| 10.42 Vesper::OpenGLImGuiLayer Class Reference | 192 |
| 10.42.1 Constructor & Destructor Documentation | 194 |
| 10.42.1.1 OpenGLImGuiLayer() | 194 |
| 10.42.1.2 ~OpenGLImGuiLayer() | 194 |

| | |
|---|-----|
| 10.42.2 Member Function Documentation | 194 |
| 10.42.2.1 Begin() | 194 |
| 10.42.2.2 End() | 194 |
| 10.42.2.3 OnAttach() | 194 |
| 10.42.2.4 OnDetach() | 195 |
| 10.42.2.5 OnEvent() | 195 |
| 10.42.2.6 OnImGuiRender() | 195 |
| 10.42.2.7 SetBlockEvents() | 195 |
| 10.42.2.8 SetDarkThemeColors() | 196 |
| 10.43 Vesper::OpenGLIndexBuffer Class Reference | 196 |
| 10.43.1 Constructor & Destructor Documentation | 197 |
| 10.43.1.1 OpenGLIndexBuffer() | 197 |
| 10.43.1.2 ~OpenGLIndexBuffer() | 197 |
| 10.43.2 Member Function Documentation | 197 |
| 10.43.2.1 Bind() | 197 |
| 10.43.2.2 GetCount() | 197 |
| 10.43.2.3 Unbind() | 198 |
| 10.43.3 Member Data Documentation | 198 |
| 10.43.3.1 m_Count | 198 |
| 10.43.3.2 m_RendererID | 198 |
| 10.44 Vesper::OpenGLRendererAPI Class Reference | 198 |
| 10.44.1 Detailed Description | 199 |
| 10.44.2 Member Function Documentation | 199 |
| 10.44.2.1 Clear() | 199 |
| 10.44.2.2 DrawIndexed() | 200 |
| 10.44.2.3 Init() | 200 |
| 10.44.2.4 SetClearColor() | 200 |
| 10.44.2.5 SetViewport() | 200 |
| 10.45 Vesper::OpenGLShader Class Reference | 201 |
| 10.45.1 Constructor & Destructor Documentation | 202 |
| 10.45.1.1 OpenGLShader() [1/2] | 202 |
| 10.45.1.2 OpenGLShader() [2/2] | 202 |
| 10.45.1.3 ~OpenGLShader() | 202 |
| 10.45.2 Member Function Documentation | 203 |
| 10.45.2.1 Bind() | 203 |
| 10.45.2.2 Compile() | 203 |
| 10.45.2.3 GetName() | 204 |
| 10.45.2.4 PreProcess() | 204 |
| 10.45.2.5 ReadFile() | 205 |
| 10.45.2.6 SetFloat() | 205 |
| 10.45.2.7 SetFloat3() | 205 |
| 10.45.2.8 SetFloat4() | 205 |

| | |
|---|-----|
| 10.45.2.9 SetInt() | 206 |
| 10.45.2.10 SetIntArray() | 206 |
| 10.45.2.11 SetMat4() | 206 |
| 10.45.2.12 Unbind() | 206 |
| 10.45.2.13 UploadUniformFloat() | 207 |
| 10.45.2.14 UploadUniformFloat2() | 207 |
| 10.45.2.15 UploadUniformFloat3() | 207 |
| 10.45.2.16 UploadUniformFloat4() | 207 |
| 10.45.2.17 UploadUniformInt() | 207 |
| 10.45.2.18 UploadUniformMat3() | 208 |
| 10.45.2.19 UploadUniformMat4() | 208 |
| 10.45.3 Member Data Documentation | 208 |
| 10.45.3.1 m_Name | 208 |
| 10.45.3.2 m_RendererID | 208 |
| 10.46 Vesper::OpenGLTexture2D Class Reference | 208 |
| 10.46.1 Constructor & Destructor Documentation | 209 |
| 10.46.1.1 OpenGLTexture2D() [1/2] | 209 |
| 10.46.1.2 OpenGLTexture2D() [2/2] | 210 |
| 10.46.1.3 ~OpenGLTexture2D() | 210 |
| 10.46.2 Member Function Documentation | 210 |
| 10.46.2.1 Bind() | 210 |
| 10.46.2.2 GetHeight() | 211 |
| 10.46.2.3 GetName() | 211 |
| 10.46.2.4 GetRendererID() | 211 |
| 10.46.2.5 GetWidth() | 211 |
| 10.46.2.6 operator==() | 212 |
| 10.46.2.7 SetData() | 212 |
| 10.46.3 Member Data Documentation | 212 |
| 10.46.3.1 m_DataFormat | 212 |
| 10.46.3.2 m_Height | 212 |
| 10.46.3.3 m_InternalFormat | 212 |
| 10.46.3.4 m_Path | 213 |
| 10.46.3.5 m_RendererID | 213 |
| 10.46.3.6 m_Width | 213 |
| 10.47 Vesper::OpenGLUniformBuffer Class Reference | 213 |
| 10.47.1 Constructor & Destructor Documentation | 214 |
| 10.47.1.1 OpenGLUniformBuffer() | 214 |
| 10.47.1.2 ~OpenGLUniformBuffer() | 214 |
| 10.47.2 Member Function Documentation | 214 |
| 10.47.2.1 SetData() | 214 |
| 10.47.3 Member Data Documentation | 215 |
| 10.47.3.1 m_RendererID | 215 |

| | |
|--|-----|
| 10.48 Vesper::OpenGLVertexArray Class Reference | 215 |
| 10.48.1 Constructor & Destructor Documentation | 216 |
| 10.48.1.1 OpenGLVertexArray() | 216 |
| 10.48.1.2 ~OpenGLVertexArray() | 216 |
| 10.48.2 Member Function Documentation | 216 |
| 10.48.2.1 AddVertexBuffer() | 216 |
| 10.48.2.2 Bind() | 217 |
| 10.48.2.3 GetIndexBuffer() | 217 |
| 10.48.2.4 GetVertexBuffers() | 217 |
| 10.48.2.5 SetIndexBuffer() | 217 |
| 10.48.2.6 Unbind() | 217 |
| 10.48.3 Member Data Documentation | 218 |
| 10.48.3.1 m_IndexBuffer | 218 |
| 10.48.3.2 m_RendererID | 218 |
| 10.48.3.3 m_VertexBufferIndex | 218 |
| 10.48.3.4 m_VertexBuffers | 218 |
| 10.49 Vesper::OpenGLVertexBuffer Class Reference | 218 |
| 10.49.1 Constructor & Destructor Documentation | 219 |
| 10.49.1.1 OpenGLVertexBuffer() [1/2] | 219 |
| 10.49.1.2 OpenGLVertexBuffer() [2/2] | 219 |
| 10.49.1.3 ~OpenGLVertexBuffer() | 219 |
| 10.49.2 Member Function Documentation | 220 |
| 10.49.2.1 Bind() | 220 |
| 10.49.2.2 GetLayout() | 220 |
| 10.49.2.3 SetData() | 220 |
| 10.49.2.4 SetLayout() | 220 |
| 10.49.2.5 Unbind() | 220 |
| 10.49.3 Member Data Documentation | 221 |
| 10.49.3.1 m_Layout | 221 |
| 10.49.3.2 m_RendererID | 221 |
| 10.50 Vesper::OrthographicCamera Class Reference | 221 |
| 10.50.1 Constructor & Destructor Documentation | 222 |
| 10.50.1.1 OrthographicCamera() | 222 |
| 10.50.2 Member Function Documentation | 222 |
| 10.50.2.1GetPosition() | 222 |
| 10.50.2.2GetProjectionMatrix() | 222 |
| 10.50.2.3GetRotation() | 222 |
| 10.50.2.4GetViewMatrix() | 222 |
| 10.50.2.5GetViewProjectionMatrix() | 222 |
| 10.50.2.6RecalculateViewMatrix() | 223 |
| 10.50.2.7SetPosition() | 223 |
| 10.50.2.8SetProjection() | 223 |

| | |
|--|-----|
| 10.50.2.9 SetRotation() | 223 |
| 10.50.3 Member Data Documentation | 223 |
| 10.50.3.1 m_Position | 223 |
| 10.50.3.2 m_ProjectionMatrix | 223 |
| 10.50.3.3 m_Rotation | 224 |
| 10.50.3.4 m_ViewMatrix | 224 |
| 10.50.3.5 m_ViewProjectionMatrix | 224 |
| 10.51 Vesper::OrthographicCameraBounds Struct Reference | 224 |
| 10.51.1 Member Function Documentation | 224 |
| 10.51.1.1 GetHeight() | 224 |
| 10.51.1.2 GetWidth() | 225 |
| 10.51.2 Member Data Documentation | 225 |
| 10.51.2.1 Bottom | 225 |
| 10.51.2.2 Left | 225 |
| 10.51.2.3 Right | 225 |
| 10.51.2.4 Top | 225 |
| 10.52 Vesper::OrthographicCameraController Class Reference | 225 |
| 10.52.1 Constructor & Destructor Documentation | 227 |
| 10.52.1.1 OrthographicCameraController() | 227 |
| 10.52.2 Member Function Documentation | 227 |
| 10.52.2.1 CalculateView() | 227 |
| 10.52.2.2 CanRotate() | 227 |
| 10.52.2.3 GetAspectRatio() | 227 |
| 10.52.2.4 GetBounds() | 228 |
| 10.52.2.5 GetCamera() [1/2] | 228 |
| 10.52.2.6 GetCamera() [2/2] | 228 |
| 10.52.2.7 GetMoveSpeed() | 228 |
| 10.52.2.8GetPosition() | 228 |
| 10.52.2.9 GetRotation() | 228 |
| 10.52.2.10 GetRotationSpeed() | 228 |
| 10.52.2.11 GetZoomLevel() | 229 |
| 10.52.2.12 OnEvent() | 229 |
| 10.52.2.13 OnImGuiRender() | 229 |
| 10.52.2.14 OnMouseScrolled() | 230 |
| 10.52.2.15 OnResize() | 230 |
| 10.52.2.16 OnUpdate() | 230 |
| 10.52.2.17 OnUpdateBounds() | 231 |
| 10.52.2.18 OnWindowResized() | 231 |
| 10.52.2.19 SetAspectRatio() | 231 |
| 10.52.2.20 SetCanRotate() | 231 |
| 10.52.2.21 SetMoveSpeed() | 231 |
| 10.52.2.22 SetPosition() | 232 |

| | |
|---|-----|
| 10.52.2.23 SetRotation() | 232 |
| 10.52.2.24 SetRotationSpeed() | 232 |
| 10.52.2.25 SetZoomLevel() | 232 |
| 10.52.2.26 UpdateCameraBounds() | 232 |
| 10.52.3 Member Data Documentation | 233 |
| 10.52.3.1 camera | 233 |
| 10.52.3.2 m_AspectRatio | 233 |
| 10.52.3.3 m_Bounds | 233 |
| 10.52.3.4 m_CameraMoveSpeed | 233 |
| 10.52.3.5 m_CameraPosition | 233 |
| 10.52.3.6 m_CameraRotation | 233 |
| 10.52.3.7 m_CameraRotationSpeed | 233 |
| 10.52.3.8 m_Rotation | 234 |
| 10.52.3.9 m_ZoomLevel | 234 |
| 10.53 Vesper::ParticleSystem Class Reference | 234 |
| 10.53.1 Class Documentation | 235 |
| 10.53.1.1 struct Vesper::ParticleSystem::Particle | 235 |
| 10.53.2 Constructor & Destructor Documentation | 235 |
| 10.53.2.1 ParticleSystem() [1/2] | 235 |
| 10.53.2.2 ParticleSystem() [2/2] | 235 |
| 10.53.3 Member Function Documentation | 235 |
| 10.53.3.1 Emit() | 235 |
| 10.53.3.2 OnRender() | 236 |
| 10.53.3.3 OnUpdate() | 236 |
| 10.53.3.4 SetParticleProps() | 237 |
| 10.53.4 Member Data Documentation | 237 |
| 10.53.4.1 m_ParticlePool | 237 |
| 10.53.4.2 m_PoolIndex | 237 |
| 10.53.4.3 m_Props | 237 |
| 10.54 Vesper::RenderCommand Class Reference | 237 |
| 10.54.1 Detailed Description | 238 |
| 10.54.2 Member Function Documentation | 238 |
| 10.54.2.1 Clear() | 238 |
| 10.54.2.2 DrawIndexed() | 238 |
| 10.54.2.3 Init() | 238 |
| 10.54.2.4 SetClearColor() | 239 |
| 10.54.2.5 SetViewport() | 239 |
| 10.54.3 Member Data Documentation | 239 |
| 10.54.3.1 s_RendererAPI | 239 |
| 10.55 Vesper::Renderer Class Reference | 239 |
| 10.55.1 Detailed Description | 240 |
| 10.55.2 Class Documentation | 240 |

| | |
|---|-----|
| 10.55.2.1 struct Vesper::Renderer::SceneData | 240 |
| 10.55.3 Member Function Documentation | 240 |
| 10.55.3.1 BeginScene() | 240 |
| 10.55.3.2 EndScene() | 241 |
| 10.55.3.3 GetAPI() | 241 |
| 10.55.3.4 Init() | 241 |
| 10.55.3.5 OnWindowResize() | 242 |
| 10.55.3.6 Submit() | 242 |
| 10.55.4 Member Data Documentation | 242 |
| 10.55.4.1 s_SceneData | 242 |
| 10.56 Vesper::Renderer2D Class Reference | 243 |
| 10.56.1 Detailed Description | 244 |
| 10.56.2 Member Function Documentation | 245 |
| 10.56.2.1 BeginScene() [1/3] | 245 |
| 10.56.2.2 BeginScene() [2/3] | 245 |
| 10.56.2.3 BeginScene() [3/3] | 246 |
| 10.56.2.4 DrawQuad() [1/3] | 246 |
| 10.56.2.5 DrawQuad() [2/3] | 247 |
| 10.56.2.6 DrawQuad() [3/3] | 247 |
| 10.56.2.7 DrawQuadRotated() [1/3] | 247 |
| 10.56.2.8 DrawQuadRotated() [2/3] | 248 |
| 10.56.2.9 DrawQuadRotated() [3/3] | 248 |
| 10.56.2.10 DrawQuadRotatedWithTexture() [1/6] | 249 |
| 10.56.2.11 DrawQuadRotatedWithTexture() [2/6] | 250 |
| 10.56.2.12 DrawQuadRotatedWithTexture() [3/6] | 251 |
| 10.56.2.13 DrawQuadRotatedWithTexture() [4/6] | 251 |
| 10.56.2.14 DrawQuadRotatedWithTexture() [5/6] | 252 |
| 10.56.2.15 DrawQuadRotatedWithTexture() [6/6] | 252 |
| 10.56.2.16 DrawQuadWithTexture() [1/6] | 253 |
| 10.56.2.17 DrawQuadWithTexture() [2/6] | 253 |
| 10.56.2.18 DrawQuadWithTexture() [3/6] | 254 |
| 10.56.2.19 DrawQuadWithTexture() [4/6] | 255 |
| 10.56.2.20 DrawQuadWithTexture() [5/6] | 255 |
| 10.56.2.21 DrawQuadWithTexture() [6/6] | 256 |
| 10.56.2.22 EndScene() | 256 |
| 10.56.2.23 Flush() | 257 |
| 10.56.2.24 FlushAndReset() | 257 |
| 10.56.2.25 GetStats() | 257 |
| 10.56.2.26 GetWhiteTexture() | 257 |
| 10.56.2.27 Init() | 258 |
| 10.56.2.28 ResetStats() | 258 |
| 10.56.2.29 Shutdown() | 259 |

| | |
|---|-----|
| 10.56.2.30 StartBatch() | 259 |
| 10.57 Vesper::Renderer2DData Struct Reference | 259 |
| 10.57.1 Class Documentation | 260 |
| 10.57.1.1 struct Vesper::Renderer2DData::CameraData | 260 |
| 10.57.2 Member Data Documentation | 260 |
| 10.57.2.1 CameraBuffer | 260 |
| 10.57.2.2 CameraUniformBuffer | 260 |
| 10.57.2.3 MaxIndices | 260 |
| 10.57.2.4 MaxQuads | 261 |
| 10.57.2.5 MaxTextureSlots | 261 |
| 10.57.2.6 MaxVertices | 261 |
| 10.57.2.7 QuadIndexCount | 261 |
| 10.57.2.8 QuadVertexArray | 261 |
| 10.57.2.9 QuadVertexBuffer | 261 |
| 10.57.2.10 QuadVertexBufferBase | 261 |
| 10.57.2.11 QuadVertexBufferPtr | 262 |
| 10.57.2.12 QuadVertexPositions | 262 |
| 10.57.2.13 Stats | 262 |
| 10.57.2.14 TextureShader | 262 |
| 10.57.2.15 TextureSlotIndex | 262 |
| 10.57.2.16 TextureSlots | 262 |
| 10.57.2.17 WhiteTexture | 262 |
| 10.58 Vesper::RendererAPI Class Reference | 263 |
| 10.58.1 Detailed Description | 263 |
| 10.58.2 Member Enumeration Documentation | 264 |
| 10.58.2.1 API | 264 |
| 10.58.3 Constructor & Destructor Documentation | 264 |
| 10.58.3.1 ~RendererAPI() | 264 |
| 10.58.4 Member Function Documentation | 264 |
| 10.58.4.1 Clear() | 264 |
| 10.58.4.2 DrawIndexed() | 264 |
| 10.58.4.3 GetAPI() | 265 |
| 10.58.4.4 Init() | 265 |
| 10.58.4.5 SetClearColor() | 265 |
| 10.58.4.6 SetViewport() | 265 |
| 10.58.5 Member Data Documentation | 265 |
| 10.58.5.1 s_API | 265 |
| 10.59 Vesper::Scene Class Reference | 266 |
| 10.59.1 Constructor & Destructor Documentation | 267 |
| 10.59.1.1 Scene() [1/2] | 267 |
| 10.59.1.2 Scene() [2/2] | 267 |
| 10.59.1.3 ~Scene() | 267 |

| | |
|--|-----|
| 10.59.2 Member Function Documentation | 267 |
| 10.59.2.1 CreateEntity() [1/2] | 267 |
| 10.59.2.2 CreateEntity() [2/2] | 268 |
| 10.59.2.3 DestroyEntity() | 268 |
| 10.59.2.4 GetName() | 268 |
| 10.59.2.5 GetPrimaryCameraEntity() | 268 |
| 10.59.2.6 OnComponentAdded() [1/17] | 268 |
| 10.59.2.7 OnComponentAdded() [2/17] | 269 |
| 10.59.2.8 OnComponentAdded() [3/17] | 269 |
| 10.59.2.9 OnComponentAdded() [4/17] | 269 |
| 10.59.2.10 OnComponentAdded() [5/17] | 269 |
| 10.59.2.11 OnComponentAdded() [6/17] | 269 |
| 10.59.2.12 OnComponentAdded() [7/17] | 270 |
| 10.59.2.13 OnComponentAdded() [8/17] | 270 |
| 10.59.2.14 OnComponentAdded() [9/17] | 270 |
| 10.59.2.15 OnComponentAdded() [10/17] | 270 |
| 10.59.2.16 OnComponentAdded() [11/17] | 270 |
| 10.59.2.17 OnComponentAdded() [12/17] | 271 |
| 10.59.2.18 OnComponentAdded() [13/17] | 271 |
| 10.59.2.19 OnComponentAdded() [14/17] | 271 |
| 10.59.2.20 OnComponentAdded() [15/17] | 271 |
| 10.59.2.21 OnComponentAdded() [16/17] | 271 |
| 10.59.2.22 OnComponentAdded() [17/17] | 272 |
| 10.59.2.23 OnUpdateEditor() | 272 |
| 10.59.2.24 OnUpdateRuntime() | 272 |
| 10.59.2.25 OnViewportResize() | 274 |
| 10.59.2.26 SetName() | 274 |
| 10.59.3 Friends And Related Symbol Documentation | 274 |
| 10.59.3.1 Entity | 274 |
| 10.59.3.2 SceneHierarchyPanel | 274 |
| 10.59.3.3 SceneSerializer | 274 |
| 10.59.4 Member Data Documentation | 275 |
| 10.59.4.1 m_Name | 275 |
| 10.59.4.2 m_Registry | 275 |
| 10.59.4.3 m_ViewportHeight | 275 |
| 10.59.4.4 m_ViewportWidth | 275 |
| 10.60 Vesper::SceneCamera Class Reference | 275 |
| 10.60.1 Member Enumeration Documentation | 277 |
| 10.60.1.1 ProjectionType | 277 |
| 10.60.2 Constructor & Destructor Documentation | 277 |
| 10.60.2.1 SceneCamera() | 277 |
| 10.60.2.2 ~SceneCamera() | 277 |

| | |
|---|-----|
| 10.60.3 Member Function Documentation | 277 |
| 10.60.3.1 GetOrthographicFarClip() | 277 |
| 10.60.3.2 GetOrthographicNearClip() | 277 |
| 10.60.3.3 GetOrthographicSize() | 277 |
| 10.60.3.4 GetPerspectiveFarClip() | 278 |
| 10.60.3.5 GetPerspectiveNearClip() | 278 |
| 10.60.3.6 GetPerspectiveVerticalFOV() | 278 |
| 10.60.3.7 GetProjectionType() | 278 |
| 10.60.3.8 RecalculateProjection() | 278 |
| 10.60.3.9 SetOrthographic() | 279 |
| 10.60.3.10 SetOrthographicFarClip() | 279 |
| 10.60.3.11 SetOrthographicNearClip() | 279 |
| 10.60.3.12 SetOrthographicSize() | 279 |
| 10.60.3.13 SetPerspective() | 279 |
| 10.60.3.14 SetPerspectiveFarClip() | 280 |
| 10.60.3.15 SetPerspectiveNearClip() | 280 |
| 10.60.3.16 SetPerspectiveVerticalFOV() | 280 |
| 10.60.3.17 SetProjectionType() | 280 |
| 10.60.3.18 SetViewportSize() | 280 |
| 10.60.4 Member Data Documentation | 280 |
| 10.60.4.1 m_AspectRatio | 280 |
| 10.60.4.2 m_OrthographicFar | 281 |
| 10.60.4.3 m_OrthographicNear | 281 |
| 10.60.4.4 m_OrthographicSize | 281 |
| 10.60.4.5 m_PerspectiveFar | 281 |
| 10.60.4.6 m_PerspectiveFOV | 281 |
| 10.60.4.7 m_PerspectiveNear | 281 |
| 10.60.4.8 m_ProjectionType | 281 |
| 10.61 Vesper::SceneHierarchyPanel Class Reference | 282 |
| 10.61.1 Constructor & Destructor Documentation | 282 |
| 10.61.1.1 SceneHierarchyPanel() [1/2] | 282 |
| 10.61.1.2 SceneHierarchyPanel() [2/2] | 282 |
| 10.61.2 Member Function Documentation | 283 |
| 10.61.2.1 DisplayAddComponentEntry() | 283 |
| 10.61.2.2 DrawComponents() | 283 |
| 10.61.2.3 DrawEntityNode() | 285 |
| 10.61.2.4 GetSelectedEntity() | 286 |
| 10.61.2.5 OnImGuiRender() | 286 |
| 10.61.2.6 SetContext() | 287 |
| 10.61.2.7 SetSelectedEntity() | 287 |
| 10.61.3 Member Data Documentation | 287 |
| 10.61.3.1 m_Context | 287 |

| | |
|--|-----|
| 10.61.3.2 m_Framebuffer | 287 |
| 10.61.3.3 m_SelectionContext | 287 |
| 10.62 Vesper::SceneSerializer Class Reference | 287 |
| 10.62.1 Constructor & Destructor Documentation | 288 |
| 10.62.1.1 SceneSerializer() | 288 |
| 10.62.2 Member Function Documentation | 288 |
| 10.62.2.1 Deserialize() | 288 |
| 10.62.2.2 DeserializeRuntime() | 289 |
| 10.62.2.3 Serialize() | 289 |
| 10.62.2.4 SerializeRuntime() | 290 |
| 10.62.3 Member Data Documentation | 290 |
| 10.62.3.1 m_Scene | 290 |
| 10.63 Vesper::ScriptableEntity Class Reference | 290 |
| 10.63.1 Detailed Description | 291 |
| 10.63.2 Constructor & Destructor Documentation | 291 |
| 10.63.2.1 ~ScriptableEntity() | 291 |
| 10.63.3 Member Function Documentation | 291 |
| 10.63.3.1 GetComponent() | 291 |
| 10.63.3.2 OnCreate() | 291 |
| 10.63.3.3 OnDestroy() | 292 |
| 10.63.3.4 OnUpdate() | 292 |
| 10.63.4 Friends And Related Symbol Documentation | 292 |
| 10.63.4.1 Scene | 292 |
| 10.63.5 Member Data Documentation | 292 |
| 10.63.5.1 m_Entity | 292 |
| 10.64 Vesper::Shader Class Reference | 292 |
| 10.64.1 Detailed Description | 293 |
| 10.64.2 Constructor & Destructor Documentation | 293 |
| 10.64.2.1 ~Shader() | 293 |
| 10.64.3 Member Function Documentation | 293 |
| 10.64.3.1 Bind() | 293 |
| 10.64.3.2 Create() [1/2] | 294 |
| 10.64.3.3 Create() [2/2] | 294 |
| 10.64.3.4 GetName() | 294 |
| 10.64.3.5 SetFloat() | 294 |
| 10.64.3.6 SetFloat3() | 294 |
| 10.64.3.7 SetFloat4() | 295 |
| 10.64.3.8 SetInt() | 295 |
| 10.64.3.9 SetIntArray() | 295 |
| 10.64.3.10 SetMat4() | 295 |
| 10.64.3.11 Unbind() | 295 |
| 10.65 Vesper::ShaderLibrary Class Reference | 296 |

| | |
|--|-----|
| 10.65.1 Detailed Description | 296 |
| 10.65.2 Member Function Documentation | 296 |
| 10.65.2.1 Add() [1/2] | 296 |
| 10.65.2.2 Add() [2/2] | 296 |
| 10.65.2.3 Exists() | 297 |
| 10.65.2.4 Get() | 297 |
| 10.65.2.5 Load() [1/2] | 297 |
| 10.65.2.6 Load() [2/2] | 297 |
| 10.65.3 Member Data Documentation | 297 |
| 10.65.3.1 m_Shaders | 297 |
| 10.66 Vesper::SpriteRendererComponent Struct Reference | 298 |
| 10.66.1 Detailed Description | 298 |
| 10.66.2 Constructor & Destructor Documentation | 298 |
| 10.66.2.1 SpriteRendererComponent() [1/3] | 298 |
| 10.66.2.2 SpriteRendererComponent() [2/3] | 298 |
| 10.66.2.3 SpriteRendererComponent() [3/3] | 299 |
| 10.66.3 Member Function Documentation | 299 |
| 10.66.3.1 GetColor() | 299 |
| 10.66.3.2 operator const glm::vec4 &() | 299 |
| 10.66.3.3 operator glm::vec4 &() | 299 |
| 10.66.4 Member Data Documentation | 299 |
| 10.66.4.1 Billboard | 299 |
| 10.66.4.2 Color | 299 |
| 10.66.4.3 Texture | 300 |
| 10.66.4.4 TextureEnabled | 300 |
| 10.66.4.5 TilingFactor | 300 |
| 10.67 Vesper::Renderer2D::Statistics Struct Reference | 300 |
| 10.67.1 Detailed Description | 300 |
| 10.67.2 Member Function Documentation | 301 |
| 10.67.2.1 GetTotalIndexCount() | 301 |
| 10.67.2.2 GetTotalVertexCount() | 301 |
| 10.67.3 Member Data Documentation | 301 |
| 10.67.3.1 DrawCalls | 301 |
| 10.67.3.2 QuadCount | 301 |
| 10.68 Vesper::SubTexture2D Class Reference | 301 |
| 10.68.1 Detailed Description | 302 |
| 10.68.2 Constructor & Destructor Documentation | 302 |
| 10.68.2.1 SubTexture2D() | 302 |
| 10.68.3 Member Function Documentation | 303 |
| 10.68.3.1 CreateFromCoords() | 303 |
| 10.68.3.2 GetTexCoords() | 303 |
| 10.68.3.3 GetTexture() | 303 |

| | |
|--|-----|
| 10.68.4 Member Data Documentation | 303 |
| 10.68.4.1 m_TexCoords | 303 |
| 10.68.4.2 m_Texture | 304 |
| 10.69 Vesper::SubTextureComponent Struct Reference | 304 |
| 10.69.1 Detailed Description | 304 |
| 10.69.2 Constructor & Destructor Documentation | 305 |
| 10.69.2.1 SubTextureComponent() [1/3] | 305 |
| 10.69.2.2 SubTextureComponent() [2/3] | 305 |
| 10.69.2.3 SubTextureComponent() [3/3] | 305 |
| 10.69.3 Member Function Documentation | 305 |
| 10.69.3.1 GetSubTexture() | 305 |
| 10.69.3.2 operator const Ref< SubTexture2D > &() | 305 |
| 10.69.3.3 operator Ref< SubTexture2D > &() | 305 |
| 10.69.3.4 SetOffset() | 305 |
| 10.69.3.5 SetTexture() | 306 |
| 10.69.3.6 SetTilingFactor() | 306 |
| 10.69.4 Member Data Documentation | 306 |
| 10.69.4.1 Offset | 306 |
| 10.69.4.2 SubTexture | 306 |
| 10.69.4.3 TilingFactor | 306 |
| 10.70 Vesper::Texture Class Reference | 307 |
| 10.70.1 Detailed Description | 307 |
| 10.70.2 Constructor & Destructor Documentation | 307 |
| 10.70.2.1 ~Texture() | 307 |
| 10.70.3 Member Function Documentation | 307 |
| 10.70.3.1 Bind() | 307 |
| 10.70.3.2 GetHeight() | 308 |
| 10.70.3.3 GetName() | 308 |
| 10.70.3.4 GetRendererID() | 308 |
| 10.70.3.5 GetWidth() | 308 |
| 10.70.3.6 operator==() | 308 |
| 10.70.3.7 SetData() | 309 |
| 10.71 Vesper::Texture2D Class Reference | 309 |
| 10.71.1 Detailed Description | 310 |
| 10.71.2 Member Function Documentation | 310 |
| 10.71.2.1 Create() [1/2] | 310 |
| 10.71.2.2 Create() [2/2] | 310 |
| 10.72 Vesper::TextureAnimationComponent Struct Reference | 310 |
| 10.72.1 Detailed Description | 311 |
| 10.72.2 Constructor & Destructor Documentation | 311 |
| 10.72.2.1 TextureAnimationComponent() [1/3] | 311 |
| 10.72.2.2 TextureAnimationComponent() [2/3] | 311 |

| | |
|---|-----|
| 10.72.2.3 TextureAnimationComponent() [3/3] | 312 |
| 10.72.3 Member Function Documentation | 312 |
| 10.72.3.1 GetCurrentFrame() | 312 |
| 10.72.3.2 GetSubTextures() | 312 |
| 10.72.3.3 operator const std::vector< Ref< SubTexture2D > &() | 312 |
| 10.72.3.4 operator std::vector< Ref< SubTexture2D > > &() | 312 |
| 10.72.3.5 Update() | 313 |
| 10.72.4 Member Data Documentation | 313 |
| 10.72.4.1 CurrentFrame | 313 |
| 10.72.4.2 FrameTime | 313 |
| 10.72.4.3 SubTextures | 313 |
| 10.72.4.4 TimeAccumulator | 313 |
| 10.73 Vesper::TextureLibrary Class Reference | 314 |
| 10.73.1 Detailed Description | 314 |
| 10.73.2 Member Function Documentation | 314 |
| 10.73.2.1 Add() [1/2] | 314 |
| 10.73.2.2 Add() [2/2] | 315 |
| 10.73.2.3 Exists() | 315 |
| 10.73.2.4 Get() | 315 |
| 10.73.2.5 Load() [1/2] | 315 |
| 10.73.2.6 Load() [2/2] | 315 |
| 10.73.3 Member Data Documentation | 316 |
| 10.73.3.1 m_Textures | 316 |
| 10.74 Vesper::Timestep Class Reference | 316 |
| 10.74.1 Detailed Description | 316 |
| 10.74.2 Constructor & Destructor Documentation | 317 |
| 10.74.2.1 Timestep() | 317 |
| 10.74.3 Member Function Documentation | 317 |
| 10.74.3.1 GetMilliseconds() | 317 |
| 10.74.3.2 GetSeconds() | 317 |
| 10.74.3.3 operator float() | 317 |
| 10.74.4 Member Data Documentation | 318 |
| 10.74.4.1 m_Time | 318 |
| 10.75 Vesper::TransformComponent Struct Reference | 318 |
| 10.75.1 Detailed Description | 318 |
| 10.75.2 Constructor & Destructor Documentation | 318 |
| 10.75.2.1 TransformComponent() [1/3] | 318 |
| 10.75.2.2 TransformComponent() [2/3] | 319 |
| 10.75.2.3 TransformComponent() [3/3] | 319 |
| 10.75.3 Member Function Documentation | 319 |
| 10.75.3.1 GetTransform() | 319 |
| 10.75.4 Member Data Documentation | 319 |

| | |
|--|-----|
| 10.75.4.1 Rotation | 319 |
| 10.75.4.2 Scale | 319 |
| 10.75.4.3 Translation | 319 |
| 10.76 Vesper::UniformBuffer Class Reference | 320 |
| 10.76.1 Detailed Description | 320 |
| 10.76.2 Constructor & Destructor Documentation | 320 |
| 10.76.2.1 ~UniformBuffer() | 320 |
| 10.76.3 Member Function Documentation | 320 |
| 10.76.3.1 Create() | 320 |
| 10.76.3.2 SetData() | 321 |
| 10.77 Vesper::UUID Struct Reference | 321 |
| 10.77.1 Detailed Description | 321 |
| 10.77.2 Constructor & Destructor Documentation | 321 |
| 10.77.2.1 UUID() [1/2] | 321 |
| 10.77.2.2 UUID() [2/2] | 322 |
| 10.77.3 Member Function Documentation | 322 |
| 10.77.3.1 operator const std::string &() | 322 |
| 10.77.3.2 operator std::string &() | 322 |
| 10.77.4 Member Data Documentation | 322 |
| 10.77.4.1 ID | 322 |
| 10.78 Vesper::UUIDComponent Struct Reference | 322 |
| 10.78.1 Detailed Description | 323 |
| 10.78.2 Constructor & Destructor Documentation | 323 |
| 10.78.2.1 UUIDComponent() [1/3] | 323 |
| 10.78.2.2 UUIDComponent() [2/3] | 323 |
| 10.78.2.3 UUIDComponent() [3/3] | 323 |
| 10.78.3 Member Data Documentation | 323 |
| 10.78.3.1 ID | 323 |
| 10.79 Vesper::VertexArray Class Reference | 323 |
| 10.79.1 Detailed Description | 324 |
| 10.79.2 Constructor & Destructor Documentation | 324 |
| 10.79.2.1 ~VertexArray() | 324 |
| 10.79.3 Member Function Documentation | 324 |
| 10.79.3.1 AddVertexBuffer() | 324 |
| 10.79.3.2 Bind() | 324 |
| 10.79.3.3 Create() | 325 |
| 10.79.3.4 GetIndexBuffer() | 325 |
| 10.79.3.5 GetVertexBuffers() | 325 |
| 10.79.3.6 SetIndexBuffer() | 325 |
| 10.79.3.7 Unbind() | 325 |
| 10.80 Vesper::VertexBuffer Class Reference | 326 |
| 10.80.1 Detailed Description | 326 |

| | |
|--|-----|
| 10.80.2 Constructor & Destructor Documentation | 326 |
| 10.80.2.1 ~VertexBuffer() | 326 |
| 10.80.3 Member Function Documentation | 326 |
| 10.80.3.1 Bind() | 326 |
| 10.80.3.2 Create() [1/2] | 327 |
| 10.80.3.3 Create() [2/2] | 327 |
| 10.80.3.4 GetLayout() | 327 |
| 10.80.3.5 SetData() | 327 |
| 10.80.3.6 SetLayout() | 327 |
| 10.80.3.7 Unbind() | 328 |
| 10.81 Vesper::VesperEditor Class Reference | 328 |
| 10.81.1 Constructor & Destructor Documentation | 329 |
| 10.81.1.1 VesperEditor() [1/2] | 329 |
| 10.81.1.2 ~VesperEditor() [1/2] | 329 |
| 10.81.1.3 VesperEditor() [2/2] | 329 |
| 10.81.1.4 ~VesperEditor() [2/2] | 329 |
| 10.82 Vesper::Window Class Reference | 330 |
| 10.82.1 Detailed Description | 330 |
| 10.82.2 Member Typedef Documentation | 331 |
| 10.82.2.1 EventCallbackFn | 331 |
| 10.82.3 Constructor & Destructor Documentation | 331 |
| 10.82.3.1 ~Window() | 331 |
| 10.82.4 Member Function Documentation | 331 |
| 10.82.4.1 Created() | 331 |
| 10.82.4.2 GetHeight() | 331 |
| 10.82.4.3 GetNativeWindow() | 332 |
| 10.82.4.4 GetWidth() | 332 |
| 10.82.4.5 IsVSync() | 332 |
| 10.82.4.6 OnUpdate() | 332 |
| 10.82.4.7 SetEventCallback() | 332 |
| 10.82.4.8 SetVSync() | 333 |
| 10.83 Vesper::WindowCloseEvent Class Reference | 333 |
| 10.83.1 Detailed Description | 334 |
| 10.83.2 Constructor & Destructor Documentation | 334 |
| 10.83.2.1 WindowCloseEvent() | 334 |
| 10.84 Vesper::WindowProps Struct Reference | 334 |
| 10.84.1 Detailed Description | 334 |
| 10.84.2 Constructor & Destructor Documentation | 335 |
| 10.84.2.1 WindowProps() | 335 |
| 10.84.3 Member Data Documentation | 335 |
| 10.84.3.1 Height | 335 |
| 10.84.3.2 Title | 335 |

| | |
|---|------------|
| 10.84.3.3 Width | 335 |
| 10.85 Vesper::WindowResizeEvent Class Reference | 336 |
| 10.85.1 Detailed Description | 337 |
| 10.85.2 Constructor & Destructor Documentation | 337 |
| 10.85.2.1 WindowResizeEvent() | 337 |
| 10.85.3 Member Function Documentation | 337 |
| 10.85.3.1 GetHeight() | 337 |
| 10.85.3.2 GetWidth() | 337 |
| 10.85.3.3 ToString() | 338 |
| 10.85.4 Member Data Documentation | 338 |
| 10.85.4.1 m_Height | 338 |
| 10.85.4.2 m_Width | 338 |
| 10.86 Vesper::WindowsWindow Class Reference | 338 |
| 10.86.1 Class Documentation | 340 |
| 10.86.1.1 struct Vesper::WindowsWindow::WindowData | 340 |
| 10.86.2 Constructor & Destructor Documentation | 340 |
| 10.86.2.1 WindowsWindow() | 340 |
| 10.86.2.2 ~WindowsWindow() | 340 |
| 10.86.3 Member Function Documentation | 340 |
| 10.86.3.1 GetHeight() | 340 |
| 10.86.3.2 GetNativeWindow() | 341 |
| 10.86.3.3 GetWidth() | 341 |
| 10.86.3.4 Init() | 341 |
| 10.86.3.5 IsVSync() | 342 |
| 10.86.3.6 OnUpdate() | 343 |
| 10.86.3.7 SetEventCallback() | 343 |
| 10.86.3.8 SetVSync() | 343 |
| 10.86.3.9 Shutdown() | 343 |
| 10.86.4 Member Data Documentation | 344 |
| 10.86.4.1 m_Context | 344 |
| 10.86.4.2 m_Data | 344 |
| 10.86.4.3 m_Window | 344 |
| 11 File Documentation | 345 |
| 11.1 docs/README.md File Reference | 345 |
| 11.2 Vesper-Editor/src/EditorLayer.cpp File Reference | 345 |
| 11.2.1 Variable Documentation | 345 |
| 11.2.1.1 s_MapHeight | 345 |
| 11.2.1.2 s_MapTiles | 346 |
| 11.2.1.3 s_MapWidth | 346 |
| 11.3 Vesper-Editor/src/EditorLayer.h File Reference | 346 |
| 11.4 EditorLayer.h | 347 |

| | |
|---|-----|
| 11.5 Vesper-Editor/src/Panels/SceneHierarchyPanel.cpp File Reference | 348 |
| 11.6 Vesper-Editor/src/Panels/SceneHierarchyPanel.h File Reference | 348 |
| 11.7 SceneHierarchyPanel.h | 349 |
| 11.8 Vesper-Editor/src/VesperEditorApp.cpp File Reference | 349 |
| 11.9 Vesper/src/Platform/Windows/WindowsInput.cpp File Reference | 350 |
| 11.10 Vesper/src/Platform/Windows/WindowsPlatformUtils.cpp File Reference | 350 |
| 11.10.1 Macro Definition Documentation | 351 |
| 11.10.1.1 GLFW_EXPOSE_NATIVE_WIN32 | 351 |
| 11.11 Vesper/src/Platform/Windows/WindowsWindow.cpp File Reference | 351 |
| 11.12 Vesper/src/Platform/Windows/WindowsWindow.h File Reference | 351 |
| 11.12.1 Class Documentation | 352 |
| 11.12.1.1 struct Vesper::WindowsWindow::WindowData | 352 |
| 11.13 WindowsWindow.h | 352 |
| 11.14 Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.cpp File Reference | 352 |
| 11.15 Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.h File Reference | 353 |
| 11.16 OpenGLBuffer.h | 353 |
| 11.17 Vesper/src/RenderAPI/OpenGL/OpenGLContext.cpp File Reference | 354 |
| 11.18 Vesper/src/RenderAPI/OpenGL/OpenGLContext.h File Reference | 354 |
| 11.19 OpenGLContext.h | 354 |
| 11.20 Vesper/src/RenderAPI/OpenGL/OpenGLFramebuffer.cpp File Reference | 355 |
| 11.21 Vesper/src/RenderAPI/OpenGL/OpenGLFramebuffer.h File Reference | 355 |
| 11.22 OpenGLFramebuffer.h | 355 |
| 11.23 Vesper/src/RenderAPI/OpenGL/OpenGLImGuiLayer.cpp File Reference | 356 |
| 11.24 Vesper/src/RenderAPI/OpenGL/OpenGLImGuiLayer.h File Reference | 356 |
| 11.25 OpenGLImGuiLayer.h | 357 |
| 11.26 Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.cpp File Reference | 357 |
| 11.27 Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.h File Reference | 357 |
| 11.28 OpenGLRendererAPI.h | 358 |
| 11.29 Vesper/src/RenderAPI/OpenGL/OpenGLShader.cpp File Reference | 358 |
| 11.30 Vesper/src/RenderAPI/OpenGL/OpenGLShader.h File Reference | 359 |
| 11.30.1 Typedef Documentation | 359 |
| 11.30.1.1 GLenum | 359 |
| 11.31 OpenGLShader.h | 359 |
| 11.32 Vesper/src/RenderAPI/OpenGL/OpenGLTexture.cpp File Reference | 360 |
| 11.33 Vesper/src/RenderAPI/OpenGL/OpenGLTexture.h File Reference | 360 |
| 11.34 OpenGLTexture.h | 361 |
| 11.35 Vesper/src/RenderAPI/OpenGL/OpenGLUniformBuffer.cpp File Reference | 361 |
| 11.36 Vesper/src/RenderAPI/OpenGL/OpenGLUniformBuffer.h File Reference | 361 |
| 11.37 OpenGLUniformBuffer.h | 362 |
| 11.38 Vesper/src/RenderAPI/OpenGL/OpenGLVertexArray.cpp File Reference | 362 |
| 11.39 Vesper/src/RenderAPI/OpenGL/OpenGLVertexArray.h File Reference | 362 |
| 11.40 OpenGLVertexArray.h | 363 |

| | |
|--|-----|
| 11.41 Vesper/src/Vesper.h File Reference | 363 |
| 11.41.1 Detailed Description | 364 |
| 11.42 Vesper.h | 364 |
| 11.43 Vesper/src/Vesper/App/Application.cpp File Reference | 365 |
| 11.44 Vesper/src/Vesper/App/Application.h File Reference | 365 |
| 11.44.1 Detailed Description | 366 |
| 11.44.2 Class Documentation | 366 |
| 11.44.2.1 struct Vesper::ApplicationSettings | 366 |
| 11.45 Application.h | 367 |
| 11.46 Vesper/src/Vesper/App/EntryPoint.h File Reference | 368 |
| 11.47 EntryPoint.h | 368 |
| 11.48 Vesper/src/Vesper/App/Layer.cpp File Reference | 368 |
| 11.49 Vesper/src/Vesper/App/Layer.h File Reference | 369 |
| 11.49.1 Detailed Description | 369 |
| 11.50 Layer.h | 369 |
| 11.51 Vesper/src/Vesper/App/LayerStack.cpp File Reference | 370 |
| 11.52 Vesper/src/Vesper/App/LayerStack.h File Reference | 370 |
| 11.52.1 Detailed Description | 371 |
| 11.53 LayerStack.h | 371 |
| 11.54 Vesper/src/Vesper/App/Window.h File Reference | 372 |
| 11.54.1 Detailed Description | 372 |
| 11.55 Window.h | 373 |
| 11.56 Vesper/src/Vesper/Core/Asserts.h File Reference | 373 |
| 11.56.1 Detailed Description | 374 |
| 11.56.2 Macro Definition Documentation | 374 |
| 11.56.2.1 VZ_ASSERT | 374 |
| 11.56.2.2 VZ_CORE_ASSERT | 374 |
| 11.57 Asserts.h | 374 |
| 11.58 Vesper/src/Vesper/Core/Base.h File Reference | 375 |
| 11.58.1 Detailed Description | 375 |
| 11.59 Base.h | 375 |
| 11.60 Vesper/src/Vesper/Core/Color.h File Reference | 375 |
| 11.60.1 Detailed Description | 376 |
| 11.61 Color.h | 377 |
| 11.62 Vesper/src/Vesper/Core/Config.h File Reference | 377 |
| 11.62.1 Detailed Description | 378 |
| 11.62.2 Macro Definition Documentation | 378 |
| 11.62.2.1 VZ_DEFAULT_TEXTURE | 378 |
| 11.63 Config.h | 378 |
| 11.64 Vesper/src/Vesper/Core/Defines_Macros.h File Reference | 378 |
| 11.64.1 Detailed Description | 379 |
| 11.64.2 Macro Definition Documentation | 380 |

| | |
|---|-----|
| 11.64.2.1 BIND_EVENT_FN | 380 |
| 11.64.2.2 BIT | 380 |
| 11.64.2.3 EVENT_CLASS_CATEGORY | 381 |
| 11.64.2.4 EVENT_CLASS_TYPE | 381 |
| 11.64.2.5 VZ_BIND_EVENT_FN | 381 |
| 11.65 Defines_Macros.h | 382 |
| 11.66 Vesper/src/Vesper/Core/Log.cpp File Reference | 383 |
| 11.66.1 Detailed Description | 383 |
| 11.67 Vesper/src/Vesper/Core/Log.h File Reference | 383 |
| 11.67.1 Detailed Description | 384 |
| 11.67.2 Macro Definition Documentation | 384 |
| 11.67.2.1 VZ_CORE_ERROR | 384 |
| 11.67.2.2 VZ_CORE_FATAL | 384 |
| 11.67.2.3 VZ_CORE_INFO | 385 |
| 11.67.2.4 VZ_CORE_TRACE | 385 |
| 11.67.2.5 VZ_CORE_WARN | 385 |
| 11.67.2.6 VZ_ERROR | 385 |
| 11.67.2.7 VZ_FATAL | 385 |
| 11.67.2.8 VZ_INFO | 386 |
| 11.67.2.9 VZ_TRACE | 386 |
| 11.67.2.10 VZ_WARN | 386 |
| 11.68 Log.h | 386 |
| 11.69 Vesper/src/Vesper/Core/Math.cpp File Reference | 387 |
| 11.69.1 Macro Definition Documentation | 387 |
| 11.69.1.1 GLM_ENABLE_EXPERIMENTAL | 387 |
| 11.70 Vesper/src/Vesper/Core/Math.h File Reference | 388 |
| 11.70.1 Detailed Description | 388 |
| 11.71 Math.h | 388 |
| 11.72 Vesper/src/Vesper/Core/PlatformDetection.h File Reference | 388 |
| 11.72.1 Detailed Description | 389 |
| 11.73 PlatformDetection.h | 389 |
| 11.74 Vesper/src/Vesper/Core/Random.h File Reference | 389 |
| 11.74.1 Detailed Description | 390 |
| 11.75 Random.h | 390 |
| 11.76 Vesper/src/Vesper/Core/Timer.h File Reference | 392 |
| 11.76.1 Detailed Description | 393 |
| 11.77 Timer.h | 393 |
| 11.78 Vesper/src/Vesper/Core/Timestep.h File Reference | 393 |
| 11.78.1 Detailed Description | 393 |
| 11.79 Timestep.h | 394 |
| 11.80 Vesper/src/Vesper/Debug/Instrumentor.h File Reference | 394 |
| 11.80.1 Detailed Description | 395 |

| | |
|---|-----|
| 11.80.2 Class Documentation | 395 |
| 11.80.2.1 struct Vesper::ProfileResult | 395 |
| 11.80.2.2 struct Vesper::InstrumentationSession | 396 |
| 11.80.2.3 struct InstrumentorUtils::ChangeResult | 396 |
| 11.80.3 Macro Definition Documentation | 396 |
| 11.80.3.1 VZ_FUNC_SIG | 396 |
| 11.80.3.2 VZ_PROFILE | 396 |
| 11.80.3.3 VZ_PROFILE_BEGIN_SESSION | 396 |
| 11.80.3.4 VZ_PROFILE_END_SESSION | 396 |
| 11.80.3.5 VZ_PROFILE_FUNCTION | 397 |
| 11.80.3.6 VZ_PROFILE_SCOPE | 397 |
| 11.80.3.7 VZ_PROFILE_SCOPE_LINE | 397 |
| 11.80.3.8 VZ_PROFILE_SCOPE_LINE2 | 397 |
| 11.81 Instrumentor.h | 397 |
| 11.82 Vesper/src/Vesper/Events/ApplicationEvent.h File Reference | 400 |
| 11.82.1 Detailed Description | 401 |
| 11.83 ApplicationEvent.h | 401 |
| 11.84 Vesper/src/Vesper/Events/Event.h File Reference | 402 |
| 11.84.1 Detailed Description | 403 |
| 11.85 Event.h | 403 |
| 11.86 Vesper/src/Vesper/Events/KeyEvent.h File Reference | 405 |
| 11.86.1 Detailed Description | 405 |
| 11.87 KeyEvent.h | 405 |
| 11.88 Vesper/src/Vesper/Events/MouseEvent.h File Reference | 407 |
| 11.88.1 Detailed Description | 407 |
| 11.89 MouseEvent.h | 408 |
| 11.90 Vesper/src/Vesper/ImGui/ImGuiBuild.cpp File Reference | 409 |
| 11.90.1 Macro Definition Documentation | 410 |
| 11.90.1.1 IMGUI_IMPL_OPENGL_LOADER_GLAD | 410 |
| 11.91 Vesper/src/Vesper/ImGui/ImGuiLayer.cpp File Reference | 410 |
| 11.92 Vesper/src/Vesper/ImGui/ImGuiLayer.h File Reference | 410 |
| 11.93 ImGuiLayer.h | 411 |
| 11.94 Vesper/src/Vesper/ImGui/VesperImGui.h File Reference | 411 |
| 11.95 VesperImGui.h | 411 |
| 11.96 Vesper/src/Vesper/ImGui/ImGuiBuild.cpp File Reference | 412 |
| 11.97 Vesper/src/Vesper/Input/Input.h File Reference | 413 |
| 11.98 Input.h | 413 |
| 11.99 Vesper/src/Vesper/Input/KeyCodes.h File Reference | 413 |
| 11.100 KeyCodes.h | 414 |
| 11.101 Vesper/src/Vesper/Input/ButtonCodes.h File Reference | 416 |
| 11.102 MouseButtonCodes.h | 417 |
| 11.103 Vesper/src/Vesper/ParticleSystem/ParticleSystem.cpp File Reference | 417 |

| | |
|---|-----|
| 11.103.1 Macro Definition Documentation | 418 |
| 11.103.1.1 GLM_ENABLE_EXPERIMENTAL | 418 |
| 11.104 Vesper/src/Vesper/ParticleSystem/ParticleSystem.h File Reference | 418 |
| 11.104.1 Class Documentation | 418 |
| 11.104.1.1 struct Vesper::ParticleProps | 418 |
| 11.104.1.2 struct Vesper::ParticleSystem::Particle | 419 |
| 11.105 ParticleSystem.h | 419 |
| 11.106 Vesper/src/Vesper/Renderer/Buffer.cpp File Reference | 420 |
| 11.107 Vesper/src/Vesper/Renderer/Buffer.h File Reference | 420 |
| 11.108 Buffer.h | 421 |
| 11.109 Vesper/src/Vesper/Renderer/Camera.h File Reference | 423 |
| 11.110 Camera.h | 423 |
| 11.111 Vesper/src/Vesper/Renderer/EditorCamera.cpp File Reference | 423 |
| 11.111.1 Macro Definition Documentation | 424 |
| 11.111.1.1 GLM_ENABLE_EXPERIMENTAL | 424 |
| 11.112 Vesper/src/Vesper/Renderer/EditorCamera.h File Reference | 424 |
| 11.113 EditorCamera.h | 425 |
| 11.114 Vesper/src/Vesper/Renderer/Framebuffer.cpp File Reference | 426 |
| 11.115 Vesper/src/Vesper/Renderer/Framebuffer.h File Reference | 426 |
| 11.115.1 Class Documentation | 426 |
| 11.115.1.1 struct Vesper::FramebufferSpecification | 426 |
| 11.116 Framebuffer.h | 427 |
| 11.117 Vesper/src/Vesper/Renderer/GraphicsContext.h File Reference | 427 |
| 11.118 GraphicsContext.h | 428 |
| 11.119 Vesper/src/Vesper/Renderer/OrthographicCamera.cpp File Reference | 428 |
| 11.120 Vesper/src/Vesper/Renderer/OrthographicCamera.h File Reference | 428 |
| 11.121 OrthographicCamera.h | 429 |
| 11.122 Vesper/src/Vesper/Renderer/OrthographicCameraController.cpp File Reference | 429 |
| 11.123 Vesper/src/Vesper/Renderer/OrthographicCameraController.h File Reference | 429 |
| 11.124 OrthographicCameraController.h | 430 |
| 11.125 Vesper/src/Vesper/Renderer/RenderCommand.cpp File Reference | 431 |
| 11.126 Vesper/src/Vesper/Renderer/RenderCommand.h File Reference | 431 |
| 11.127 RenderCommand.h | 432 |
| 11.128 Vesper/src/Vesper/Renderer/Renderer.cpp File Reference | 432 |
| 11.129 Vesper/src/Vesper/Renderer/Renderer.h File Reference | 433 |
| 11.129.1 Class Documentation | 433 |
| 11.129.1.1 struct Vesper::Renderer::SceneData | 433 |
| 11.130 Renderer.h | 433 |
| 11.131 Vesper/src/Vesper/Renderer/Renderer2D.cpp File Reference | 434 |
| 11.131.1 Class Documentation | 435 |
| 11.131.1.1 struct Vesper::QuadVertex | 435 |
| 11.131.1.2 struct Vesper::Renderer2DData::CameraData | 435 |

| | |
|--|-----|
| 11.132 Vesper/src/Vesper/Renderer/Renderer2D.h File Reference | 435 |
| 11.133 Renderer2D.h | 436 |
| 11.134 Vesper/src/Vesper/Renderer/RendererAPI.cpp File Reference | 439 |
| 11.135 Vesper/src/Vesper/Renderer/RendererAPI.h File Reference | 439 |
| 11.136 RendererAPI.h | 439 |
| 11.137 Vesper/src/Vesper/Renderer/Shader.cpp File Reference | 440 |
| 11.138 Vesper/src/Vesper/Renderer/Shader.h File Reference | 440 |
| 11.139 Shader.h | 441 |
| 11.140 Vesper/src/Vesper/Renderer/SubTexture2D.cpp File Reference | 441 |
| 11.141 Vesper/src/Vesper/Renderer/SubTexture2D.h File Reference | 442 |
| 11.142 SubTexture2D.h | 442 |
| 11.143 Vesper/src/Vesper/Renderer/Texture.cpp File Reference | 443 |
| 11.144 Vesper/src/Vesper/Renderer/Texture.h File Reference | 443 |
| 11.145 Texture.h | 443 |
| 11.146 Vesper/src/Vesper/Renderer/UniformBuffer.cpp File Reference | 444 |
| 11.147 Vesper/src/Vesper/Renderer/UniformBuffer.h File Reference | 444 |
| 11.148 UniformBuffer.h | 445 |
| 11.149 Vesper/src/Vesper/Renderer/VertexArray.cpp File Reference | 445 |
| 11.150 Vesper/src/Vesper/Renderer/VertexArray.h File Reference | 445 |
| 11.151 VertexArray.h | 446 |
| 11.152 Vesper/src/Vesper/Scene/Components.h File Reference | 446 |
| 11.152.1 Macro Definition Documentation | 447 |
| 11.152.1.1 GLM_ENABLE_EXPERIMENTAL | 447 |
| 11.153 Components.h | 448 |
| 11.154 Vesper/src/Vesper/Scene/Entity.cpp File Reference | 451 |
| 11.155 Vesper/src/Vesper/Scene/Entity.h File Reference | 451 |
| 11.156 Entity.h | 451 |
| 11.157 Vesper/src/Vesper/Scene/Scene.cpp File Reference | 453 |
| 11.158 Vesper/src/Vesper/Scene/Scene.h File Reference | 453 |
| 11.159 Scene.h | 454 |
| 11.160 Vesper/src/Vesper/Scene/SceneCamera.cpp File Reference | 454 |
| 11.161 Vesper/src/Vesper/Scene/SceneCamera.h File Reference | 454 |
| 11.162 SceneCamera.h | 455 |
| 11.163 Vesper/src/Vesper/Scene/SceneSerializer.cpp File Reference | 456 |
| 11.164 Vesper/src/Vesper/Scene/SceneSerializer.h File Reference | 456 |
| 11.165 SceneSerializer.h | 457 |
| 11.166 Vesper/src/Vesper/Scene/ScriptableEntity.h File Reference | 457 |
| 11.167 ScriptableEntity.h | 457 |
| 11.168 Vesper/src/Vesper/Utils/PlatformUtils.h File Reference | 458 |
| 11.169 PlatformUtils.h | 458 |
| 11.170 Vesper/src/vzpch.cpp File Reference | 459 |
| 11.171 Vesper/src/vzpch.h File Reference | 459 |

| | |
|---|-----|
| 11.171.1 Detailed Description | 459 |
| 11.172 vzpch.h | 460 |

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
- Platforms: Windows
 - Planned: Linux, macOS
- 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 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();
}
```

Chapter 2

Todo List

File [Application.h](#)

Convert all files/classes in App directory to Vesper::App namespace (create it)

File [Base.h](#)

Move Core classes and helpers in the Core directory to a Vesper::Core namespace.

File [Config.h](#)

: Implement Default Graphics API

File [Layer.h](#)

Add layer priority system (maybe?)

Namespace [Vesper](#)

Temporary

Temporary

Class [Vesper::Application](#)

Update class to accept [ApplicationSettings](#) in constructor and verify its utility/use

Member [Vesper::Application::Run \(\)](#)

Add [Layer](#) rendering into the main loop

Add separate threads for rendering and updating

Member [Vesper::Layer::OnRender \(\)](#)

add layer rendering into the application's render loop

Class [Vesper::LayerStack](#)

Add layer priority system (maybe?)

Class [Vesper::Log](#)

Rethink logging flow with Macros and possibly implement different loggers for different modules.

Member [Vesper::Renderer2D::BeginScene \(const OrthographicCamera &camera\)](#)

Remove once we have a proper scene system

Member [Vesper::Renderer::BeginScene \(OrthographicCamera &camera\)](#)

Support for other camera types.

Member [Vesper::Renderer::Submit \(const Ref< Shader > &shader, const Ref< VertexArray > &vertexArray, const glm::mat4 &transform=glm::mat4\(1.0f\)\)](#)

Set up a command list to encapsulate this for draw order control

Class [Vesper::Window](#)

Add [Window](#) mode functionality

File [Window.h](#)

Add support for different window modes (windowed, fullscreen, borderless).

Chapter 3

Directory Hierarchy

3.1 Directories

| | |
|-------------------------------|-----|
| App | 29 |
| Application.cpp | 365 |
| Application.h | 365 |
| EntryPoint.h | 368 |
| Layer.cpp | 368 |
| Layer.h | 369 |
| LayerStack.cpp | 370 |
| LayerStack.h | 370 |
| Window.h | 372 |
| Core | 29 |
| Asserts.h | 373 |
| Base.h | 375 |
| Color.h | 375 |
| Config.h | 377 |
| Defines_Macros.h | 378 |
| Log.cpp | 383 |
| Log.h | 383 |
| Math.cpp | 387 |
| Math.h | 388 |
| PlatformDetection.h | 388 |
| Random.h | 389 |
| Timer.h | 392 |
| Timestep.h | 393 |
| Debug | 30 |
| Instrumentor.h | 394 |
| docs | 30 |
| Events | 30 |
| ApplicationEvent.h | 400 |
| Event.h | 402 |
| KeyEvent.h | 405 |
| MouseEvent.h | 407 |
| ImGui | 30 |
| ImGuiBuild.cpp | 409 |
| ImGuiLayer.cpp | 410 |
| ImGuiLayer.h | 410 |
| VesperImGui.h | 411 |
| ImGuizmo | 31 |

| | |
|------------------------------------|-----|
| ImGuizmoBuild.cpp | 412 |
| Input | 31 |
| Input.h | 413 |
| KeyCodes.h | 413 |
| MouseButtonCodes.h | 416 |
| OpenGL | 31 |
| OpenGLBuffer.cpp | 352 |
| OpenGLBuffer.h | 353 |
| OpenGLContext.cpp | 354 |
| OpenGLContext.h | 354 |
| OpenGLFramebuffer.cpp | 355 |
| OpenGLFramebuffer.h | 355 |
| OpenGLImGuiLayer.cpp | 356 |
| OpenGLImGuiLayer.h | 356 |
| OpenGLRendererAPI.cpp | 357 |
| OpenGLRendererAPI.h | 357 |
| OpenGLShader.cpp | 358 |
| OpenGLShader.h | 359 |
| OpenGLTexture.cpp | 360 |
| OpenGLTexture.h | 360 |
| OpenGLUniformBuffer.cpp | 361 |
| OpenGLUniformBuffer.h | 361 |
| OpenGLVertexArray.cpp | 362 |
| OpenGLVertexArray.h | 362 |
| Panels | 31 |
| SceneHierarchyPanel.cpp | 348 |
| SceneHierarchyPanel.h | 348 |
| ParticleSystem | 32 |
| ParticleSystem.cpp | 417 |
| ParticleSystem.h | 418 |
| Platform | 32 |
| Windows | 34 |
| WindowsInput.cpp | 350 |
| WindowsPlatformUtils.cpp | 350 |
| WindowsWindow.cpp | 351 |
| WindowsWindow.h | 351 |
| RenderAPI | 32 |
| OpenGL | 31 |
| OpenGLBuffer.cpp | 352 |
| OpenGLBuffer.h | 353 |
| OpenGLContext.cpp | 354 |
| OpenGLContext.h | 354 |
| OpenGLFramebuffer.cpp | 355 |
| OpenGLFramebuffer.h | 355 |
| OpenGLImGuiLayer.cpp | 356 |
| OpenGLImGuiLayer.h | 356 |
| OpenGLRendererAPI.cpp | 357 |
| OpenGLRendererAPI.h | 357 |
| OpenGLShader.cpp | 358 |
| OpenGLShader.h | 359 |
| OpenGLTexture.cpp | 360 |
| OpenGLTexture.h | 360 |
| OpenGLUniformBuffer.cpp | 361 |
| OpenGLUniformBuffer.h | 361 |
| OpenGLVertexArray.cpp | 362 |
| OpenGLVertexArray.h | 362 |

| | |
|--|-----|
| Renderer | 32 |
| Buffer.cpp | 420 |
| Buffer.h | 420 |
| Camera.h | 423 |
| EditorCamera.cpp | 423 |
| EditorCamera.h | 424 |
| Framebuffer.cpp | 426 |
| Framebuffer.h | 426 |
| GraphicsContext.h | 427 |
| OrthographicCamera.cpp | 428 |
| OrthographicCamera.h | 428 |
| OrthographicCameraController.cpp | 429 |
| OrthographicCameraController.h | 429 |
| RenderCommand.cpp | 431 |
| RenderCommand.h | 431 |
| Renderer.cpp | 432 |
| Renderer.h | 433 |
| Renderer2D.cpp | 434 |
| Renderer2D.h | 435 |
| RendererAPI.cpp | 439 |
| RendererAPI.h | 439 |
| Shader.cpp | 440 |
| Shader.h | 440 |
| SubTexture2D.cpp | 441 |
| SubTexture2D.h | 442 |
| Texture.cpp | 443 |
| Texture.h | 443 |
| UniformBuffer.cpp | 444 |
| UniformBuffer.h | 444 |
| VertexArray.cpp | 445 |
| VertexArray.h | 445 |
| Scene | 33 |
| Components.h | 446 |
| Entity.cpp | 451 |
| Entity.h | 451 |
| Scene.cpp | 453 |
| Scene.h | 453 |
| SceneCamera.cpp | 454 |
| SceneCamera.h | 454 |
| SceneSerializer.cpp | 456 |
| SceneSerializer.h | 456 |
| ScriptableEntity.h | 457 |
| src | 33 |
| Panels | 31 |
| SceneHierarchyPanel.cpp | 348 |
| SceneHierarchyPanel.h | 348 |
| EditorLayer.cpp | 345 |
| EditorLayer.h | 346 |
| VesperEditorApp.cpp | 349 |
| src | 33 |
| Platform | 32 |
| Windows | 34 |
| WindowsInput.cpp | 350 |
| WindowsPlatformUtils.cpp | 350 |
| WindowsWindow.cpp | 351 |
| WindowsWindow.h | 351 |
| RenderAPI | 32 |

| | |
|-----------------------------------|-----|
| OpenGL | 31 |
| OpenGLBuffer.cpp | 352 |
| OpenGLBuffer.h | 353 |
| OpenGLContext.cpp | 354 |
| OpenGLContext.h | 354 |
| OpenGLFramebuffer.cpp | 355 |
| OpenGLFramebuffer.h | 355 |
| OpenGLImGuiLayer.cpp | 356 |
| OpenGLImGuiLayer.h | 356 |
| OpenGLRendererAPI.cpp | 357 |
| OpenGLRendererAPI.h | 357 |
| OpenGLShader.cpp | 358 |
| OpenGLShader.h | 359 |
| OpenGLTexture.cpp | 360 |
| OpenGLTexture.h | 360 |
| OpenGLUniformBuffer.cpp | 361 |
| OpenGLUniformBuffer.h | 361 |
| OpenGLVertexArray.cpp | 362 |
| OpenGLVertexArray.h | 362 |
| Vesper | 34 |
| App | 29 |
| Application.cpp | 365 |
| Application.h | 365 |
| EntryPoint.h | 368 |
| Layer.cpp | 368 |
| Layer.h | 369 |
| LayerStack.cpp | 370 |
| LayerStack.h | 370 |
| Window.h | 372 |
| Core | 29 |
| Asserts.h | 373 |
| Base.h | 375 |
| Color.h | 375 |
| Config.h | 377 |
| Defines_Macros.h | 378 |
| Log.cpp | 383 |
| Log.h | 383 |
| Math.cpp | 387 |
| Math.h | 388 |
| PlatformDetection.h | 388 |
| Random.h | 389 |
| Timer.h | 392 |
| Timestep.h | 393 |
| Debug | 30 |
| Instrumentor.h | 394 |
| Events | 30 |
| ApplicationEvent.h | 400 |
| Event.h | 402 |
| KeyEvent.h | 405 |
| MouseEvent.h | 407 |
| ImGui | 30 |
| ImGuiBuild.cpp | 409 |
| ImGuiLayer.cpp | 410 |
| ImGuiLayer.h | 410 |
| VesperImGui.h | 411 |
| ImGuizmo | 31 |
| ImGuizmoBuild.cpp | 412 |
| Input | 31 |

| | |
|--|-----|
| Input.h | 413 |
| KeyCodes.h | 413 |
| MouseButtonCodes.h | 416 |
| ParticleSystem | 32 |
| ParticleSystem.cpp | 417 |
| ParticleSystem.h | 418 |
| Renderer | 32 |
| Buffer.cpp | 420 |
| Buffer.h | 420 |
| Camera.h | 423 |
| EditorCamera.cpp | 423 |
| EditorCamera.h | 424 |
| Framebuffer.cpp | 426 |
| Framebuffer.h | 426 |
| GraphicsContext.h | 427 |
| OrthographicCamera.cpp | 428 |
| OrthographicCamera.h | 428 |
| OrthographicCameraController.cpp | 429 |
| OrthographicCameraController.h | 429 |
| RenderCommand.cpp | 431 |
| RenderCommand.h | 431 |
| Renderer.cpp | 432 |
| Renderer.h | 433 |
| Renderer2D.cpp | 434 |
| Renderer2D.h | 435 |
| RendererAPI.cpp | 439 |
| RendererAPI.h | 439 |
| Shader.cpp | 440 |
| Shader.h | 440 |
| SubTexture2D.cpp | 441 |
| SubTexture2D.h | 442 |
| Texture.cpp | 443 |
| Texture.h | 443 |
| UniformBuffer.cpp | 444 |
| UniformBuffer.h | 444 |
| VertexArray.cpp | 445 |
| VertexArray.h | 445 |
| Scene | 33 |
| Components.h | 446 |
| Entity.cpp | 451 |
| Entity.h | 451 |
| Scene.cpp | 453 |
| Scene.h | 453 |
| SceneCamera.cpp | 454 |
| SceneCamera.h | 454 |
| SceneSerializer.cpp | 456 |
| SceneSerializer.h | 456 |
| ScriptableEntity.h | 457 |
| Utils | 33 |
| PlatformUtils.h | 458 |
| Vesper.h | 363 |
| vzpch.cpp | 459 |
| vzpch.h | 459 |
| Utils | 33 |
| PlatformUtils.h | 458 |
| Vesper | 34 |
| src | 33 |

| | |
|------------------------------------|-----|
| Platform | 32 |
| Windows | 34 |
| WindowsInput.cpp | 350 |
| WindowsPlatformUtils.cpp | 350 |
| WindowsWindow.cpp | 351 |
| WindowsWindow.h | 351 |
| RenderAPI | 32 |
| OpenGL | 31 |
| OpenGLBuffer.cpp | 352 |
| OpenGLBuffer.h | 353 |
| OpenGLContext.cpp | 354 |
| OpenGLContext.h | 354 |
| OpenGLFramebuffer.cpp | 355 |
| OpenGLFramebuffer.h | 355 |
| OpenGLImGuiLayer.cpp | 356 |
| OpenGLImGuiLayer.h | 356 |
| OpenGLRendererAPI.cpp | 357 |
| OpenGLRendererAPI.h | 357 |
| OpenGLShader.cpp | 358 |
| OpenGLShader.h | 359 |
| OpenGLTexture.cpp | 360 |
| OpenGLTexture.h | 360 |
| OpenGLUniformBuffer.cpp | 361 |
| OpenGLUniformBuffer.h | 361 |
| OpenGLVertexArray.cpp | 362 |
| OpenGLVertexArray.h | 362 |
| Vesper | 34 |
| App | 29 |
| Application.cpp | 365 |
| Application.h | 365 |
| EntryPoint.h | 368 |
| Layer.cpp | 368 |
| Layer.h | 369 |
| LayerStack.cpp | 370 |
| LayerStack.h | 370 |
| Window.h | 372 |
| Core | 29 |
| Asserts.h | 373 |
| Base.h | 375 |
| Color.h | 375 |
| Config.h | 377 |
| Defines_Macros.h | 378 |
| Log.cpp | 383 |
| Log.h | 383 |
| Math.cpp | 387 |
| Math.h | 388 |
| PlatformDetection.h | 388 |
| Random.h | 389 |
| Timer.h | 392 |
| Timestep.h | 393 |
| Debug | 30 |
| Instrumentor.h | 394 |
| Events | 30 |
| ApplicationEvent.h | 400 |
| Event.h | 402 |
| KeyEvent.h | 405 |
| MouseEvent.h | 407 |
| ImGui | 30 |

| | |
|--|-----|
| ImGuiBuild.cpp | 409 |
| ImGuiLayer.cpp | 410 |
| ImGuiLayer.h | 410 |
| VesperImGui.h | 411 |
| ImGuiGizmo | 31 |
| ImGuiGizmoBuild.cpp | 412 |
| Input | 31 |
| Input.h | 413 |
| KeyCodes.h | 413 |
| MouseButtonCodes.h | 416 |
| ParticleSystem | 32 |
| ParticleSystem.cpp | 417 |
| ParticleSystem.h | 418 |
| Renderer | 32 |
| Buffer.cpp | 420 |
| Buffer.h | 420 |
| Camera.h | 423 |
| EditorCamera.cpp | 423 |
| EditorCamera.h | 424 |
| Framebuffer.cpp | 426 |
| Framebuffer.h | 426 |
| GraphicsContext.h | 427 |
| OrthographicCamera.cpp | 428 |
| OrthographicCamera.h | 428 |
| OrthographicCameraController.cpp | 429 |
| OrthographicCameraController.h | 429 |
| RenderCommand.cpp | 431 |
| RenderCommand.h | 431 |
| Renderer.cpp | 432 |
| Renderer.h | 433 |
| Renderer2D.cpp | 434 |
| Renderer2D.h | 435 |
| RendererAPI.cpp | 439 |
| RendererAPI.h | 439 |
| Shader.cpp | 440 |
| Shader.h | 440 |
| SubTexture2D.cpp | 441 |
| SubTexture2D.h | 442 |
| Texture.cpp | 443 |
| Texture.h | 443 |
| UniformBuffer.cpp | 444 |
| UniformBuffer.h | 444 |
| VertexArray.cpp | 445 |
| VertexArray.h | 445 |
| Scene | 33 |
| Components.h | 446 |
| Entity.cpp | 451 |
| Entity.h | 451 |
| Scene.cpp | 453 |
| Scene.h | 453 |
| SceneCamera.cpp | 454 |
| SceneCamera.h | 454 |
| SceneSerializer.cpp | 456 |
| SceneSerializer.h | 456 |
| ScriptableEntity.h | 457 |
| Utils | 33 |
| PlatformUtils.h | 458 |
| Vesper.h | 363 |

| | |
|------------------------|-----|
| vzpch.cpp | 459 |
| vzpch.h | 459 |
| Vesper | 34 |
| App | 29 |
| Application.cpp | 365 |
| Application.h | 365 |
| EntryPoint.h | 368 |
| Layer.cpp | 368 |
| Layer.h | 369 |
| LayerStack.cpp | 370 |
| LayerStack.h | 370 |
| Window.h | 372 |
| Core | 29 |
| Asserts.h | 373 |
| Base.h | 375 |
| Color.h | 375 |
| Config.h | 377 |
| Defines_Macros.h | 378 |
| Log.cpp | 383 |
| Log.h | 383 |
| Math.cpp | 387 |
| Math.h | 388 |
| PlatformDetection.h | 388 |
| Random.h | 389 |
| Timer.h | 392 |
| Timestep.h | 393 |
| Debug | 30 |
| Instrumentor.h | 394 |
| Events | 30 |
| ApplicationEvent.h | 400 |
| Event.h | 402 |
| KeyEvent.h | 405 |
| MouseEvent.h | 407 |
| ImGui | 30 |
| ImGuiBuild.cpp | 409 |
| ImGuiLayer.cpp | 410 |
| ImGuiLayer.h | 410 |
| VesperImGui.h | 411 |
| ImGuiZmo | 31 |
| ImGuiZmoBuild.cpp | 412 |
| Input | 31 |
| Input.h | 413 |
| KeyCodes.h | 413 |
| MouseButtonCodes.h | 416 |
| ParticleSystem | 32 |
| ParticleSystem.cpp | 417 |
| ParticleSystem.h | 418 |
| Renderer | 32 |
| Buffer.cpp | 420 |
| Buffer.h | 420 |
| Camera.h | 423 |
| EditorCamera.cpp | 423 |
| EditorCamera.h | 424 |
| Framebuffer.cpp | 426 |
| Framebuffer.h | 426 |
| GraphicsContext.h | 427 |
| OrthographicCamera.cpp | 428 |

| | |
|--|-----|
| OrthographicCamera.h | 428 |
| OrthographicCameraController.cpp | 429 |
| OrthographicCameraController.h | 429 |
| RenderCommand.cpp | 431 |
| RenderCommand.h | 431 |
| Renderer.cpp | 432 |
| Renderer.h | 433 |
| Renderer2D.cpp | 434 |
| Renderer2D.h | 435 |
| RendererAPI.cpp | 439 |
| RendererAPI.h | 439 |
| Shader.cpp | 440 |
| Shader.h | 440 |
| SubTexture2D.cpp | 441 |
| SubTexture2D.h | 442 |
| Texture.cpp | 443 |
| Texture.h | 443 |
| UniformBuffer.cpp | 444 |
| UniformBuffer.h | 444 |
| VertexArray.cpp | 445 |
| VertexArray.h | 445 |
| Scene | 33 |
| Components.h | 446 |
| Entity.cpp | 451 |
| Entity.h | 451 |
| Scene.cpp | 453 |
| Scene.h | 453 |
| SceneCamera.cpp | 454 |
| SceneCamera.h | 454 |
| SceneSerializer.cpp | 456 |
| SceneSerializer.h | 456 |
| ScriptableEntity.h | 457 |
| Utils | 33 |
| PlatformUtils.h | 458 |
| Vesper-Editor | 34 |
| src | 33 |
| Panels | 31 |
| SceneHierarchyPanel.cpp | 348 |
| SceneHierarchyPanel.h | 348 |
| EditorLayer.cpp | 345 |
| EditorLayer.h | 346 |
| VesperEditorApp.cpp | 349 |
| Windows | 34 |
| WindowsInput.cpp | 350 |
| WindowsPlatformUtils.cpp | 350 |
| WindowsWindow.cpp | 351 |
| WindowsWindow.h | 351 |

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

| | |
|--|----|
| InstrumentorUtils | 35 |
| Vesper | |
| TEMPORARY | 36 |
| Vesper::Color | |
| Provides commonly used colors and color manipulation functions | 54 |
| Vesper::Key | |
| Namespace for keyboard key codes | 57 |
| Vesper::Math | 63 |
| Vesper::Mouse | |
| Namespace for mouse button codes | 64 |
| Vesper::Random | 65 |
| YAML | 70 |

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

| | |
|--|-----|
| Vesper::Application | 73 |
| Vesper::VesperEditor | 328 |
| Vesper::VesperEditor | 328 |
| Vesper::ApplicationSettings | 36 |
| Vesper::BufferElement | 83 |
| Vesper::BufferLayout | 85 |
| Vesper::Camera | 88 |
| Vesper::EditorCamera | 94 |
| Vesper::SceneCamera | 275 |
| Vesper::CameraComponent | 89 |
| Vesper::Renderer2DData::CameraData | 259 |
| InstrumentorUtils::ChangeResult< N > | 35 |
| YAML::convert< glm::vec2 > | 91 |
| YAML::convert< glm::vec3 > | 92 |
| YAML::convert< glm::vec4 > | 93 |
| Vesper::Entity | 122 |
| Vesper::Event | 128 |
| Vesper::AppRenderEvent | 80 |
| Vesper::AppTickEvent | 81 |
| Vesper::AppUpdateEvent | 82 |
| Vesper::KeyEvent | 155 |
| Vesper::KeyPressedEvent | 157 |
| Vesper::KeyReleasedEvent | 160 |
| Vesper::KeyTypedEvent | 162 |
| Vesper::MouseButtonEvent | 172 |
| Vesper::MouseButtonPressedEvent | 174 |
| Vesper::MouseButtonReleasedEvent | 176 |
| Vesper::MouseMovedEvent | 178 |
| Vesper::MouseScrolledEvent | 181 |
| Vesper::WindowCloseEvent | 333 |
| Vesper::WindowResizeEvent | 336 |
| Vesper::EventDispatcher | 131 |
| Vesper::FileDialogs | 133 |
| Vesper::FileSystem | 135 |

| | |
|--|-----|
| Vesper::Framebuffer | 138 |
| Vesper::OpenGLFramebuffer | 189 |
| Vesper::FramebufferSpecification | 36 |
| Vesper::GraphicsContext | 140 |
| Vesper::OpenGLContext | 187 |
| Vesper::IndexBuffer | 147 |
| Vesper::OpenGLIndexBuffer | 196 |
| Vesper::Input | 148 |
| Vesper::InstrumentationSession | 36 |
| Vesper::InstrumentationTimer | 151 |
| Vesper::Instrumentor | 153 |
| Vesper::Layer | 164 |
| Vesper::EditorLayer | 103 |
| Vesper::ImGuiLayer | 141 |
| Vesper::OpenGLImGuiLayer | 192 |
| Vesper::LayerStack | 167 |
| Vesper::Log | 171 |
| Vesper::NameComponent | 184 |
| Vesper::NativeScriptComponent | 185 |
| Vesper::OrthographicCamera | 221 |
| Vesper::OrthographicCameraBounds | 224 |
| Vesper::OrthographicCameraController | 225 |
| Vesper::ParticleSystem::Particle | 234 |
| Vesper::ParticleProps | 36 |
| Vesper::ParticleSystem | 234 |
| Vesper::ProfileResult | 36 |
| Vesper::QuadVertex | 36 |
| Vesper::RenderCommand | 237 |
| Vesper::Renderer | 239 |
| Vesper::Renderer2D | 243 |
| Vesper::Renderer2DData | 259 |
| Vesper::RendererAPI | 263 |
| Vesper::OpenGLRendererAPI | 198 |
| Vesper::Scene | 266 |
| Vesper::Renderer::SceneData | 239 |
| Vesper::SceneHierarchyPanel | 282 |
| Vesper::SceneSerializer | 287 |
| Vesper::ScriptableEntity | 290 |
| Vesper::Shader | 292 |
| Vesper::OpenGLShader | 201 |
| Vesper::ShaderLibrary | 296 |
| Vesper::SpriteRendererComponent | 298 |
| Vesper::Renderer2D::Statistics | 300 |
| Vesper::SubTexture2D | 301 |
| Vesper::SubTextureComponent | 304 |
| Vesper::Texture | 307 |
| Vesper::Texture2D | 309 |
| Vesper::OpenGLTexture2D | 208 |
| Vesper::TextureAnimationComponent | 310 |
| Vesper::TextureLibrary | 314 |
| Vesper::Timestep | 316 |
| Vesper::TransformComponent | 318 |
| Vesper::UniformBuffer | 320 |
| Vesper::OpenGLUniformBuffer | 213 |
| Vesper::UUID | 321 |
| Vesper::UUIDComponent | 322 |

| | |
|---|-----|
| Vesper::VertexArray | 323 |
| Vesper::OpenGLVertexArray | 215 |
| Vesper::VertexBuffer | 326 |
| Vesper::OpenGLVertexBuffer | 218 |
| Vesper::Window | 330 |
| Vesper::WindowsWindow | 338 |
| Vesper::WindowsWindow::WindowData | 338 |
| Vesper::WindowProps | 334 |

Chapter 6

Class Index

6.1 Class List

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

| | | |
|--|---|-----|
| Vesper::Application | The core application class that manages the main loop, window, layers, and event handling | 73 |
| Vesper::AppRenderEvent | Event for registering application render | 80 |
| Vesper::AppTickEvent | Event for registering application tick | 81 |
| Vesper::AppUpdateEvent | Event for registering application update | 82 |
| Vesper::BufferElement | Represents a single element in a buffer layout | 83 |
| Vesper::BufferLayout | Represents the layout of a buffer, consisting of multiple BufferElements | 85 |
| Vesper::Camera | | 88 |
| Vesper::CameraComponent | Component that holds camera data | 89 |
| YAML::convert< glm::vec2 > | | 91 |
| YAML::convert< glm::vec3 > | | 92 |
| YAML::convert< glm::vec4 > | | 93 |
| Vesper::EditorCamera | | 94 |
| Vesper::EditorLayer | | 103 |
| Vesper::Entity | Represents an entity in a scene | 122 |
| Vesper::Event | Abstract base class for all events | 128 |
| Vesper::EventDispatcher | Stack-based templated event dispatcher | 131 |
| Vesper::FileDialogs | Cross-platform file dialog utilities | 133 |
| Vesper::FileSystem | | 135 |
| Vesper::Framebuffer | Abstract class representing a framebuffer | 138 |
| Vesper::GraphicsContext | Abstract class representing a graphics context | 140 |
| Vesper::ImGuiLayer | | 141 |
| Vesper::IndexBuffer | Abstract base class for an index buffer | 147 |

| | |
|--|-----|
| Vesper::Input | |
| Base input class for querying input states | 148 |
| Vesper::InstrumentationTimer | |
| | 151 |
| Vesper::Instrumentor | |
| | 153 |
| Vesper::KeyEvent | |
| Base class for keyboard events | 155 |
| Vesper::KeyPressedEvent | |
| Event for registering key press | 157 |
| Vesper::KeyReleasedEvent | |
| Event for registering key release | 160 |
| Vesper::KeyTypedEvent | |
| Event for registering key typing | 162 |
| Vesper::Layer | |
| Represents a reusable application layer that receives lifecycle callbacks (attach, detach, update, events, render, and ImGui render). Intended as a base class for concrete layers | 164 |
| Vesper::LayerStack | |
| Manages an ordered stack of Layer pointers. Layers can be pushed or popped and the stack can be iterated in forward or reverse order | 167 |
| Vesper::Log | |
| A logging utility class for the Vesper engine | 171 |
| Vesper::MouseEvent | |
| Base class for mouse button events | 172 |
| Vesper::MousePressedEvent | |
| Event for registering mouse button presses | 174 |
| Vesper::MouseButtonReleasedEvent | |
| Event for registering mouse button releases | 176 |
| Vesper::MouseMovedEvent | |
| Event for registering mouse movement | 178 |
| Vesper::MouseScrolledEvent | |
| Event for registering mouse scroll wheel movement | 181 |
| Vesper::NameComponent | |
| Component that holds the name of an entity | 184 |
| Vesper::NativeScriptComponent | |
| Component that holds scripting data for an entity | 185 |
| Vesper::OpenGLContext | |
| | 187 |
| Vesper::OpenGLFramebuffer | |
| | 189 |
| Vesper::OpenGLImGuiLayer | |
| | 192 |
| Vesper::OpenGLIndexBuffer | |
| | 196 |
| Vesper::OpenGLRendererAPI | |
| An implementation of the RendererAPI for OpenGL | 198 |
| Vesper::OpenGLShader | |
| | 201 |
| Vesper::OpenGLTexture2D | |
| | 208 |
| Vesper::OpenGLUniformBuffer | |
| | 213 |
| Vesper::OpenGLVertexArray | |
| | 215 |
| Vesper::OpenGLVertexBuffer | |
| | 218 |
| Vesper::OrthographicCamera | |
| | 221 |
| Vesper::OrthographicCameraBounds | |
| | 224 |
| Vesper::OrthographicCameraController | |
| | 225 |
| Vesper::ParticleSystem | |
| | 234 |
| Vesper::RenderCommand | |
| A static class that provides an interface for issuing rendering commands | 237 |
| Vesper::Renderer | |
| The main renderer class responsible for managing rendering operations | 239 |
| Vesper::Renderer2D | |
| A 2D renderer for drawing quads and sprites | 243 |
| Vesper::Renderer2DData | |
| | 259 |
| Vesper::RendererAPI | |
| An abstract class defining the interface for a rendering API | 263 |

| | |
|---|-----|
| Vesper::Scene | 266 |
| Vesper::SceneCamera | 275 |
| Vesper::SceneHierarchyPanel | 282 |
| Vesper::SceneSerializer | 287 |
| Vesper::ScriptableEntity | |
| Base class for scriptable entities within a scene | 290 |
| Vesper::Shader | |
| An abstraction for a shader program | 292 |
| Vesper::ShaderLibrary | |
| A library for managing and storing shaders | 296 |
| Vesper::SpriteRendererComponent | |
| Component that holds sprite rendering data | 298 |
| Vesper::Renderer2D::Statistics | |
| 2D Renderer Statistics | 300 |
| Vesper::SubTexture2D | |
| Represents a sub-region of a 2D texture, useful for sprite sheets | 301 |
| Vesper::SubTextureComponent | |
| Component that holds sub-texture data for sprites | 304 |
| Vesper::Texture | |
| An abstraction for a texture | 307 |
| Vesper::Texture2D | |
| An abstraction for a 2D texture | 309 |
| Vesper::TextureAnimationComponent | |
| Animates through a series of sub textures | 310 |
| Vesper::TextureLibrary | |
| A library for managing and storing textures | 314 |
| Vesper::Timestep | |
| Represents a time step in seconds | 316 |
| Vesper::TransformComponent | |
| Component that holds the transform of an entity | 318 |
| Vesper::UniformBuffer | |
| An abstraction for a uniform buffer object (UBO) | 320 |
| Vesper::UUID | |
| Universally Unique Identifier | 321 |
| Vesper::UUIDComponent | |
| Component that holds a UUID | 322 |
| Vesper::VertexArray | |
| An abstraction for a vertex array object (VAO) | 323 |
| Vesper::VertexBuffer | |
| Abstract base class for a vertex buffer | 326 |
| Vesper::VesperEditor | |
| Vesper::Window | |
| Abstract interface representing an application window | 330 |
| Vesper::WindowCloseEvent | |
| Event for registering window close | 333 |
| Vesper::WindowProps | |
| Holds the data for window configuration | 334 |
| Vesper::WindowResizeEvent | |
| Event for registering window resize | 336 |
| Vesper::WindowsWindow | |

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

| | |
|---|-----|
| Vesper-Editor/src/ EditorLayer.cpp | 345 |
| Vesper-Editor/src/ EditorLayer.h | 346 |
| Vesper-Editor/src/ VesperEditorApp.cpp | 349 |
| Vesper-Editor/src/Panels/ SceneHierarchyPanel.cpp | 348 |
| Vesper-Editor/src/Panels/ SceneHierarchyPanel.h | 348 |
| Vesper/src/ Vesper.h | |
| Main header file for the Vesper engine. For use by clients in Vesper applications | 363 |
| Vesper/src/ vzpch.cpp | 459 |
| Vesper/src/ vzpch.h | |
| Precompiled header for the Vesper engine | 459 |
| Vesper/src/Platform/Windows/ WindowsInput.cpp | 350 |
| Vesper/src/Platform/Windows/ WindowsPlatformUtils.cpp | 350 |
| Vesper/src/Platform/Windows/ WindowsWindow.cpp | 351 |
| Vesper/src/Platform/Windows/ WindowsWindow.h | 351 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLBuffer.cpp | 352 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLBuffer.h | 353 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLContext.cpp | 354 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLContext.h | 354 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLFramebuffer.cpp | 355 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLFramebuffer.h | 355 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLImGuiLayer.cpp | 356 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLImGuiLayer.h | 356 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLRendererAPI.cpp | 357 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLRendererAPI.h | 357 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLShader.cpp | 358 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLShader.h | 359 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLTexture.cpp | 360 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLTexture.h | 360 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLUniformBuffer.cpp | 361 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLUniformBuffer.h | 361 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLVertexArray.cpp | 362 |
| Vesper/src/RenderAPI/OpenGL/ OpenGLVertexArray.h | 362 |
| Vesper/src/Vesper/App/ Application.cpp | 365 |
| Vesper/src/Vesper/App/ Application.h | |
| Controls the run of the Vesper Engine | 365 |

| | |
|--|-----|
| Vesper/src/Vesper/App/ EntryPoint.h | 368 |
| Vesper/src/Vesper/App/ Layer.cpp | 368 |
| Vesper/src/Vesper/App/ Layer.h | |
| Defines the Layer class for creating reusable application layers | 369 |
| Vesper/src/Vesper/App/ LayerStack.cpp | 370 |
| Vesper/src/Vesper/App/ LayerStack.h | |
| Defines the LayerStack class for managing a stack of application layers | 370 |
| Vesper/src/Vesper/App/ Window.h | |
| Defines the abstract Window class and WindowProps struct for window management | 372 |
| Vesper/src/Vesper/Core/ Asserts.h | |
| Provides assertion macros for debugging and error handling | 373 |
| Vesper/src/Vesper/Core/ Base.h | |
| The base header file that includes core definitions and configurations for the Vesper engine | 375 |
| Vesper/src/Vesper/Core/ Color.h | |
| Provides commonly used colors and color manipulation functions | 375 |
| Vesper/src/Vesper/Core/ Config.h | |
| Configuration macros for the Vesper engine and editor | 377 |
| Vesper/src/Vesper/Core/ Defines_Macros.h | |
| Provides commonly used macros along with project wide typedefs | 378 |
| Vesper/src/Vesper/Core/ Log.cpp | |
| Implements the logging system for the Vesper engine | 383 |
| Vesper/src/Vesper/Core/ Log.h | |
| Declares the logging system for the Vesper engine | 383 |
| Vesper/src/Vesper/Core/ Math.cpp | 387 |
| Vesper/src/Vesper/Core/ Math.h | |
| Provides mathematical utility functions | 388 |
| Vesper/src/Vesper/Core/ PlatformDetection.h | |
| Detects the current platform and defines corresponding macros | 388 |
| Vesper/src/Vesper/Core/ Random.h | |
| Provides random number generation utilities | 389 |
| Vesper/src/Vesper/Core/ Timer.h | |
| Declares a Timer class for measuring elapsed time | 392 |
| Vesper/src/Vesper/Core/ Timestep.h | |
| Declares the Timestep class representing a time step in seconds | 393 |
| Vesper/src/Vesper/Debug/ Instrumentor.h | |
| Provides instrumentation and profiling utilities | 394 |
| Vesper/src/Vesper/Events/ ApplicationEvent.h | |
| Defines application event classes | 400 |
| Vesper/src/Vesper/Events/ Event.h | |
| Defines the base Event class and related enumerations and macros | 402 |
| Vesper/src/Vesper/Events/ KeyEvent.h | |
| Defines keyboard event classes | 405 |
| Vesper/src/Vesper/Events/ MouseEvent.h | |
| Defines mouse event classes | 407 |
| Vesper/src/Vesper/ImGui/ ImGuiBuild.cpp | 409 |
| Vesper/src/Vesper/ImGui/ ImGuiLayer.cpp | 410 |
| Vesper/src/Vesper/ImGui/ ImGuiLayer.h | 410 |
| Vesper/src/Vesper/ImGui/ VesperImGui.h | 411 |
| Vesper/src/Vesper/ImGui/ ImGuiBuild.cpp | 412 |
| Vesper/src/Vesper/Input/ Input.h | 413 |
| Vesper/src/Vesper/Input/ KeyCodes.h | 413 |
| Vesper/src/Vesper/Input/ MouseButtonCodes.h | 416 |
| Vesper/src/Vesper/ParticleSystem/ ParticleSystem.cpp | 417 |
| Vesper/src/Vesper/ParticleSystem/ ParticleSystem.h | 418 |
| Vesper/src/Vesper/Renderer/ Buffer.cpp | 420 |
| Vesper/src/Vesper/Renderer/ Buffer.h | 420 |
| Vesper/src/Vesper/Renderer/ Camera.h | 423 |
| Vesper/src/Vesper/Renderer/ EditorCamera.cpp | 423 |

| | |
|--|-----|
| Vesper/src/Vesper/Renderer/ EditorCamera.h | 424 |
| Vesper/src/Vesper/Renderer/ Framebuffer.cpp | 426 |
| Vesper/src/Vesper/Renderer/ Framebuffer.h | 426 |
| Vesper/src/Vesper/Renderer/ GraphicsContext.h | 427 |
| Vesper/src/Vesper/Renderer/ OrthographicCamera.cpp | 428 |
| Vesper/src/Vesper/Renderer/ OrthographicCamera.h | 428 |
| Vesper/src/Vesper/Renderer/ OrthographicCameraController.cpp | 429 |
| Vesper/src/Vesper/Renderer/ OrthographicCameraController.h | 429 |
| Vesper/src/Vesper/Renderer/ RenderCommand.cpp | 431 |
| Vesper/src/Vesper/Renderer/ RenderCommand.h | 431 |
| Vesper/src/Vesper/Renderer/ Renderer.cpp | 432 |
| Vesper/src/Vesper/Renderer/ Renderer.h | 433 |
| Vesper/src/Vesper/Renderer/ Renderer2D.cpp | 434 |
| Vesper/src/Vesper/Renderer/ Renderer2D.h | 435 |
| Vesper/src/Vesper/Renderer/ RendererAPI.cpp | 439 |
| Vesper/src/Vesper/Renderer/ RendererAPI.h | 439 |
| Vesper/src/Vesper/Renderer/ Shader.cpp | 440 |
| Vesper/src/Vesper/Renderer/ Shader.h | 440 |
| Vesper/src/Vesper/Renderer/ SubTexture2D.cpp | 441 |
| Vesper/src/Vesper/Renderer/ SubTexture2D.h | 442 |
| Vesper/src/Vesper/Renderer/ Texture.cpp | 443 |
| Vesper/src/Vesper/Renderer/ Texture.h | 443 |
| Vesper/src/Vesper/Renderer/ UniformBuffer.cpp | 444 |
| Vesper/src/Vesper/Renderer/ UniformBuffer.h | 444 |
| Vesper/src/Vesper/Renderer/ VertexArray.cpp | 445 |
| Vesper/src/Vesper/Renderer/ VertexArray.h | 445 |
| Vesper/src/Vesper/Scene/ Components.h | 446 |
| Vesper/src/Vesper/Scene/ Entity.cpp | 451 |
| Vesper/src/Vesper/Scene/ Entity.h | 451 |
| Vesper/src/Vesper/Scene/ Scene.cpp | 453 |
| Vesper/src/Vesper/Scene/ Scene.h | 453 |
| Vesper/src/Vesper/Scene/ SceneCamera.cpp | 454 |
| Vesper/src/Vesper/Scene/ SceneCamera.h | 454 |
| Vesper/src/Vesper/Scene/ SceneSerializer.cpp | 456 |
| Vesper/src/Vesper/Scene/ SceneSerializer.h | 456 |
| Vesper/src/Vesper/Scene/ ScriptableEntity.h | 457 |
| Vesper/src/Vesper/Utils/ PlatformUtils.h | 458 |

Chapter 8

Directory Documentation

8.1 Vesper/src/Vesper/App Directory Reference

Files

- file [Application.cpp](#)
Controls the run of the `Vesper` Engine.
- file [Application.h](#)
- file [EntryPoint.h](#)
- file [Layer.cpp](#)
Defines the `Layer` class for creating reusable application layers.
- file [Layer.h](#)
- file [LayerStack.cpp](#)
Defines the `LayerStack` class for managing a stack of application layers.
- file [LayerStack.h](#)
- file [Window.h](#)
Defines the abstract `Window` class and `WindowProps` struct for window management.

8.2 Vesper/src/Vesper/Core Directory Reference

Files

- file [Asserts.h](#)
Provides assertion macros for debugging and error handling.
- file [Base.h](#)
The base header file that includes core definitions and configurations for the `Vesper` engine.
- file [Color.h](#)
Provides commonly used colors and color manipulation functions.
- file [Config.h](#)
Configuration macros for the `Vesper` engine and editor.
- file [Defines_Macros.h](#)
Provides commonly used macros along with project wide typedefs.
- file [Log.cpp](#)
Implements the logging system for the `Vesper` engine.

- file [Log.h](#)
Declares the logging system for the [Vesper](#) engine.
- file [Math.cpp](#)
- file [Math.h](#)
Provides mathematical utility functions.
- file [PlatformDetection.h](#)
Detects the current platform and defines corresponding macros.
- file [Random.h](#)
Provides random number generation utilities.
- file [Timer.h](#)
Declares a Timer class for measuring elapsed time.
- file [Timestep.h](#)
Declares the Timestep class representing a time step in seconds.

8.3 Vesper/src/Vesper/Debug Directory Reference

Files

- file [Instrumentor.h](#)
Provides instrumentation and profiling utilities.

8.4 docs Directory Reference

8.5 Vesper/src/Vesper/Events Directory Reference

Files

- file [ApplicationEvent.h](#)
Defines application event classes.
- file [Event.h](#)
Defines the base Event class and related enumerations and macros.
- file [KeyEvent.h](#)
Defines keyboard event classes.
- file [MouseEvent.h](#)
Defines mouse event classes.

8.6 Vesper/src/Vesper/ImGui Directory Reference

Files

- file [ImGuiBuild.cpp](#)
- file [ImGuiLayer.cpp](#)
- file [ImGuiLayer.h](#)
- file [VesperImGui.h](#)

8.7 Vesper/src/Vesper/ImGui Directory Reference

Files

- file [ImGuiBuild.cpp](#)

8.8 Vesper/src/Vesper/Input Directory Reference

Files

- file [Input.h](#)
- file [KeyCodes.h](#)
- file [MouseButtonCodes.h](#)

8.9 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](#)

8.10 Vesper-Editor/src/Panels Directory Reference

Files

- file [SceneHierarchyPanel.cpp](#)
- file [SceneHierarchyPanel.h](#)

8.11 Vesper/src/Vesper/ParticleSystem Directory Reference

Files

- file [ParticleSystem.cpp](#)
- file [ParticleSystem.h](#)

8.12 Vesper/src/Platform Directory Reference

Directories

- directory [Windows](#)

8.13 Vesper/src/RenderAPI Directory Reference

Directories

- directory [OpenGL](#)

8.14 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](#)

8.15 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](#)

8.16 Vesper-Editor/src Directory Reference

Directories

- directory [Panels](#)

Files

- file [EditorLayer.cpp](#)
- file [EditorLayer.h](#)
- file [VesperEditorApp.cpp](#)

8.17 Vesper/src Directory Reference

Directories

- directory [Platform](#)
- directory [RenderAPI](#)
- directory [Vesper](#)

Files

- file [Vesper.h](#)
Main header file for the [Vesper](#) engine. For use by clients in [Vesper](#) applications.
- file [vzpch.cpp](#)
- file [vzpch.h](#)
Precompiled header for the [Vesper](#) engine.

8.18 Vesper/src/Vesper/Utils Directory Reference

Files

- file [PlatformUtils.h](#)

8.19 Vesper Directory Reference

Directories

- directory [src](#)

8.20 Vesper/src/Vesper Directory Reference

Directories

- directory [App](#)
- directory [Core](#)
- directory [Debug](#)
- directory [Events](#)
- directory [ImGui](#)
- directory [ImGuiizmo](#)
- directory [Input](#)
- directory [ParticleSystem](#)
- directory [Renderer](#)
- directory [Scene](#)
- directory [Utils](#)

8.21 Vesper-Editor Directory Reference

Directories

- directory [src](#)

8.22 Vesper/src/Platform/Windows Directory Reference

Files

- file [WindowsInput.cpp](#)
- file [WindowsPlatformUtils.cpp](#)
- file [WindowsWindow.cpp](#)
- file [WindowsWindow.h](#)

Chapter 9

Namespace Documentation

9.1 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])

9.1.1 Class Documentation

9.1.1.1 struct InstrumentorUtils::ChangeResult

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

Class Members

| | | |
|------|--------------------------|--|
| char | Data \leftarrow [N] | |
|------|--------------------------|--|

9.1.2 Function Documentation

9.1.2.1 CleanupOutputString()

```

template<size_t N, size_t K>
auto InstrumentorUtils::CleanupOutputString (
    const char(&) expr[N],
    const char(&) remove[K]) [constexpr]
00191 {
00192     ChangeResult<N> result = {};
00193
00194     size_t srcIndex = 0;
00195     size_t dstIndex = 0;
00196     while (srcIndex < N)
00197     {
00198         size_t matchIndex = 0;
00199         while (matchIndex < K - 1 && srcIndex + matchIndex < N - 1 && expr[srcIndex + matchIndex]
00200             == remove[matchIndex])
00201             matchIndex++;
00202         if (matchIndex == K - 1)
00203             srcIndex += matchIndex;
00204         result.Data[dstIndex++] = expr[srcIndex] == '"' ? '\"' : expr[srcIndex];
00205         srcIndex++;
00206     }
00207     return result;
}

```

9.2 Vesper Namespace Reference

TEMPORARY.

Namespaces

- namespace [Math](#)
- namespace [Key](#)

Namespace for keyboard key codes.
- namespace [Random](#)
- namespace [Color](#)

Provides commonly used colors and color manipulation functions.
- namespace [Mouse](#)

Namespace for mouse button codes.

Classes

- class [Input](#)

Base input class for querying input states.
- struct [ApplicationSettings](#)

WIP. More...
- class [Application](#)

The core application class that manages the main loop, window, layers, and event handling.
- class [FileDialogs](#)

Cross-platform file dialog utilities.
- class [FileSystem](#)
- class [WindowsWindow](#)
- class [WindowResizeEvent](#)

- class [WindowCloseEvent](#)
Event for registering window close.
- class [AppTickEvent](#)
Event for registering application tick.
- class [AppUpdateEvent](#)
Event for registering application update.
- class [AppRenderEvent](#)
Event for registering application render.
- class [MouseMovedEvent](#)
Event for registering mouse movement.
- class [MouseScrolledEvent](#)
Event for registering mouse scroll wheel movement.
- class [MouseButtonEvent](#)
Base class for mouse button events.
- class [MouseButtonPressedEvent](#)
Event for registering mouse button presses.
- class [MouseButtonReleasedEvent](#)
Event for registering mouse button releases.
- class [KeyEvent](#)
Base class for keyboard events.
- class [KeyPressedEvent](#)
Event for registering key press.
- class [KeyReleasedEvent](#)
Event for registering key release.
- class [KeyTypedEvent](#)
Event for registering key typing.
- class [OpenGLContext](#)
- class [GraphicsContext](#)
Abstract class representing a graphics context.
- class [OpenGLFramebuffer](#)
- class [OpenGLImGuiLayer](#)
- class [RendererAPI](#)
An abstract class defining the interface for a rendering API.
- class [OpenGLRendererAPI](#)
An implementation of the [RendererAPI](#) for OpenGL.
- class [OpenGLShader](#)
- class [OpenGLTexture2D](#)
- class [OpenGLUniformBuffer](#)
- class [OpenGLVertexArray](#)
- class [Layer](#)
Represents a reusable application layer that receives lifecycle callbacks (`attach`, `detach`, `update`, `events`, `render`, and `ImGui render`). Intended as a base class for concrete layers.
- class [LayerStack](#)
Manages an ordered stack of [Layer](#) pointers. Layers can be pushed or popped and the stack can be iterated in forward or reverse order.
- class [Log](#)
A logging utility class for the [Vesper](#) engine.
- class [ImGuiLayer](#)
- struct [ParticleProps](#)
- class [ParticleSystem](#)
- class [Renderer2D](#)

- class [OrthographicCamera](#)
 - struct [BufferElement](#)

Represents a single element in a buffer layout.
 - class [BufferLayout](#)

Represents the layout of a buffer, consisting of multiple BufferElements.
 - class [VertexBuffer](#)

Abstract base class for a vertex buffer.
 - class [IndexBuffer](#)

Abstract base class for an index buffer.
 - class [OpenGLVertexBuffer](#)
 - class [OpenGLIndexBuffer](#)
 - class [EditorCamera](#)
 - struct [FramebufferSpecification](#)

Specification for creating a Framebuffer. More...
 - class [Framebuffer](#)

Abstract class representing a framebuffer.
 - class [Renderer](#)

The main renderer class responsible for managing rendering operations.
 - struct [ProfileResult](#)
 - struct [InstrumentationSession](#)
 - class [Instrumentor](#)
 - class [InstrumentationTimer](#)
 - struct [OrthographicCameraBounds](#)
 - class [OrthographicCameraController](#)
 - class [RenderCommand](#)

A static class that provides an interface for issuing rendering commands.
 - class [Shader](#)

An abstraction for a shader program.
 - class [ShaderLibrary](#)

A library for managing and storing shaders.
 - class [Texture](#)

An abstraction for a texture.
 - class [Texture2D](#)

An abstraction for a 2D texture.
 - class [TextureLibrary](#)

A library for managing and storing textures.
 - class [SubTexture2D](#)

Represents a sub-region of a 2D texture, useful for sprite sheets.
 - class [Camera](#)
 - class [Timestep](#)

Represents a time step in seconds.
 - class [Event](#)

Abstract base class for all events.
 - class [EventDispatcher](#)

Stack-based templated event dispatcher.
 - struct [QuadVertex](#)
 - struct [Renderer2DData](#)
 - class [UniformBuffer](#)

An abstraction for a uniform buffer object (UBO).
 - class [VertexArray](#)

An abstraction for a vertex array object (VAO).

- class [Entity](#)
Represents an entity in a scene.
- class [Scene](#)
- class [ScriptableEntity](#)
Base class for scriptable entities within a scene.
- class [SceneCamera](#)
- class [EditorLayer](#)
- class [SceneSerializer](#)
- class [SceneHierarchyPanel](#)
- struct [UUID](#)
Universally Unique Identifier.
- struct [UUIDComponent](#)
Component that holds a [UUID](#).
- struct [NameComponent](#)
Component that holds the name of an entity.
- struct [TransformComponent](#)
Component that holds the transform of an entity.
- struct [SpriteRendererComponent](#)
Component that holds sprite rendering data.
- struct [SubTextureComponent](#)
Component that holds sub-texture data for sprites.
- struct [TextureAnimationComponent](#)
Animates through a series of sub textures.
- struct [CameraComponent](#)
Component that holds camera data.
- struct [NativeScriptComponent](#)
Component that holds scripting data for an entity.
- class [VesperEditor](#)
- struct [WindowProps](#)
Holds the data for window configuration.
- class [Window](#)
Abstract interface representing an application window.

Typedefs

- using [FloatingPointMicroseconds](#) = std::chrono::duration<double, std::micro>
- using [KeyCode](#) = uint16_t
Alias for keyboard key code type.
- using [MouseCode](#) = uint8_t
Alias for mouse button code type.
- template<typename T>
using [Scope](#) = std::unique_ptr<T>
A smart pointer type representing exclusive ownership of an object.
- template<typename T>
using [Ref](#) = std::shared_ptr<T>
A smart pointer type representing shared ownership of an object.

Enumerations

- enum class `ShaderDataType` {
 `None` = 0 , `Float` , `Float2` , `Float3` ,
 `Float4` , `Mat3` , `Mat4` , `Int` ,
 `Int2` , `Int3` , `Int4` , `Bool` }

The different data types that can be used in shaders.
- enum class `EventType` {
 `None` = 0 , `WindowClose` , `WindowResize` , `WindowFocus` ,
 `WindowLostFocus` , `WindowMoved` , `AppTick` , `AppUpdate` ,
 `AppRender` , `KeyPressed` , `KeyReleased` , `KeyTyped` ,
 `MouseButtonPressed` , `MouseButtonReleased` , `MouseMoved` , `MouseScrolled` }

Enumeration of event types.
- enum `EventCategory` {
 `None` = 0 , `EventCategoryApplication` = `BIT(0)` , `EventCategoryInput` = `BIT(1)` , `EventCategoryKeyboard` = `BIT(2)` ,
 `EventCategoryMouse` = `BIT(3)` , `EventCategoryMouseButton` = `BIT(4)` }

Enumeration of event categories.
- enum class `WindowMode` { `Windowed` = 0 , `Fullscreen` = 1 , `Borderless` = 2 }

WIP.

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)

Returns the size in bytes of the given `ShaderDataType`.
- std::string `format_as` (const `Event` &e)

Format an event as a string.
- 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)
- template<typename T, typename... Args>
 `constexpr Scope< T > CreateScope` (Args &&... args)

Creates a `Scope` (unique_ptr) for the given type and constructor arguments.
- template<typename T, typename... Args>
 `constexpr Ref< T > CreateRef` (Args &&... args)

Creates a `Ref` (shared_ptr) for the given type and constructor arguments.

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`

9.2.1 Detailed Description

TEMPORARY.

The main namespace for the [Vesper](#) engine.

Temporary.

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

Todo Temporary

Todo Temporary

9.2.2 Class Documentation

9.2.2.1 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 | |

9.2.2.2 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 } | | |

9.2.2.3 struct Vesper::FramebufferSpecification

Specification for creating a [Framebuffer](#).

Class Members

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

9.2.2.4 struct Vesper::ProfileResult

Class Members

| | | | |
|-----------|----------|--|--|
| long long | End | | |
| string | Name | | |
| long long | Start | | |
| uint32_t | ThreadID | | |

9.2.2.5 struct Vesper::InstrumentationSession

Class Members

| | | |
|--------|------|--|
| string | Name | |
|--------|------|--|

9.2.2.6 struct Vesper::QuadVertex

Class Members

| | | | |
|-------|--------------|--|--|
| vec4 | Color | | |
| vec3 | Position | | |
| vec2 | TexCoord | | |
| float | TexIndex | | |
| float | TilingFactor | | |

9.2.3 Typedef Documentation

9.2.3.1 FloatingPointMicroseconds

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

9.2.3.2 KeyCode

```
using Vesper::KeyCode = uint16_t
```

Alias for keyboard key code type.

9.2.3.3 MouseCode

```
using Vesper::MouseCode = uint8_t
```

Alias for mouse button code type.

9.2.3.4 Ref

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

A smart pointer type representing shared ownership of an object.

Template Parameters

| | |
|----------|-------------------------------|
| <i>T</i> | The type of object to manage. |
|----------|-------------------------------|

Note

This is an alias for std::shared_ptr.

9.2.3.5 Scope

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

A smart pointer type representing exclusive ownership of an object.

Template Parameters

| | |
|----------|-------------------------------|
| <i>T</i> | The type of object to manage. |
|----------|-------------------------------|

Note

This is an alias for std::unique_ptr.

9.2.4 Enumeration Type Documentation

9.2.4.1 EventCategory

```
enum Vesper::EventCategory
```

Enumeration of event categories.

Enumerator

| | |
|--------------------------|--|
| None | |
| EventCategoryApplication | |
| EventCategoryInput | |
| EventCategoryKeyboard | |
| EventCategoryMouse | |
| EventCategoryMouseButton | |

```
00024  {
00025      None = 0,
00026      EventCategoryApplication = BIT(0),
00027      EventCategoryInput = BIT(1),
00028      EventCategoryKeyboard = BIT(2),
00029      EventCategoryMouse = BIT(3),
00030      EventCategoryMouseButton = BIT(4)
00031  };
```

9.2.4.2 EventType

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

Enumeration of event types.

Enumerator

| | |
|---------------------|--|
| None | |
| WindowClose | |
| WindowResize | |
| WindowFocus | |
| WindowLostFocus | |
| WindowMoved | |
| AppTick | |
| AppUpdate | |
| AppRender | |
| KeyPressed | |
| KeyReleased | |
| KeyTyped | |
| MouseButtonPressed | |
| MouseButtonReleased | |
| MouseMove | |

| | |
|---------------|--|
| MouseScrolled | |
|---------------|--|

```
00014      {
00015          None = 0,
00016          WindowClose, WindowResize, WindowFocus, WindowLostFocus, WindowMoved,
00017          AppTick, AppUpdate, AppRender,
00018          KeyPressed, KeyReleased, KeyTyped,
00019          MouseButtonPressed, MouseButtonReleased, MouseMoved, MouseScrolled
00020      };
```

9.2.4.3 ShaderDataType

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

The different data types that can be used in shaders.

Enumerator

| | |
|--------|--|
| None | |
| Float | |
| Float2 | |
| Float3 | |
| Float4 | |
| Mat3 | |
| Mat4 | |
| Int | |
| Int2 | |
| Int3 | |
| Int4 | |
| Bool | |

```
00006          {
00007              None = 0,
00008              Float, Float2, Float3, Float4,
00009              Mat3, Mat4,
00010              Int, Int2, Int3, Int4,
00011              Bool
00012      };
```

9.2.4.4 WindowMode

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

WIP.

Enumerator

| | |
|------------|--|
| Windowed | |
| Fullscreen | |
| Borderless | |

```
00016      {
00017          Windowed = 0,
00018          Fullscreen = 1,
00019          Borderless = 2
00020      };
```

9.2.5 Function Documentation

9.2.5.1 CreateApplication()

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

References [Vesper::VesperEditor::VesperEditor\(\)](#).

9.2.5.2 CreateRef()

```
template<typename T, typename... Args>
Ref< T > Vesper::CreateRef (
    Args &&... args) [constexpr]
```

Creates a [Ref \(shared_ptr\)](#) for the given type and constructor arguments.

Template Parameters

| | |
|-------------|-------------------------------------|
| <i>T</i> | The type of object to create. |
| <i>Args</i> | The types of constructor arguments. |

Parameters

| | |
|-------------|----------------------------|
| <i>args</i> | The constructor arguments. |
|-------------|----------------------------|

Returns

A [Ref \(shared_ptr\)](#) managing the created object.

```
00073     {
00074         return std::make_shared<T>(std::forward<Args>(args)...);
00075     }
```

9.2.5.3 CreateScope()

```
template<typename T, typename... Args>
Scope< T > Vesper::CreateScope (
    Args &&... args) [constexpr]
```

Creates a [Scope \(unique_ptr\)](#) for the given type and constructor arguments.

Template Parameters

| | |
|-------------|-------------------------------------|
| <i>T</i> | The type of object to create. |
| <i>Args</i> | The types of constructor arguments. |

Parameters

| | |
|-------------------|----------------------------|
| <code>args</code> | The constructor arguments. |
|-------------------|----------------------------|

Returns

A [Scope](#) (unique_ptr) managing the created object.

```
00053     {
00054         return std::make_unique<T>(std::forward<Args>(args)...);
00055     }
```

9.2.5.4 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\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\(\)](#).

9.2.5.5 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 }
```

9.2.5.6 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     }

```

9.2.5.7 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     ImGui::PushStyleVar(ImGuiStyleVar_ItemSpacing, ImVec2{ 0, 0 });
00167
00168     float lineHeight = ImGui::GetFontSize() + ImGui::GetStyle().FramePadding.y * 2.0f;
00169     ImVec2 buttonSize = { lineHeight + 3.0f, lineHeight };
00170
00171     // Compute available width for the three float controls in the right column
00172     float availableWidth = ImGui::GetContentRegionAvail().x;
00173     float itemSpacing = ImGui::GetStyle().ItemSpacing.x;
00174     float totalButtonWidth = buttonSize.x * 3.0f;
00175     // Account for SameLine() spacings between button+control pairs (conservative estimate)
00176     float totalSpacing = itemSpacing * 6.0f;
00177     float itemWidth = (availableWidth - totalButtonWidth - totalSpacing) / 3.0f;
00178     if (itemWidth <= 0.0f)
00179         itemWidth = ImGui::CalcItemWidth();
00180
00181     ImGui::PushStyleColor(ImGuiCol_Button, ImVec4{ 0.8f, 0.1f, 0.15f, 1.0f });
00182     ImGui::PushStyleColor(ImGuiCol_ButtonHovered, ImVec4{ 0.9f, 0.2f, 0.2f, 1.0f });
00183     ImGui::PushStyleColor(ImGuiCol_ButtonActive, ImVec4{ 0.8f, 0.1f, 0.15f, 1.0f });
00184     ImGui::PushFont(boldFont);
00185     if (ImGui::Button("X", buttonSize))
00186         values.x = resetValue;
00187     ImGui::PopFont();
00188     ImGui::PopStyleColor(3);
00189
00190     ImGui::SameLine();
00191     ImGui::PushItemWidth(itemWidth);
00192     ImGui::DragFloat("#X", &values.x, 0.1f, 0.0f, 0.0f, "%.2f");
00193     ImGui::PopItemWidth();
00194     ImGui::SameLine();
00195
00196     ImGui::PushStyleColor(ImGuiCol_Button, ImVec4{ 0.2f, 0.7f, 0.2f, 1.0f });
00197     ImGui::PushStyleColor(ImGuiCol_ButtonHovered, ImVec4{ 0.3f, 0.8f, 0.3f, 1.0f });
00198     ImGui::PushStyleColor(ImGuiCol_ButtonActive, ImVec4{ 0.2f, 0.7f, 0.2f, 1.0f });
00199     ImGui::PushFont(boldFont);
00200     if (ImGui::Button("Y", buttonSize))
00201         values.y = resetValue;
00202     ImGui::PopFont();
00203     ImGui::PopStyleColor(3);
00204
00205     ImGui::SameLine();
00206     ImGui::PushItemWidth(itemWidth);
00207     ImGui::DragFloat("#Y", &values.y, 0.1f, 0.0f, 0.0f, "%.2f");
00208     ImGui::PopItemWidth();
00209     ImGui::SameLine();
00210
00211     ImGui::PushStyleColor(ImGuiCol_Button, ImVec4{ 0.1f, 0.25f, 0.8f, 1.0f });
00212     ImGui::PushStyleColor(ImGuiCol_ButtonHovered, ImVec4{ 0.2f, 0.35f, 0.9f, 1.0f });
00213     ImGui::PushStyleColor(ImGuiCol_ButtonActive, ImVec4{ 0.1f, 0.25f, 0.8f, 1.0f });
00214     ImGui::PushFont(boldFont);
00215     if (ImGui::Button("Z", buttonSize))
00216         values.z = resetValue;
00217     ImGui::PopFont();
00218     ImGui::PopStyleColor(3);
00219
00220     ImGui::SameLine();
00221     ImGui::PushItemWidth(itemWidth);
00222     ImGui::DragFloat("#Z", &values.z, 0.1f, 0.0f, 0.0f, "%.2f");
00223     ImGui::PopItemWidth();
00224
00225     ImGui::PopStyleVar();
00226
00227     ImGui::Columns(1);
00228
00229     ImGui::PopID();
00230
00231 }
```

9.2.5.8 `format_as()`

```
std::string Vesper::format_as (
    const Event & e) [inline]
```

Format an event as a string.

```
00105     {
00106         return e.ToString();
00107     }
```

9.2.5.9 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\(\)](#).

9.2.5.10 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 }
```

9.2.5.11 ShaderDataTypeSize()

```
uint32_t Vesper::ShaderDataTypeSize (
    ShaderDataType type) [static]
```

Returns the size in bytes of the given [ShaderDataType](#).

Parameters

| | |
|-------------------|--|
| <code>type</code> | The ShaderDataType to get the size of. |
|-------------------|--|

Returns

The size in bytes of the [ShaderDataType](#). Will assert if the type is unknown.

```
00019
00020     switch (type) {
00021         case ShaderDataType::Float:      return 4;
00022         case ShaderDataType::Float2:    return 4 * 2;
00023         case ShaderDataType::Float3:    return 4 * 3;
00024         case ShaderDataType::Float4:    return 4 * 4;
00025         case ShaderDataType::Mat3:     return 4 * 3 * 3;
00026         case ShaderDataType::Mat4:     return 4 * 4 * 4;
00027         case ShaderDataType::Int:      return 4;
00028         case ShaderDataType::Int2:    return 4 * 2;
00029         case ShaderDataType::Int3:    return 4 * 3;
00030         case ShaderDataType::Int4:    return 4 * 4;
00031         case ShaderDataType::Bool:    return 1;
00032     }
00033     VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00034     return 0;
00035 }
```

References [Bool](#), [Float](#), [Float2](#), [Float3](#), [Float4](#), [Int](#), [Int2](#), [Int3](#), [Int4](#), [Mat3](#), and [Mat4](#).

Referenced by [Vesper::BufferElement::BufferElement\(\)](#).

9.2.5.12 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 }
```

9.2.5.13 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 }
```

9.2.5.14 SubTextEdit()

```

void Vesper::SubTextEdit (
    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     else {
00325         ImGui::Text("No texture assigned.");
00326     }
00327 }
00328 }
00329 }
```

9.2.6 Variable Documentation

9.2.6.1 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::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::EndScene\(\)](#), [Vesper::Renderer2D::Flush\(\)](#), [Vesper::Renderer2D::FlushAndReset\(\)](#), [Vesper::Renderer2D::GetStats\(\)](#), [Vesper::Renderer2D::GetWhiteTexture\(\)](#), [Vesper::Renderer2D::Init\(\)](#), [Vesper::Renderer2D::ResetStats\(\)](#), [Vesper::Renderer2D::Shutdown\(\)](#) and [Vesper::Renderer2D::StartBatch\(\)](#).

9.2.6.2 s_GLFWInitialized

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

Referenced by [Vesper::WindowsWindow::Init\(\)](#).

9.2.6.3 s_MaxFrameBufferSize

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

TODO: Get the actual maximum size from the GPU!

9.3 Vesper::Color Namespace Reference

Provides commonly used colors and color manipulation functions.

Functions

- static `glm::vec4 White ()`
Returns a vec4 representing the color white.
- static `glm::vec4 Black ()`
Returns a vec4 representing the color black.
- static `glm::vec4 Gray ()`
Returns a vec4 representing the color gray.
- static `glm::vec4 Red ()`
Returns a vec4 representing the color red.
- static `glm::vec4 Orange ()`
Returns a vec4 representing the color orange.
- static `glm::vec4 Yellow ()`
Returns a vec4 representing the color yellow.
- static `glm::vec4 Green ()`
Returns a vec4 representing the color green.
- static `glm::vec4 Blue ()`
Returns a vec4 representing the color blue.
- static `glm::vec4 Indigo ()`
Returns a vec4 representing the color indigo.
- static `glm::vec4 Purple ()`
Returns a vec4 representing the color purple.
- static `glm::vec4 Cyan ()`
Returns a vec4 representing the color cyan.
- static `glm::vec4 Magenta ()`
Returns a vec4 representing the color magenta.
- static `glm::vec4 Pink ()`
Returns a vec4 representing the color pink.
- static `glm::vec4 Brown ()`
Returns a vec4 representing the color brown.
- static `glm::vec4 Transparent ()`
Returns a vec4 representing a fully transparent color.
- static `glm::vec4 StripAlpha (const glm::vec4 &color)`
Strips the alpha component from the given color, setting it to 1.0 (fully opaque).
- static `glm::vec4 SetAlpha (const glm::vec4 &color, float alpha=0.0f)`
Sets the alpha component of the given color to the specified value.

9.3.1 Detailed Description

Provides commonly used colors and color manipulation functions.

9.3.2 Function Documentation

9.3.2.1 Black()

```
glm::vec4 Vesper::Color::Black () [static]
```

Returns a vec4 representing the color black.

```
00018 { return glm::vec4(0.0f, 0.0f, 0.0f, 1.0f); }
```

9.3.2.2 Blue()

```
glm::vec4 Vesper::Color::Blue () [static]
```

Returns a vec4 representing the color blue.

```
00031 { return glm::vec4(0.0f, 0.0f, 1.0f, 1.0f); }
```

9.3.2.3 Brown()

```
glm::vec4 Vesper::Color::Brown () [static]
```

Returns a vec4 representing the color brown.

```
00044 { return glm::vec4(0.6f, 0.4f, 0.2f, 1.0f); }
```

9.3.2.4 Cyan()

```
glm::vec4 Vesper::Color::Cyan () [static]
```

Returns a vec4 representing the color cyan.

```
00038 { return glm::vec4(0.0f, 1.0f, 1.0f, 1.0f); }
```

9.3.2.5 Gray()

```
glm::vec4 Vesper::Color::Gray () [static]
```

Returns a vec4 representing the color gray.

```
00020 { return glm::vec4(0.5f, 0.5f, 0.5f, 1.0f); }
```

9.3.2.6 Green()

```
glm::vec4 Vesper::Color::Green () [static]
```

Returns a vec4 representing the color green.

```
00029 { return glm::vec4(0.0f, 1.0f, 0.0f, 1.0f); }
```

9.3.2.7 Indigo()

```
glm::vec4 Vesper::Color::Indigo () [static]
```

Returns a vec4 representing the color indigo.

```
00033 { return glm::vec4(0.29f, 0.0f, 0.51f, 1.0f); }
```

9.3.2.8 Magenta()

```
glm::vec4 Vesper::Color::Magenta () [static]
```

Returns a vec4 representing the color magenta.

```
00040 { return glm::vec4(1.0f, 0.0f, 1.0f, 1.0f); }
```

9.3.2.9 Orange()

```
glm::vec4 Vesper::Color::Orange () [static]
```

Returns a vec4 representing the color orange.

```
00025 { return glm::vec4(1.0f, 0.5f, 0.0f, 1.0f); }
```

9.3.2.10 Pink()

```
glm::vec4 Vesper::Color::Pink () [static]
```

Returns a vec4 representing the color pink.

```
00042 { return glm::vec4(1.0f, 0.75f, 0.8f, 1.0f); }
```

9.3.2.11 Purple()

```
glm::vec4 Vesper::Color::Purple () [static]
```

Returns a vec4 representing the color purple.

```
00035 { return glm::vec4(0.5f, 0.0f, 0.5f, 1.0f); }
```

9.3.2.12 Red()

```
glm::vec4 Vesper::Color::Red () [static]
```

Returns a vec4 representing the color red.

```
00023 { return glm::vec4(1.0f, 0.0f, 0.0f, 1.0f); }
```

9.3.2.13 SetAlpha()

```
glm::vec4 Vesper::Color::SetAlpha (
    const glm::vec4 & color,
    float alpha = 0.0f) [static]
```

Sets the alpha component of the given color to the specified value.

```
00051 { return glm::vec4(color.x, color.y, color.z, alpha); }
```

9.3.2.14 StripAlpha()

```
glm::vec4 Vesper::Color::StripAlpha (
    const glm::vec4 & color) [static]
```

Strips the alpha component from the given color, setting it to 1.0 (fully opaque).

```
00049 { return glm::vec4(color.x, color.y, color.z, 1.0f); }
```

9.3.2.15 Transparent()

```
glm::vec4 Vesper::Color::Transparent () [static]
```

Returns a vec4 representing a fully transparent color.

```
00046 { return glm::vec4(0.0f, 0.0f, 0.0f, 0.0f); }
```

9.3.2.16 White()

```
glm::vec4 Vesper::Color::White () [static]
```

Returns a vec4 representing the color white.

```
00016 { return glm::vec4(1.0f, 1.0f, 1.0f, 1.0f); }
```

9.3.2.17 Yellow()

```
glm::vec4 Vesper::Color::Yellow () [static]
```

Returns a vec4 representing the color yellow.

```
00027 { return glm::vec4(1.0f, 1.0f, 0.0f, 1.0f); }
```

9.4 Vesper::Key Namespace Reference

Namespace for keyboard key codes.

Enumerations

- enum : KeyCode {
 Space = 32 , Apostrophe = 39 , Comma = 44 , Minus = 45 ,
 Period = 46 , Slash = 47 , D0 = 48 , D1 = 49 ,
 D2 = 50 , D3 = 51 , D4 = 52 , D5 = 53 ,
 D6 = 54 , D7 = 55 , D8 = 56 , D9 = 57 ,
 Semicolon = 59 , Equal = 61 , A = 65 , B = 66 ,
 C = 67 , D = 68 , E = 69 , F = 70 ,
 G = 71 , H = 72 , I = 73 , J = 74 ,
 K = 75 , L = 76 , M = 77 , N = 78 ,
 O = 79 , P = 80 , Q = 81 , R = 82 ,
 S = 83 , T = 84 , U = 85 , V = 86 ,
 W = 87 , X = 88 , Y = 89 , Z = 90 ,
 LeftBracket = 91 , Backslash = 92 , RightBracket = 93 , GraveAccent = 96 ,
 World1 = 161 , World2 = 162 , Escape = 256 , Enter = 257 ,

```
Tab = 258 , Backspace = 259 , Insert = 260 , Delete = 261 ,
Right = 262 , Left = 263 , Down = 264 , Up = 265 ,
PageUp = 266 , PageDown = 267 , Home = 268 , End = 269 ,
CapsLock = 280 , ScrollLock = 281 , NumLock = 282 , PrintScreen = 283 ,
Pause = 284 , F1 = 290 , F2 = 291 , F3 = 292 ,
F4 = 293 , F5 = 294 , F6 = 295 , F7 = 296 ,
F8 = 297 , F9 = 298 , F10 = 299 , F11 = 300 ,
F12 = 301 , F13 = 302 , F14 = 303 , F15 = 304 ,
F16 = 305 , F17 = 306 , F18 = 307 , F19 = 308 ,
F20 = 309 , F21 = 310 , F22 = 311 , F23 = 312 ,
F24 = 313 , F25 = 314 , KP_0 = 320 , KP_1 = 321 ,
KP_2 = 322 , KP_3 = 323 , KP_4 = 324 , KP_5 = 325 ,
KP_6 = 326 , KP_7 = 327 , KP_8 = 328 , KP_9 = 329 ,
KP_DECIMAL = 330 , KP_DIVIDE = 331 , KP_MULTIPLY = 332 , KP_SUBTRACT = 333 ,
KP_ADD = 334 , KP_ENTER = 335 , KP_EQUAL = 336 , LeftShift = 340 ,
LeftControl = 341 , LeftAlt = 342 , LeftSuper = 343 , RightShift = 344 ,
RightControl = 345 , RightAlt = 346 , RightSuper = 347 , Menu = 348 }
```

Enumeration of keyboard key codes.

9.4.1 Detailed Description

Namespace for keyboard key codes.

9.4.2 Enumeration Type Documentation

9.4.2.1 anonymous enum

```
anonymous enum : KeyCode
```

Enumeration of keyboard key codes.

Enumerator

| | |
|------------|--|
| Space | |
| Apostrophe | |
| Comma | |
| Minus | |
| Period | |
| Slash | |
| D0 | |
| D1 | |
| D2 | |
| D3 | |
| D4 | |
| D5 | |
| D6 | |
| D7 | |

| | |
|--------------|--|
| D8 | |
| D9 | |
| Semicolon | |
| Equal | |
| A | |
| B | |
| C | |
| D | |
| E | |
| F | |
| G | |
| H | |
| I | |
| J | |
| K | |
| L | |
| M | |
| N | |
| O | |
| P | |
| Q | |
| R | |
| S | |
| T | |
| U | |
| V | |
| W | |
| X | |
| Y | |
| Z | |
| LeftBracket | |
| Backslash | |
| RightBracket | |
| GraveAccent | |
| World1 | |
| World2 | |
| Escape | |
| Enter | |
| Tab | |
| Backspace | |
| Insert | |
| Delete | |

| | |
|-------------|--|
| Right | |
| Left | |
| Down | |
| Up | |
| PageUp | |
| PageDown | |
| Home | |
| End | |
| CapsLock | |
| ScrollLock | |
| NumLock | |
| PrintScreen | |
| Pause | |
| F1 | |
| F2 | |
| F3 | |
| F4 | |
| F5 | |
| F6 | |
| F7 | |
| F8 | |
| F9 | |
| F10 | |
| F11 | |
| F12 | |
| F13 | |
| F14 | |
| F15 | |
| F16 | |
| F17 | |
| F18 | |
| F19 | |
| F20 | |
| F21 | |
| F22 | |
| F23 | |
| F24 | |
| F25 | |
| KP_0 | |
| KP_1 | |

| | |
|--------------|--|
| KP_2 | |
| KP_3 | |
| KP_4 | |
| KP_5 | |
| KP_6 | |
| KP_7 | |
| KP_8 | |
| KP_9 | |
| KP_DECIMAL | |
| KP_DIVIDE | |
| KP_MULTIPLY | |
| KP_SUBTRACT | |
| KP_ADD | |
| KP_ENTER | |
| KP_EQUAL | |
| LeftShift | |
| LeftControl | |
| LeftAlt | |
| LeftSuper | |
| RightShift | |
| RightControl | |
| RightAlt | |
| RightSuper | |
| Menu | |

```

00014      {
00015          // From glfw3.h
00016
00017      Space = 32,
00018      Apostrophe = 39, /* ' */
00019      Comma = 44, /* , */
00020      Minus = 45, /* - */
00021      Period = 46, /* . */
00022      Slash = 47, /* / */
00023
00024      D0 = 48,
00025      D1 = 49,
00026      D2 = 50,
00027      D3 = 51,
00028      D4 = 52,
00029      D5 = 53,
00030      D6 = 54,
00031      D7 = 55,
00032      D8 = 56,
00033      D9 = 57,
00034
00035      Semicolon = 59, /* ; */
00036      Equal = 61, /* = */
00037

```

```
00038      A = 65,
00039      B = 66,
00040      C = 67,
00041      D = 68,
00042      E = 69,
00043      F = 70,
00044      G = 71,
00045      H = 72,
00046      I = 73,
00047      J = 74,
00048      K = 75,
00049      L = 76,
00050      M = 77,
00051      N = 78,
00052      O = 79,
00053      P = 80,
00054      Q = 81,
00055      R = 82,
00056      S = 83,
00057      T = 84,
00058      U = 85,
00059      V = 86,
00060      W = 87,
00061      X = 88,
00062      Y = 89,
00063      Z = 90,
00064
00065      LeftBracket = 91, /* [ */
00066      Backslash = 92, /* \ */
00067      RightBracket = 93, /* ] */
00068      GraveAccent = 96, /* ` */
00069
00070      World1 = 161, /* non-US #1 */
00071      World2 = 162, /* non-US #2 */
00072
00073      /* Function keys */
00074      Escape = 256,
00075      Enter = 257,
00076      Tab = 258,
00077      Backspace = 259,
00078      Insert = 260,
00079      Delete = 261,
00080      Right = 262,
00081      Left = 263,
00082      Down = 264,
00083      Up = 265,
00084      PageUp = 266,
00085      PageDown = 267,
00086      Home = 268,
00087      End = 269,
00088      CapsLock = 280,
00089      ScrollLock = 281,
00090      NumLock = 282,
00091      PrintScreen = 283,
00092      Pause = 284,
00093      F1 = 290,
00094      F2 = 291,
00095      F3 = 292,
00096      F4 = 293,
00097      F5 = 294,
00098      F6 = 295,
00099      F7 = 296,
00100      F8 = 297,
00101      F9 = 298,
00102      F10 = 299,
00103      F11 = 300,
00104      F12 = 301,
00105      F13 = 302,
00106      F14 = 303,
00107      F15 = 304,
00108      F16 = 305,
00109      F17 = 306,
00110      F18 = 307,
00111      F19 = 308,
00112      F20 = 309,
00113      F21 = 310,
00114      F22 = 311,
00115      F23 = 312,
00116      F24 = 313,
00117      F25 = 314,
00118
00119      /* Keypad */
00120      KP_0 = 320,
00121      KP_1 = 321,
00122      KP_2 = 322,
00123      KP_3 = 323,
00124      KP_4 = 324,
```

```

00125     KP_5 = 325,
00126     KP_6 = 326,
00127     KP_7 = 327,
00128     KP_8 = 328,
00129     KP_9 = 329,
00130     KP_DECIMAL = 330,
00131     KP_DIVIDE = 331,
00132     KP_MULTIPLY = 332,
00133     KP_SUBTRACT = 333,
00134     KP_ADD = 334,
00135     KP_ENTER = 335,
00136     KP_EQUAL = 336,
00137
00138
00139     LeftShift = 340,
00140     LeftControl = 341,
00141     LeftAlt = 342,
00142     LeftSuper = 343,
00143     RightShift = 344,
00144     RightControl = 345,
00145     RightAlt = 346,
00146     RightSuper = 347,
00147     Menu = 348
00148 };

```

9.5 Vesper::Math Namespace Reference

Functions

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

Decomposes a transformation matrix into its translation, rotation, and scale components.

9.5.1 Function Documentation

9.5.1.1 DecomposeTransform()

```

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

```

Decomposes a transformation matrix into its translation, rotation, and scale components.

```

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     return true;
00079 }

```

9.6 Vesper::Mouse Namespace Reference

Namespace for mouse button codes.

Enumerations

- enum : MouseCode {
 Button0 = 0 , Button1 = 1 , Button2 = 2 , Button3 = 3 ,
 Button4 = 4 , Button5 = 5 , Button6 = 6 , Button7 = 7 ,
 ButtonLast = Button7 , ButtonLeft = Button0 , ButtonRight = Button1 , ButtonMiddle = Button2 }
- Enumeration of mouse button codes.*

9.6.1 Detailed Description

Namespace for mouse button codes.

9.6.2 Enumeration Type Documentation

9.6.2.1 anonymous enum

```
anonymous enum : MouseCode
```

Enumeration of mouse button codes.

Enumerator

| | |
|--------------|--|
| Button0 | |
| Button1 | |
| Button2 | |
| Button3 | |
| Button4 | |
| Button5 | |
| Button6 | |
| Button7 | |
| ButtonLast | |
| ButtonLeft | |
| ButtonRight | |
| ButtonMiddle | |

```
00014      {
00015          // From glfw3.h
00016          Button0 = 0,
00017          Button1 = 1,
00018          Button2 = 2,
00019          Button3 = 3,
00020          Button4 = 4,
00021          Button5 = 5,
00022          Button6 = 6,
00023          Button7 = 7,
00024
00025          ButtonLast = Button7,
00026          ButtonLeft = Button0,
00027          ButtonRight = Button1,
00028          ButtonMiddle = Button2,
00029      };
```

9.7 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)`

9.7.1 Function Documentation

9.7.1.1 Bool1()

```
bool Vesper::Random::Bool1 (
    float trueChance) [inline]
00031 {
00032     VZ_PROFILE_FUNCTION();
00033     std::uniform_real_distribution<float> dist(0.0f, 1.0f);
00034     return dist(GetRNG()) < trueChance;
00035 }
```

9.7.1.2 Char()

```
unsigned char Vesper::Random::Char () [inline]
00037 {
00038     VZ_PROFILE_FUNCTION();
00039     std::uniform_int_distribution<int> dist(0, 255);
00040     return static_cast<unsigned char>(dist(GetRNG()));
00041 }
```

9.7.1.3 Float1()

```
float Vesper::Random::Float1 () [inline]
00076 {
00077     VZ_PROFILE_FUNCTION();
00078     static thread_local std::uniform_real_distribution<float> dist(0.0f, 1.0f);
00079     return dist(GetRNG());
00080 }
```

Referenced by [Vesper::ParticleSystem::Emit\(\)](#).

9.7.1.4 Float2()

```
glm::vec2 Vesper::Random::Float2 () [inline]
00100 {
00101     return glm::vec2( Float1(), Float1() );
00102 }
```

9.7.1.5 **Float3()**

```
glm::vec3 Vesper::Random::Float3 () [inline]
00121         {
00122             return glm::vec3{ Float1(), Float1(), Float1() };
00123 }
```

9.7.1.6 **Float4()**

```
glm::vec4 Vesper::Random::Float4 () [inline]
00141         {
00142             return glm::vec4{ Float1(), Float1(), Float1(), Float1() };
00143 }
```

9.7.1.7 **GetRNG()**

```
std::mt19937 & Vesper::Random::GetRNG () [inline]
00016         {
00017             static thread_local std::mt19937 rng{ std::random_device{}() };
00018             return rng;
00019 }
```

Referenced by [Seed\(\)](#).

9.7.1.8 **HexString()**

```
std::string Vesper::Random::HexString (
    size_t length) [inline]
00057         {
00058             VZ_PROFILE_FUNCTION();
00059             const char charset[] =
00060                 "0123456789";
00061                 "ABCDEF";
00062             const size_t max_index = (sizeof(charset) - 1);
00063             std::string str(length, 0);
00064             for (size_t i = 0; i < length; ++i) {
00065                 str[i] = charset[UIInt1(static_cast<uint32_t>(max_index))];
00066             }
00067             return str;
00068 }
```

References [UIInt1\(\)](#).

9.7.1.9 **RangeF1()**

```
float Vesper::Random::RangeF1 (
    float min,
    float max) [inline]
00084         {
00085             VZ_PROFILE_FUNCTION();
00086             if (min > max) std::swap(min, max);
00087             std::uniform_real_distribution<float> dist(min, max);
00088             return dist(GetRNG());
00089 }
```

9.7.1.10 RangeF1_Inclusive()

```

float Vesper::Random::RangeF1_Inclusive (
    float min,
    float max) [inline]
00092 {
00093     if (min > max) std::swap(min, max);
00094     float upper = std::nextafter(max, std::numeric_limits<float>::infinity());
00095     std::uniform_real_distribution<float> dist(min, upper);
00096     return dist(GetRNG());
00097 }
```

9.7.1.11 RangeF2() [1/3]

```

glm::vec2 Vesper::Random::RangeF2 (
    const glm::vec2 & minRange,
    const glm::vec2 & maxRange) [inline]
00115 {
00116     return glm::vec2{ RangeF1(minRange.x, maxRange.x), RangeF1(minRange.y, maxRange.y) };
00117 }
```

9.7.1.12 RangeF2() [2/3]

```

glm::vec2 Vesper::Random::RangeF2 (
    float min,
    float max) [inline]
00105 {
00106     return glm::vec2{ RangeF1(min, max), RangeF1(min, max) };
00107 }
```

9.7.1.13 RangeF2() [3/3]

```

glm::vec2 Vesper::Random::RangeF2 (
    float min1,
    float max1,
    float min2,
    float max2) [inline]
00110 {
00111     return glm::vec2{ RangeF1(min1, max1), RangeF1(min2, max2) };
00112 }
```

9.7.1.14 RangeF3() [1/3]

```

glm::vec3 Vesper::Random::RangeF3 (
    const glm::vec2 & range1,
    const glm::vec2 & range2,
    const glm::vec2 & range3) [inline]
00136 {
00137     return glm::vec3{ RangeF1(range1.x, range1.y), RangeF1(range2.x, range2.y),
00138     RangeF1(range3.x, range3.y) };
00139 }
```

9.7.1.15 RangeF3() [2/3]

```
glm::vec3 Vesper::Random::RangeF3 (
    float min,
    float max) [inline]
00126     {
00127         return glm::vec3{ RangeF1(min, max), RangeF1(min, max), RangeF1(min, max) };
00128     }
```

9.7.1.16 RangeF3() [3/3]

```
glm::vec3 Vesper::Random::RangeF3 (
    float min1,
    float max1,
    float min2,
    float max2,
    float min3,
    float max3) [inline]
00131     {
00132         return glm::vec3{ RangeF1(min1, max1), RangeF1(min2, max2), RangeF1(min3, max3) };
00133     }
```

9.7.1.17 RangeF4()

```
glm::vec4 Vesper::Random::RangeF4 (
    float min,
    float max) [inline]
00146     {
00147         return glm::vec4{ RangeF1(min, max), RangeF1(min, max), RangeF1(min, max), RangeF1(min,
max) };
00148     }
```

9.7.1.18 Seed()

```
void Vesper::Random::Seed (
    uint32_t seed) [inline]
00021     {
00022         GetRNG().seed(seed);
00023     }
```

References [GetRNG\(\)](#).

9.7.1.19 String()

```
std::string Vesper::Random::String (
    size_t length) [inline]
00043     {
00044         VZ_PROFILE_FUNCTION();
00045         const char charset[] =
00046             "0123456789";
00047             "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
00048             "abcdefghijklmnopqrstuvwxyz";
00049         const size_t max_index = (sizeof(charset) - 1);
00050         std::string str(length, 0);
00051         for (size_t i = 0; i < length; ++i) {
00052             str[i] = charset[UIInt1(static_cast<uint32_t>(max_index))];
00053         }
00054         return str;
00055     }
```

References [UIInt1\(\)](#).

9.7.1.20 UInt1()

```

00025     uint32_t Vesper::Random::UInt1 (
00026         uint32_t max) [inline]
00027     {
00028         VZ_PROFILE_FUNCTION();
00029         std::uniform_int_distribution<uint32_t> dist(0, max - 1);
00030         return dist(GetRNG());
00031     }

```

Referenced by [HexString\(\)](#), and [String\(\)](#).

9.7.1.21 UUID()

```

std::string Vesper::Random::UUID () [inline]
00070     {
00071         VZ_PROFILE_FUNCTION();
00072         return HexString(8) + "-" + HexString(4) + "-" + HexString(4) + "-" +
00073             HexString(4) + "-" + HexString(12);

```

9.8 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<< \(YAML::Emitter &out, const glm::vec4 &v\)](#)

9.8.1 Function Documentation

9.8.1.1 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     }

```

9.8.1.2 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     }
```

9.8.1.3 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 10

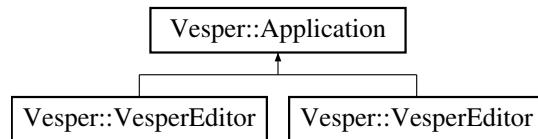
Class Documentation

10.1 Vesper::Application Class Reference

The core application class that manages the main loop, window, layers, and event handling.

```
#include <Application.h>
```

Inheritance diagram for Vesper::Application:



Public Member Functions

- **Application** (const std::string &name="")
Constructs the Application with the given name.
- virtual **~Application** ()
- void **Run** ()
Starts the main application loop.
- void **OnEvent** (Event &e)
Handles incoming events and dispatches them to the appropriate handlers.
- void **PushLayer** (Layer *layer)
Adds a layer to the application layer stack.
- void **PushOverlay** (Layer *overlay)
Adds an overlay layer to the application layer stack.
- void **Close** ()
Closes the application.
- ImGuiLayer * **GetImGuiLayer** ()
Retrieves the ImGui layer.
- Window & **GetWindow** ()
Retrieves the application window.

Static Public Member Functions

- static [Application & Get \(\)](#)
Retrieves the singleton instance of the Application.

Private Member Functions

- bool [OnWindowClose \(WindowCloseEvent &e\)](#)
Event handler for window close events.
- bool [OnWindowResize \(WindowResizeEvent &e\)](#)
Event handler for window resize events.

Private Attributes

- [Scope< Window > m_Window](#)
Scoped pointer to the applications underlying window.
- [ImGuiLayer * m_ImGuiLayer](#)
ImGui layer for rendering GUI elements.
- bool [m_Running = true](#)
Flag indicating whether the application is running.
- bool [m_Minimized = false](#)
Flag indicating whether the application is minimized.
- [LayerStack m_LayerStack](#)
Stack of layers managed by the application.
- float [m_LastFrameTime = 0.0f](#)
Time of the last frame, used for calculating timestep.

Static Private Attributes

- static [Application * s_Instance = nullptr](#)

10.1.1 Detailed Description

The core application class that manages the main loop, window, layers, and event handling.

Todo Update class to accept [ApplicationSettings](#) in constructor and verify its utility/use

10.1.2 Constructor & Destructor Documentation

10.1.2.1 Application()

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

Constructs the [Application](#) with the given name.

Parameters

| | |
|-------------|---|
| <i>name</i> | The name of the application. Defaults to an empty string. |
|-------------|---|

```
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         Renderer::Init();
00029
00030
00031         m_ImGuiLayer = new ImGuiLayer();
00032         PushOverlay(m_ImGuiLayer);
00033
00034     }
```

References [Vesper::ImGuiLayer::ImGuiLayer\(\)](#), [Vesper::Renderer::Init\(\)](#), [m_ImGuiLayer](#), [PushOverlay\(\)](#), and [s_Instance](#).

10.1.2.2 ~Application()

```
Vesper::Application::~Application () [virtual]
00038     {
00039 }
```

10.1.3 Member Function Documentation

10.1.3.1 Close()

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

Closes the application.

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

References [m_Running](#).

Referenced by [Vesper::EditorLayer::OnImGuiRender\(\)](#).

10.1.3.2 Get()

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

Retrieves the singleton instance of the [Application](#).

```
00067 { 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\(\)](#).

10.1.3.3 GetImGuiLayer()

```
ImGuiLayer * Vesper::Application::GetImGuiLayer () [inline]
```

Retrieves the ImGui layer.

```
00064 { return m_ImGuiLayer; }
```

References [m_ImGuiLayer](#).

Referenced by [Vesper::EditorLayer::OnImGuiRender\(\)](#).

10.1.3.4 GetWindow()

```
Window & Vesper::Application::GetWindow () [inline]
```

Retrieves the application window.

```
00070 { return *m_Window; }
```

Referenced by [Vesper::ImGuiLayer::End\(\)](#), [Vesper::Input::GetMousePosition\(\)](#), [Vesper::Input::IsKeyPressed\(\)](#), and [Vesper::Input::IsMouseButtonPressed\(\)](#).

10.1.3.5 OnEvent()

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

Handles incoming events and dispatches them to the appropriate handlers.

```
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\(\)](#).

10.1.3.6 OnWindowClose()

```
bool Vesper::Application::OnWindowClose (
    WindowCloseEvent & e) [private]
```

Event handler for window close events.

```
00112     {
00113         VZ_PROFILE_FUNCTION();
00114         m_Running = false;
00115         return true;
00116     }
```

References [m_Running](#).

Referenced by [OnEvent\(\)](#).

10.1.3.7 OnWindowResize()

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

Event handler for window resize events.

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

References [Vesper::WindowResizeEvent::GetHeight\(\)](#), [Vesper::WindowResizeEvent::GetWidth\(\)](#), [m_Minimized](#), and [Vesper::Renderer::OnWindowResize\(\)](#).

Referenced by [OnEvent\(\)](#).

10.1.3.8 PushLayer()

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

Adds a layer to the application layer stack.

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

References [Vesper::Layer::OnAttach\(\)](#).

Referenced by [Vesper::VesperEditor::VesperEditor\(\)](#).

10.1.3.9 PushOverlay()

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

Adds an overlay layer to the application layer stack.

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

References [Vesper::Layer::OnAttach\(\)](#).

Referenced by [Application\(\)](#).

10.1.3.10 Run()

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

Starts the main application loop.

Todo Add [Layer](#) rendering into the main loop

Add separate threads for rendering and updating

```
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](#).

10.1.4 Member Data Documentation

10.1.4.1 m_ImGuiLayer

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

ImGui layer for rendering GUI elements.

Referenced by [Application\(\)](#), [GetImGuiLayer\(\)](#), and [Run\(\)](#).

10.1.4.2 m_LastFrameTime

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

Time of the last frame, used for calculating timestep.

Referenced by [Run\(\)](#).

10.1.4.3 m_LayerStack

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

Stack of layers managed by the application.

10.1.4.4 m_Minimized

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

Flag indicating whether the application is minimized.

Referenced by [OnWindowResize\(\)](#), and [Run\(\)](#).

10.1.4.5 m_Running

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

Flag indicating whether the application is running.

Referenced by [Close\(\)](#), [OnWindowClose\(\)](#), and [Run\(\)](#).

10.1.4.6 m_Window

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

Scoped pointer to the applications underlying window.

10.1.4.7 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:

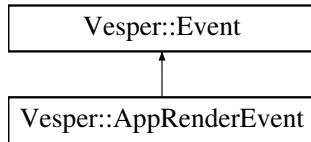
- Vesper/src/Vesper/App/[Application.h](#)
- Vesper/src/Vesper/App/[Application.cpp](#)

10.2 Vesper::AppRenderEvent Class Reference

[Event](#) for registering application render.

```
#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](#)

Get the type of the event.
- virtual const char * [GetName \(\) const =0](#)

Get the name of the event.
- virtual int [GetCategoryFlags \(\) const =0](#)

Get the category flags of the event.
- virtual std::string [ToString \(\) const](#)

Convert the event to a string representation.
- bool [IsInCategory \(EventCategory category\)](#)

Check if the event is in a specific category.

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled = false](#)

Indicates whether the event has been handled.

10.2.1 Detailed Description

[Event](#) for registering application render.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 AppRenderEvent()

```
Vesper::AppRenderEvent::AppRenderEvent () [inline]
00084 {}
```

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

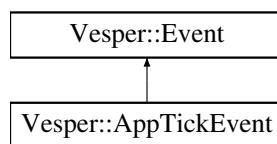
- Vesper/src/Vesper/Events/[ApplicationEvent.h](#)

10.3 Vesper::AppTickEvent Class Reference

[Event](#) for registering application tick.

```
#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](#)
Get the type of the event.
- virtual const char * [GetName \(\) const =0](#)
Get the name of the event.
- virtual int [GetCategoryFlags \(\) const =0](#)
Get the category flags of the event.
- virtual std::string [ToString \(\) const](#)
Convert the event to a string representation.
- bool [IsInCategory \(EventCategory category\)](#)
Check if the event is in a specific category.

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled](#) = false
Indicates whether the event has been handled.

10.3.1 Detailed Description

[Event](#) for registering application tick.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 AppTickEvent()

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

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

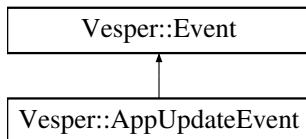
- Vesper/src/Vesper/Events/[ApplicationEvent.h](#)

10.4 Vesper::AppUpdateEvent Class Reference

[Event](#) for registering application update.

```
#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](#)
Get the type of the event.
- virtual const char * [GetName \(\) const =0](#)
Get the name of the event.
- virtual int [GetCategoryFlags \(\) const =0](#)
Get the category flags of the event.
- virtual std::string [ToString \(\) const](#)
Convert the event to a string representation.
- bool [IsInCategory \(EventCategory category\)](#)
Check if the event is in a specific category.

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled](#) = false
Indicates whether the event has been handled.

10.4.1 Detailed Description

[Event](#) for registering application update.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 AppUpdateEvent()

```
Vesper::AppUpdateEvent::AppUpdateEvent () [inline]  
00073 {}
```

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

- Vesper/src/Vesper/Events/[ApplicationEvent.h](#)

10.5 Vesper::BufferElement Struct Reference

Represents a single element in a buffer layout.

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

Public Member Functions

- [BufferElement \(\)](#)
- [BufferElement \(ShaderDataType type, const std::string &name, bool normalized=false\)](#)
Constructs a [BufferElement](#) with the given type, name, and normalization flag.
- [uint32_t GetComponentCount \(\) const](#)
Returns the number of components in the buffer element based on its [ShaderDataType](#).

Public Attributes

- [std::string Name](#)
The name of the buffer element.
- [ShaderDataType Type](#)
The data type of the buffer element.
- [uint32_t Size](#)
The size in bytes of the buffer element.
- [uint32_t Offset](#)
The offset in bytes of the buffer element from the start of the buffer.
- [bool Normalized](#)
Whether the buffer element is normalized.

10.5.1 Detailed Description

Represents a single element in a buffer layout.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 BufferElement() [1/2]

```
Vesper::BufferElement::BufferElement () [inline]
00051 { }
```

10.5.2.2 BufferElement() [2/2]

```
Vesper::BufferElement::BufferElement (
    ShaderDataType type,
    const std::string & name,
    bool normalized = false) [inline]
```

Constructs a [BufferElement](#) with the given type, name, and normalization flag.

Parameters

| | |
|-------------------|---|
| <i>type</i> | The ShaderDataType of the buffer element. |
| <i>name</i> | The name of the buffer element. |
| <i>normalized</i> | Whether the buffer element is normalized. |

```
00058     : Name(name), Type(type), Size(ShaderDataTypeSize(type)), Offset(0),
00059     Normalized(normalized)
00060 }
```

References [BufferElement\(\)](#), [Normalized](#), [Offset](#), [Vesper::ShaderDataTypeSize\(\)](#), [Size](#), and [Type](#).

Referenced by [BufferElement\(\)](#).

10.5.3 Member Function Documentation

10.5.3.1 GetComponentCount()

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

Returns the number of components in the buffer element based on its [ShaderDataType](#).

```
00063                                     {
00064     switch (Type) {
00065         case ShaderDataType::Float:      return 1;
00066         case ShaderDataType::Float2:    return 2;
00067         case ShaderDataType::Float3:    return 3;
00068         case ShaderDataType::Float4:    return 4;
00069         case ShaderDataType::Mat3:     return 3 * 3;
00070         case ShaderDataType::Mat4:     return 4 * 4;
00071         case ShaderDataType::Int:      return 1;
00072         case ShaderDataType::Int2:    return 2;
00073         case ShaderDataType::Int3:    return 3;
00074         case ShaderDataType::Int4:    return 4;
00075         case ShaderDataType::Bool:    return 1;
00076     }
00077     VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00078     return 0;
00079 }
```

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](#).

10.5.4 Member Data Documentation

10.5.4.1 Name

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

The name of the buffer element.

10.5.4.2 Normalized

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

Whether the buffer element is normalized.

Referenced by [BufferElement\(\)](#).

10.5.4.3 Offset

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

The offset in bytes of the buffer element from the start of the buffer.

Referenced by [BufferElement\(\)](#).

10.5.4.4 Size

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

The size in bytes of the buffer element.

Referenced by [BufferElement\(\)](#).

10.5.4.5 Type

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

The data type of the buffer element.

Referenced by [BufferElement\(\)](#), and [GetComponentCount\(\)](#).

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

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

10.6 Vesper::BufferLayout Class Reference

Represents the layout of a buffer, consisting of multiple BufferElements.

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

Public Member Functions

- [BufferLayout \(\)](#)
- [BufferLayout \(const std::initializer_list< BufferElement > &elements\)](#)

Constructs a [BufferLayout](#) with the given list of BufferElements.
- [const std::vector< BufferElement > & GetElements \(\) const](#)

Returns the list of BufferElements in the layout.
- [uint32_t GetStride \(\) const](#)

Returns the stride (total size in bytes) of the buffer layout.
- [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 \(\)](#)

Calculates the offsets and stride for the buffer layout based on its elements.

Private Attributes

- [std::vector< BufferElement > m_Elements](#)
- [uint32_t m_Stride = 0](#)

10.6.1 Detailed Description

Represents the layout of a buffer, consisting of multiple BufferElements.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 BufferLayout() [1/2]

```
Vesper::BufferLayout::BufferLayout () [inline]
00086 { }
```

10.6.2.2 BufferLayout() [2/2]

```
Vesper::BufferLayout::BufferLayout (
    const std::initializer_list< BufferElement > & elements) [inline]
```

Constructs a [BufferLayout](#) with the given list of BufferElements.

Parameters

| | |
|-----------------------|--|
| <code>elements</code> | The list of BufferElements that make up the layout. Automatically calculates offsets and stride. |
|-----------------------|--|

```
00093     : m_Elements(elements), m_Stride(0)
00094     {
00095         CalculateOffsetsAndStride();
00096     }
```

References [BufferLayout\(\)](#), and [m_Stride](#).

Referenced by [BufferLayout\(\)](#).

10.6.3 Member Function Documentation

10.6.3.1 begin() [1/2]

```
std::vector< BufferElement >::iterator Vesper::BufferLayout::begin () [inline]
00104 { return m_Elements.begin(); }
```

10.6.3.2 begin() [2/2]

```
std::vector< BufferElement >::const_iterator Vesper::BufferLayout::begin () const [inline]
00105 { return m_Elements.begin(); }
```

10.6.3.3 CalculateOffsetsAndStride()

```
void Vesper::BufferLayout::CalculateOffsetsAndStride () [inline], [private]
```

Calculates the offsets and stride for the buffer layout based on its elements.

```
00113 {
00114     uint32_t offset = 0;
00115     m_Stride = 0;
00116     for (auto& element : m_Elements) {
00117         element.Offset = offset;
00118         offset += element.Size;
00119         m_Stride += element.Size;
00120     }
00121 }
```

10.6.3.4 end() [1/2]

```
std::vector< BufferElement >::iterator Vesper::BufferLayout::end () [inline]
00106 { return m_Elements.end(); }
```

10.6.3.5 end() [2/2]

```
std::vector< BufferElement >::const_iterator Vesper::BufferLayout::end () const [inline]
00107 { return m_Elements.end(); }
```

10.6.3.6 GetElements()

```
const std::vector< BufferElement > & Vesper::BufferLayout::GetElements () const [inline]
```

Returns the list of BufferElements in the layout.

```
00100 { return m_Elements; }
```

10.6.3.7 GetStride()

```
uint32_t Vesper::BufferLayout::GetStride () const [inline]
```

Returns the stride (total size in bytes) of the buffer layout.

```
00102 { return m_Stride; }
```

References [m_Stride](#).

10.6.4 Member Data Documentation

10.6.4.1 m_Elements

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

10.6.4.2 m_Stride

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

Referenced by [BufferLayout\(\)](#), and [GetStride\(\)](#).

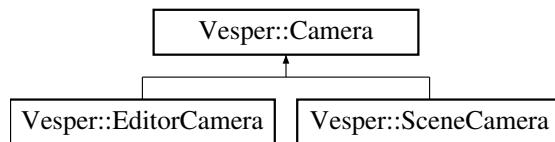
The documentation for this class was generated from the following file:

- [Vesper/src/Vesper/Renderer/Buffer.h](#)

10.7 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\(1.0f\)](#)

10.7.1 Constructor & Destructor Documentation

10.7.1.1 Camera() [1/2]

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

10.7.1.2 Camera() [2/2]

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

References [Camera\(\)](#).

Referenced by [Camera\(\)](#).

10.7.1.3 ~Camera()

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

10.7.2 Member Function Documentation

10.7.2.1 GetProjection()

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

10.7.3 Member Data Documentation

10.7.3.1 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](#)

10.8 Vesper::CameraComponent Struct Reference

Component that holds camera data.

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

Public Member Functions

- [CameraComponent \(\)=default](#)
- [CameraComponent \(const CameraComponent &\)=default](#)

Public Attributes

- **SceneCamera Camera**
The scene camera.
- bool **Primary** = true
Whether this camera is the primary camera.
- bool **FixedAspectRatio** = false
Whether the aspect ratio is fixed.

10.8.1 Detailed Description

Component that holds camera data.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 CameraComponent() [1/2]

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

10.8.2.2 CameraComponent() [2/2]

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

10.8.3 Member Data Documentation

10.8.3.1 Camera

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

The scene camera.

10.8.3.2 FixedAspectRatio

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

Whether the aspect ratio is fixed.

10.8.3.3 Primary

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

Whether this camera is the primary camera.

If multiple cameras exist, the primary camera for rendering will be the first one found marked as primary.

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

- [Vesper/src/Vesper/Scene/Components.h](#)

10.9 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)

10.9.1 Member Function Documentation

10.9.1.1 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 }
```

10.9.1.2 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 }
```

10.9.1.3 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 }
```

10.9.1.4 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.setStyleEmitterStyle::Flow);
00023         return node;
00024     }

```

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

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

10.10 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)

10.10.1 Member Function Documentation

10.10.1.1 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     }

```

10.10.1.2 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     }

```

10.10.1.3 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     }
```

10.10.1.4 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

10.11 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)

10.11.1 Member Function Documentation

10.11.1.1 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     }
```

10.11.1.2 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 }

```

10.11.1.3 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 }

```

10.11.1.4 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 }

```

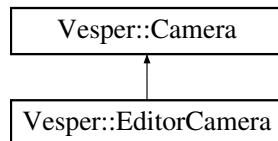
The documentation for this struct was generated from the following file:

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

10.12 Vesper::EditorCamera Class Reference

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

Inheritance diagram for Vesper::EditorCamera:



Public Member Functions

- `EditorCamera ()`
- `EditorCamera (float fov, float aspectRatio, 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)

10.12.1 Constructor & Destructor Documentation

10.12.1.1 EditorCamera() [1/2]

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

10.12.1.2 EditorCamera() [2/2]

```
Vesper::EditorCamera::EditorCamera (
    float fov,
    float aspectRatio,
    float nearClip,
    float farClip)
00016     : m\_FOV(fov), m\_AspectRatio(aspectRatio), m\_NearClip(nearClip), m\_FarClip(farClip),
00017     Camera(glm::perspective(glm::radians(fov), aspectRatio, nearClip, farClip))
00018     {
00019         UpdateView();
}
```

References [EditorCamera\(\)](#), [m_AspectRatio](#), [m_FarClip](#), [m_FOV](#), [m_NearClip](#), and [UpdateView\(\)](#).

Referenced by [EditorCamera\(\)](#).

10.12.2 Member Function Documentation

10.12.2.1 CalculatePosition()

```
glm::vec3 Vesper::EditorCamera::CalculatePosition () const [private]
00120     {
00121         return m\_FocalPoint - GetForwardDirection() * m\_Distance;
00122     }
```

10.12.2.2 GetDistance()

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

References [m_Distance](#).

10.12.2.3 GetForwardDirection()

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

10.12.2.4 GetOrientation()

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

10.12.2.5 GetPitch()

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

References [m_Pitch](#).

10.12.2.6 GetPosition()

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

10.12.2.7 GetRightDirection()

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

10.12.2.8 GetUpDirection()

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

10.12.2.9 GetViewMatrix()

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

10.12.2.10 GetViewProjection()

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

10.12.2.11 GetYaw()

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

References [m_Yaw](#).

10.12.2.12 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 }
```

10.12.2.13 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\(\)](#).

10.12.2.14 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\(\)](#).

10.12.2.15 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\(\)](#).

10.12.2.16 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\(\)](#).

10.12.2.17 OnUpdate()

```

void Vesper::EditorCamera::OnUpdate (
    Timestep ts)
00047 {
00048     if (Input::IsKeyPressed(Key::LeftAlt))
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(Mouse::ButtonMiddle))
00055             MousePan(delta);
00056         else if (Input::IsMouseButtonPressed(Mouse::ButtonLeft))
00057             MouseRotate(delta);
00058         else if (Input::IsMouseButtonPressed(Mouse::ButtonRight))
00059             MouseZoom(-delta.y);
00060     }
00061
00062     UpdateView();
00063 }

```

References [Vesper::Mouse::ButtonLeft](#), [Vesper::Mouse::ButtonMiddle](#), [Vesper::Mouse::ButtonRight](#), [Vesper::Input::IsKeyPressed\(\)](#), [Vesper::Input::IsMouseButtonPressed\(\)](#), [Vesper::Key::LeftAlt](#), [MouseZoom\(\)](#), and [UpdateView\(\)](#).

10.12.2.18 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 }

```

10.12.2.19 RotationSpeed()

```

float Vesper::EditorCamera::RotationSpeed () const [private]
00033 {
00034     return 0.8f;
00035 }

```

Referenced by [MouseRotate\(\)](#).

10.12.2.20 SetDistance()

```
void Vesper::EditorCamera::SetDistance (
    float distance) [inline]
00021 { m_Distance = distance; }
```

References [m_Distance](#).

10.12.2.21 SetPitch()

```
void Vesper::EditorCamera::SetPitch (
    float pitch) [inline]
00037 { m_Pitch = pitch; }
```

References [m_Pitch](#).

10.12.2.22 SetPosition()

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

10.12.2.23 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\(\)](#).

10.12.2.24 SetYaw()

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

References [m_Yaw](#).

10.12.2.25 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\(\)](#).

10.12.2.26 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\(\)](#).

10.12.2.27 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\(\)](#).

10.12.3 Member Data Documentation**10.12.3.1 m_AspectRatio**

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

Referenced by [EditorCamera\(\)](#), and [UpdateProjection\(\)](#).

10.12.3.2 m_Distance

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

Referenced by [GetDistance\(\)](#), [MouseZoom\(\)](#), [SetDistance\(\)](#), and [ZoomSpeed\(\)](#).

10.12.3.3 m_FarClip

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

Referenced by [EditorCamera\(\)](#).

10.12.3.4 m_FocalPoint

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

10.12.3.5 m_FOV

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

Referenced by [EditorCamera\(\)](#).

10.12.3.6 m_InitialMousePosition

```
glm::vec2 Vesper::EditorCamera::m_InitialMousePosition = { 0.0f, 0.0f } [private]
00065 { 0.0f, 0.0f };
```

10.12.3.7 m_NearClip

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

Referenced by [EditorCamera\(\)](#).

10.12.3.8 m_Pitch

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

Referenced by [GetPitch\(\)](#), [MouseRotate\(\)](#), and [SetPitch\(\)](#).

10.12.3.9 m_Position

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

10.12.3.10 m_ViewMatrix

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

10.12.3.11 m_ViewportHeight

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

Referenced by [SetViewportSize\(\)](#), and [UpdateProjection\(\)](#).

10.12.3.12 m_ViewportWidth

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

Referenced by [SetViewportSize\(\)](#), and [UpdateProjection\(\)](#).

10.12.3.13 m_Yaw

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

Referenced by [GetYaw\(\)](#), [MouseRotate\(\)](#), and [SetYaw\(\)](#).

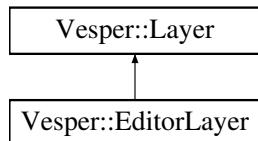
The documentation for this class was generated from the following files:

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

10.13 Vesper::EditorLayer Class Reference

```
#include <EditorLayer.h>
```

Inheritance diagram for Vesper::EditorLayer:



Public Member Functions

- [EditorLayer \(\)](#)
Called when the layer is attached to the application.
- virtual [~EditorLayer \(\)](#)=default
- virtual void [OnAttach \(\)](#) override
Called when the layer is attached to the application.
- virtual void [OnDetach \(\)](#) override
Called when the layer is detached from the application.
- virtual void [OnUpdate \(Timestep ts\)](#) override
Called every frame to update the layer with the given timestep.
- virtual void [OnImGuiRender \(\)](#) override
Called when the layer should render its ImGui components.
- virtual void [OnEvent \(Event &e\)](#) override
Called when an event is dispatched to the layer.

Public Member Functions inherited from Vesper::Layer

- [Layer \(const std::string &name="Layer"\)](#)
Constructs a Layer with an optional name for debugging purposes.
- virtual [~Layer \(\)](#)
- virtual void [OnRender \(\)](#)
Called when the layer should render its contents.
- const std::string & [GetName \(\)](#) const
Retrieves the name of the layer for debugging purposes.

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`
The name of the layer assigned at creation, used for debugging.

10.13.1 Member Enumeration Documentation

10.13.1.1 SceneState

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

Enumerator

| | |
|----------|--|
| Edit | |
| Play | |
| Simulate | |

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

10.13.2 Constructor & Destructor Documentation

10.13.2.1 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\(\)](#).

10.13.2.2 ~EditorLayer()

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

10.13.3 Member Function Documentation

10.13.3.1 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\(\)](#).

10.13.3.2 OnAttach()

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

Called when the layer is attached to the application.

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     {
00051         m_CheckerboardTexture =
00052             Texture2D::Create("../Vesper-Editor/assets/textures/Checkerboard.png");
00053         m_SpriteSheetFire =
00054             Texture2D::Create("../Vesper-Editor/assets/textures/sheets/fire_01.png");
00055         m_SpriteSheetSmoke =
00056             Texture2D::Create("../Vesper-Editor/assets/textures/sheets/fire_02.png");
00057         m_SpriteSheetTown =
00058             Texture2D::Create("../Vesper-Editor/assets/textures/sheets/town_tilesheet.png");
00059         m_SpriteSheetCrystals =
00060             Texture2D::Create("../Vesper-Editor/assets/textures/sheets/craftpix/Crystals/Crystals.png");
00061         m_SpriteSheetRocks =
00062             Texture2D::Create("../Vesper-Editor/assets/textures/sheets/craftpix/Rocks/Rocks_source.png");
00063         m_SpriteSheetCursedLands =
00064             Texture2D::Create("../Vesper-Editor/assets/textures/sheets/craftpix/CursedLand/Tiled_files/Objects.png");
00065
00066         m_SubTextureFire = SubTexture2D::CreateFromCoords(m_SpriteSheetFire, { 1, 0 }, { 128, 127 });
00067         m_SubTextureSmoke = SubTexture2D::CreateFromCoords(m_SpriteSheetSmoke, { 1, 0 }, { 128,
00068             127 });
00069         m_SubTextureTown = SubTexture2D::CreateFromCoords(m_SpriteSheetTown, { 4.25, 0.75 }, { 64,
00070             64 }, { 1, 1 });
00071         s_TextureMap['F'] = SubTexture2D::CreateFromCoords(m_SpriteSheetFire, { 1, 0 }, { 128, 127 });
00072         s_TextureMap['G'] = SubTexture2D::CreateFromCoords(m_SpriteSheetTown, { 4.25, 0.75 }, { 64,
00073             64 }, { 1, 1 });
00074         s_TextureMap['C'] = SubTexture2D::CreateFromCoords(m_SpriteSheetCrystals, { 0, 1.25 }, { 64,
00075             64 }, { 1, 1 });
00076         s_TextureMap['R'] = SubTexture2D::CreateFromCoords(m_SpriteSheetRocks, { 0, 3.75 }, { 64,
00077             64 }, { 1, 1 });
00078         s_TextureMap['P'] = SubTexture2D::CreateFromCoords(m_SpriteSheetCursedLands, { 0, 1.875 },
00079             { 128, 128 }, { 1, 1 });
00080     }
00081
00082     // Particle setup
00083 }
```

```

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, {
00127                 (float)y, (float)x }, { 128, 128 }));
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, {
00148                 (float)y, (float)x }, { 128, 128 }));
00149         }
00150         TextureAnimationComponent texAnim(smokeFrames, 0.05f);
00151         square.AddComponent<TextureAnimationComponent>(texAnim);
00152         m_SmokeEntity = square;
00153     }
00154
00155     auto quadEntity = m_ActiveScene->CreateEntity("Quad Entity");
00156     quadEntity.AddComponent<SpriteRendererComponent>(glm::vec4{ 0.2f, 0.3f, 0.8f, 1.0f });
00157

```

```

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,
00167             3.0f), Random::RangeF1(-3.0f, 3.0f), 0.0f));
00168     }
00169
00170         void OnDestroy()
00171     {
00172     }
00173
00174         void OnUpdate(Timestep ts)
00175     {
00176         auto& transform = GetComponent<TransformComponent>().GetTransform();
00177         float speed = 5.0f;
00178
00179
00180         if (Input::IsKeyPressed(VZ_KEY_A))
00181             transform[3][0] -= speed * ts;
00182
00183         if (Input::IsKeyPressed(VZ_KEY_D))
00184             transform[3][0] += speed * ts;
00185
00186         if (Input::IsKeyPressed(VZ_KEY_W))
00187             transform[3][1] += speed * ts;
00188
00189         if (Input::IsKeyPressed(VZ_KEY_S))
00190             transform[3][1] -= speed * ts;
00191     }
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) {
00208         std::string loadedScene = FileSystem::GetAbsolutePath("../.." +
00209             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: " +
00216                 FileSystem::GetCurrentWorkingDirectory());
00217             VZ_CORE_ERROR("Absolute Path Attempted: " +
00218                 FileSystem::GetAbsolutePath(loadedScene));
00219             VZ_CORE_ERROR("Error loading the scene, please check the paths and file
00220 availability.");
00221         }
00222         else {
00223             VZ_CORE_INFO("Successfully loaded the default scene: " + loadedScene);
00224         }
00225     }
00226
00227 }

```

References [Vesper::SceneSerializer::Deserialize\(\)](#), [Vesper::FramebufferSpecification::Height](#), [Vesper::FileSystem::Initialize\(\)](#), and [Vesper::FramebufferSpecification::Width](#).

10.13.3.3 OnDetach()

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

Called when the layer is detached from the application.

Reimplemented from [Vesper::Layer](#).

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

10.13.3.4 OnEvent()

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

Called when an event is dispatched to the layer.

Parameters

| | |
|--------------|-----------------------------|
| <i>event</i> | The event being dispatched. |
|--------------|-----------------------------|

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\(\)](#).

10.13.3.5 OnImGuiRender()

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

Called when the layer should render its ImGui components.

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
00435         // into,
00436         // because it would be confusing to have two docking targets within each others.
00437         ImGuiWindowFlags window_flags = ImGuiWindowFlags_MenuBar | ImGuiWindowFlags_NoDocking;
00438         if (opt_fullscreen)
00439         {
00440             const ImGuiViewport* viewport = ImGui::GetMainViewport();
00441             ImGui::SetNextWindowPos(viewport->WorkPos);
00442             ImGui::SetNextWindowSize(viewport->WorkSize);
00443             ImGui::SetNextWindowViewport(viewport->ID);
00444             ImGui::PushStyleVar(ImGuiStyleVar_WindowRounding, 0.0f);
00445             ImGui::PushStyleVar(ImGuiStyleVar_WindowBorderSize, 0.0f);
00446             window_flags |= ImGuiWindowFlags_NoTitleBar | ImGuiWindowFlags_NoCollapse |
00447                 ImGuiWindowFlags_NoResize | ImGuiWindowFlags_NoMove;
00448             window_flags |= ImGuiWindowFlags_NoBringToFrontOnFocus | ImGuiWindowFlags_NoNavFocus;
00449
00450             // When using ImGuiDockNodeFlags_PassThruCentralNode, DockSpace() will render our background
00451             // 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
00458     // docking, otherwise
00459     // any change of dockspace/settings would lead to windows being stuck in limbo and never being
00460     // visible.
00461     if (!opt_padding)
00462         ImGui::PushStyleVar(ImGuiStyleVar_WindowPadding, ImVec2(0.0f, 0.0f));
00463     ImGui::Begin("DockSpace Demo", &dockspaceOpen, window_flags);
00464     if (!opt_padding)
00465         ImGui::PopStyleVar();
00466
00467     if (opt_fullscreen)
00468         ImGui::PopStyleVar(2);
00469
00470     // Submit the DockSpace
00471     ImGuiIO& io = ImGui::GetIO();
00472     ImGuiStyle& style = ImGui::GetStyle();
00473     float minWinSize = style.WindowMinSize.x = 370.0f;
00474
00475     if (io.ConfigFlags & ImGuiConfigFlags_DockingEnable)
00476     {
00477         ImGuiID dockspace_id = ImGui::GetID("MyDockSpace");
00478         ImGui::DockSpace(dockspace_id, ImVec2(0.0f, 0.0f), dockspace_flags);
00479     }
00480
00481     style.WindowMinSize.x = minWinSize;
00482
00483     if (ImGui::BeginMenuBar())
00484     {
00485         if (ImGui::BeginMenu("File"))
00486         {
00487             // Disabling fullscreen would allow the window to be moved to the front of other
00488             // windows,
00489             // which we can't undo at the moment without finer window depth/z control.
00490
00491             if (ImGui::MenuItem("New", "Ctrl+N"))
00492                 NewScene();
00493
00494             if (ImGui::MenuItem("Open..", "Ctrl+O"))
00495                 OpenScene();
00496
00497             if (ImGui::MenuItem("Save As..", "Ctrl+Shift+S"))
00498                 SaveSceneAs();
00499
00500             if (ImGui::MenuItem("Reset Scene"))
00501                 ResetScene();
00502
00503             if (ImGui::MenuItem("Exit"))
00504                 Vesper::Application::Get().Close();
00505
00506             ImGui::EndMenu();
00507         }
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             }
00536         }
00537     }
00538 }
```

```

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     ImGui::End();
00553 }
00554
00555 m_SceneHierarchyPanel.OnImGuiRender();
00556
00557 {
00558     ImGui::Begin("Settings");
00559     ImGui::Text("Renderer2D Stats:");
00560     auto stats = Renderer2D::GetStats();
00561     ImGui::Text("\tDraw Calls: %d", stats.DrawCalls);
00562     ImGui::Text("\tQuad Count: %d", stats.QuadCount);
00563     ImGui::Text("\tVertex Count: %d", stats.GetTotalVertexCount());
00564     ImGui::Text("\tIndex Count: %d", stats.GetTotalIndexCount());
00565     ImGui::Text("Application Settings:");
00566     ImGui::Text("\tFPS: %.1f", ImGui::GetIO().Framerate);
00567     if (ImGui::ColorEdit4("Background Color", glm::value_ptr(m_ClearColor)))
00568     {
00569         RenderCommand::SetClearColor(m_ClearColor);
00570     }
00571     ImGui::End();
00572 }
00573
00574 DisplayVesperInfo_ImGui();
00575
00576 {
00577     ImGui::PushStyleVar(ImGuiStyleVar_WindowPadding, ImVec2{ 0,0 });
00578     ImGui::Begin("Viewport");
00579     m_VisitedFocused = ImGui::IsWindowFocused();
00580     m_VisitedHovered = ImGui::IsWindowHovered();
00581     Application::Get().GetImGuiLayer()->SetBlockEvents(!m_VisitedFocused &&
00582 !m_VisitedHovered);
00583     ImVec2 viewportPanelSize = ImGui::GetContentRegionAvail();
00584     if (m_VisitedSize != *(glm::vec2*)&viewportPanelSize) && viewportPanelSize.x > 0.0f &&
00585     viewportPanelSize.y > 0)
00586     {
00587         m_Framebuffer->Resize((uint32_t)viewportPanelSize.x, (uint32_t)viewportPanelSize.y);
00588         m_VisitedSize = { 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_VisitedSize.x, viewportBoundsMin.y +
00595 m_VisitedSize.y };
00596     m_VisitedBounds[0] = { viewportBoundsMin.x, viewportBoundsMin.y };
00597     m_VisitedBounds[1] = { viewportBoundsMax.x, viewportBoundsMax.y };
00598
00599     uint32_t textureID = m_Framebuffer->GetColorAttachmentRendererID();
00600     ImGui::Image(textureID, ImVec2(m_VisitedSize.x, m_VisitedSize.y), ImVec2(0, 1),
00601     ImVec2(1, 0));
00602
00603     // Gizmos
00604     Entity selectedEntity = m_SceneHierarchyPanel.GetSelectedEntity();
00605     if (selectedEntity && m_GizmoType != -1)
00606     {
00607         ImGuizmo::SetOrthographic(false);
00608         ImGuizmo::SetDrawlist();
00609         float windowHeight = (float)ImGui::GetWindowWidth();
00610         float windowWidth = (float)ImGui::GetWindowHeight();
00611         ImGuizmo::SetRect(ImGui::GetWindowPos().x, ImGui::GetWindowPos().y, windowWidth,
00612         windowHeight);
00613
00614     // Camera
00615     // Runtime camera from entity
00616     //auto cameraEntity = m_ActiveScene->GetPrimaryCameraEntity();
00617     //const auto& camera = cameraEntity.GetComponent<CameraComponent>().Camera;
00618     //const glm::mat4& cameraProjection = camera.GetProjection();
00619     //glm::mat4 cameraView =

```

```

00617     glm::inverse(cameraEntity.GetComponent<TransformComponent>().GetTransform());
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(Key::LeftControl);
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     ImGui::Manipulate(glm::value_ptr(cameraView), glm::value_ptr(cameraProjection),
00638                     (ImGui::OPERATION)m_GizmoType, ImGui::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::GetImGuiContext\(\)](#), [Vesper::Renderer2D::GetStats\(\)](#), [Vesper::Input::IsKeyPressed\(\)](#), [Vesper::Key::LeftControl](#), [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](#).

10.13.3.6 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(Key::LeftControl) ||
00676     Input::IsKeyPressed(Key::RightControl);
00677     bool shift = Input::IsKeyPressed(Key::LeftShift) || Input::IsKeyPressed(Key::RightShift);
00678     switch (e.GetKeyCode())
00679     {
00680         // Scene Shortcuts
00681         case Key::N:
00682             if (control)
00683             {
00684                 NewScene();
00685             }
00686             break;
00687
00688         case Key::O:
00689             if (control)
00690             {
00691                 OpenScene();
00692             }
00693             break;
00694
00695         case Key::S:
00696             if (control && shift)
```

```

00697         {
00698             SaveSceneAs();
00699         }
00700     break;
00701
00702
00703     // Gizmo Shortcuts
00704     case Key::Q:
00705         m_GizmoType = -1;
00706         break;
00707     case Key::W:
00708         m_GizmoType = ImGuizmo::OPERATION::TRANSLATE;
00709         break;
00710     case Key::E:
00711         m_GizmoType = ImGuizmo::OPERATION::ROTATE;
00712         break;
00713     case 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\(\)](#), [Vesper::Key::LeftControl](#), [Vesper::Key::LeftShift](#), [m_GizmoType](#), [Vesper::Key::N](#), [NewScene\(\)](#), [Vesper::Key::O](#), [OpenScene\(\)](#), [Vesper::Key::Q](#), [Vesper::Key::RightControl](#), [Vesper::Key::RightShift](#), [Vesper::Key::S](#), and [SaveSceneAs\(\)](#).

Referenced by [OnEvent\(\)](#).

10.13.3.7 OnUpdate()

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

Called every frame to update the layer with the given timestep.

Parameters

| | |
|-----------|---|
| <i>ts</i> | The timestep representing the time elapsed since the last update. |
|-----------|---|

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_VisualFocused)
00251         {
00252             m_CameraController.OnUpdate(ts);
00253         }
00254         m_EditorCamera.OnUpdate(ts);
00255
00256     // 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
00273     {
00274         // Basic scene
00275         if (scene1)
00276         {
00277             VZ_PROFILE_SCOPE("Scene 1");
00278             Renderer2D::BeginScene(m_CameraController.GetCamera());
00279
00280             // Checkerboard background
00281             Renderer2D::DrawQuadWithTexture({ 0.0f, 0.0f, -0.25f }, { 25.0f, 25.0f },
00282             m_CheckerboardTexture, 10.0f, m_BackgroundColor);
00283
00284             // Squares
00285             Renderer2D::DrawQuadRotated({ 0.0f, 1.25f, -0.165f }, { 1.0f, 1.0f },
00286             glm::radians(45.0f + m_squareRotation + rotation), m_SquareColor);
00287
00288             // Rotated Squares
00289             Renderer2D::DrawQuadRotatedWithTexture({ 0.0f, 1.25f, -0.15f }, { 0.75f, 0.75f },
00290             m_CheckerboardTexture, glm::radians(m_squareRotation * m_specialQuadRotation * rotation),
00291             m_textureScale, m_SpecialQuadColor);
00292
00293             Renderer2D::DrawQuadRotatedWithTexture({ 2.0f, -0.25f, -0.15f }, { 1.0f, 1.0f },
00294             m_CheckerboardTexture, glm::radians(m_squareRotation + rotation), m_textureScale,
00295             m_TextureTintColor1);
00296             Renderer2D::DrawQuadRotatedWithTexture({ -2.0f, -0.25f, -0.15f }, { 1.0f, 1.0f },
00297             m_CheckerboardTexture, glm::radians(m_squareRotation + rotation), m_textureScale,
00298             m_TextureTintColor2);
00299
00300             glm::vec3 startPos = { 0.0f, 0.0f, -0.175f };
00301             //Renderer2D::DrawQuad(pos, { 0.8f, 0.8f }, { 0.8f, 0.2f, 0.3f, 1.0f });
00302             glm::vec3 finalPos = startPos;
00303             float offset = 0.9f;
00304             for (int y = -10; y <= 10; y++)
00305             {
00306                 for (int x = -10; x <= 10; x++)
00307                 {
00308                     glm::vec3 newPos = { startPos.x - x * offset, startPos.y - y * offset,
00309                     startPos.z };
00310                     Renderer2D::DrawQuad(newPos, { 0.8f, 0.8f }, { (x + 5) / 10.0f, 0.4f, (y +
00311                     5) / 10.0f, 1.0f });
00312                     finalPos = newPos;
00313                 }
00314             }
00315             Renderer2D::DrawQuad(finalPos, { 0.8f, 0.8f }, { 0.8f, 0.2f, 0.3f, 1.0f });
00316             Renderer2D::EndScene();
00317
00318         }
00319
00320         // Sprite sheet drawn as full texture
00321         Renderer2D::DrawQuadWithTexture({ -1.0f, 1.5f, 0.5f }, { 1, 1 },
00322         m_SpriteSheetFire, 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00323         Renderer2D::DrawQuadRotatedWithTexture({ 1.5f, 0.0f, 0.0f }, { 1.78f, 1.0f },
00324         m_SpriteSheetTown, 0, 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00325
00326         // Sprite sheet drawn as full texture rotated
00327         Renderer2D::DrawQuadRotatedWithTexture({ -1.5f, 0.0f, 0.0f }, { 1.78f, 1.0f },
00328         m_SpriteSheetTown, glm::radians(-rotation), 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00329
00330         // Sub texture from tilesheet
00331         Renderer2D::DrawQuadWithTexture({ 2.0f, -1.5f, 0.0f }, { 1.0f, 1.0f },
00332         m_SubTextureTown, 1.0f, { 1.0f, 1.0f, 1.0f, 1.0f });
00333
00334         // Grid of sub textures from tilesheet
00335         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,
00337                         1.0f, glm::vec4(1.0f));
00338                 }
00339             }
00340             Renderer2D::DrawQuadRotatedWithTexture({ 0.0f, -1.5f, 0.0f }, { 1.0f, 1.0f },
00341             m_SubTextureFire, 0, 1.0f, { 1.0f, 1.0f, 1.0f });
00342         Renderer2D::EndScene();
00343     }
00344
00345     // Tile map scene
00346     if (scene3)
00347     {
00348         VZ_PROFILE_SCOPE("Scene 3");
00349         Renderer2D::BeginScene(m_CameraController.GetCamera());
00350         for (uint32_t y = 0; y < s_MapHeight; y++)
00351         {
00352             for (uint32_t x = 0; x < s_MapWidth; x++)
00353             {
00354                 char tileChar = s_MapTiles[x + y * s_MapWidth];
00355                 Ref<SubTexture2D> texture;
00356                 if (s_TextureMap.find(tileChar) != s_TextureMap.end())
00357                     texture = s_TextureMap[tileChar];
00358                 else
00359                     texture = s_TextureMap['G']; // Default to grass
00360
00361             Renderer2D::DrawQuadWithTexture({ x - s_MapWidth / 2.0f, s_MapHeight - y -
00362             s_MapHeight / 2.0f, 0.1f }, { 1.0f, 1.0f }, texture, 1.0f, glm::vec4(1.0f));
00363         }
00364     }
00365     Renderer2D::EndScene();
00366 }
00367
00368 // Particle scene
00369 if (scene4)
00370 {
00371     VZ_PROFILE_SCOPE("Scene 4");
00372     Renderer2D::BeginScene(m_CameraController.GetCamera());
00373     for (float y = -5.0f; y < 5.0f; y += 0.5f)
00374     {
00375         for (float x = -5.0f; x < 5.0f; x += 0.5f)
00376         {
00377             glm::vec4 color = { (x + 5.0f) / 10.0f, 0.4f, (y + 5.0f) / 10.0f, 0.35f };
00378             Renderer2D::DrawQuad({ x, y }, { 0.45f, 0.45f }, color);
00379         }
00380     }
00381
00382     if (Input::IsMouseButtonPressed(Mouse::ButtonLeft) && m_ViewportHovered)
00383     {
00384         ImVec2 mousePos = ImGui::GetMousePos();
00385
00386         mousePos.x -= m_ViewportBounds[0].x;
00387         mousePos.y -= m_ViewportBounds[0].y;
00388
00389         if (mousePos.x < 0 || mousePos.y < 0 || mousePos.x > m_ViewportSize.x ||
00390             mousePos.y > m_ViewportSize.y)
00391             return;
00392
00393         auto bounds = m_CameraController.GetBounds();
00394         auto camPos = m_CameraControllerGetPosition();
00395
00396         float width = m_ViewportSize.x;
00397         float height = m_ViewportSize.y;
00398
00399         m_ParticleProps.Position.x = (mousePos.x / width) * bounds.GetWidth() -
00400             bounds.GetWidth() * 0.5f + camPos.x;
00401         m_ParticleProps.Position.y = bounds.GetHeight() * 0.5f - (mousePos.y / height)
00402             * bounds.GetHeight() + camPos.y;
00403
00404         for (int i = 0; i < ParticleEmitCount; i++)
00405             m_ParticleSystem.Emit(m_ParticleProps);
00406     }
00407     m_ParticleSystem.OnUpdate(ts);
00408     m_ParticleSystem.OnRender(m_CameraController.GetCamera());
00409     Renderer2D::EndScene();
00410 }
00411
00412     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::Mouse::ButtonLeft](#), [Vesper::RenderCommand::Clear\(\)](#), [Vesper::Renderer2D::EndScene\(\)](#), [Vesper::Input::IsMouseButtonPressed\(\)](#), [m_ViewportFocused](#), [m_ViewportHovered](#), [ParticleEmitCount](#), [Vesper::Renderer2D::ResetScene\(\)](#), [s_MapHeight](#), [s_MapTiles](#), [s_MapWidth](#), [scene1](#), [scene2](#), [scene3](#), [scene4](#), and [useEntityScene](#).

10.13.3.8 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 serialized from: " + filePath);
00741     }
00742 }
```

References [Vesper::SceneSerializer::Deserialize\(\)](#).

Referenced by [OnImGuiRender\(\)](#), and [OnKeyPressed\(\)](#).

10.13.3.9 ResetScene()

```

void Vesper::EditorLayer::ResetScene () [private]
00756 {
00757     VZ_CORE_ASSERT(false, "Not implemented yet!");
00758 }
```

Referenced by [OnImGuiRender\(\)](#).

10.13.3.10 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\(\)](#).

10.13.4 Member Data Documentation

10.13.4.1 lastFrameTime

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

10.13.4.2 m_ActiveScene

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

10.13.4.3 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 };
```

10.13.4.4 m_CameraController

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

10.13.4.5 m_CameraEntity

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

10.13.4.6 m_CheckerboardTexture

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

10.13.4.7 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 };
```

10.13.4.8 m_EditorCamera

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

10.13.4.9 m_EditorScene

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

10.13.4.10 m_FireEntity

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

10.13.4.11 m_FlatColorShader

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

10.13.4.12 m_Framebuffer

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

10.13.4.13 m_GizmoType

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

Referenced by [OnImGuiRender\(\)](#), and [OnKeyPressed\(\)](#).

10.13.4.14 m_ParticleProps

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

10.13.4.15 m_ParticleSystem

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

10.13.4.16 m_PrimaryCamera

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

10.13.4.17 m_RotationSnap

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

Referenced by [OnImGuiRender\(\)](#).

10.13.4.18 m_ScaleSnap

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

Referenced by [OnImGuiRender\(\)](#).

10.13.4.19 m_SceneHierarchyPanel

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

10.13.4.20 m_SceneState

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

Referenced by [OnEvent\(\)](#).

10.13.4.21 m_SmokeEntity

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

10.13.4.22 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 };
```

10.13.4.23 m_specialQuadRotation

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

10.13.4.24 m_SpriteSheetCrystals

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

10.13.4.25 m_SpriteSheetCursedLands

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

10.13.4.26 m_SpriteSheetFire

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

10.13.4.27 m_SpriteSheetRocks

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

10.13.4.28 m_SpriteSheetSmoke

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

10.13.4.29 m_SpriteSheetTown

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

10.13.4.30 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 };
```

10.13.4.31 m_squareRotation

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

10.13.4.32 m_SquareVA

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

10.13.4.33 m_SubTextureFire

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

10.13.4.34 m_SubTextureSmoke

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

10.13.4.35 m_SubTextureTown

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

10.13.4.36 m_textureScale

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

10.13.4.37 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 };
```

10.13.4.38 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 };
```

10.13.4.39 m_TranslationSnap

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

Referenced by [OnImGuiRender\(\)](#).

10.13.4.40 m_UseSpecialQuadColor

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

10.13.4.41 m_ViewportBounds

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

10.13.4.42 m_ViewportFocused

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

Referenced by [OnImGuiRender\(\)](#), and [OnUpdate\(\)](#).

10.13.4.43 m_ViewportHovered

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

Referenced by [OnImGuiRender\(\)](#), and [OnUpdate\(\)](#).

10.13.4.44 m_ViewportSize

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

10.13.4.45 ParticleEmitCount

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

Referenced by [OnUpdate\(\)](#).

10.13.4.46 s_TextureMap

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

10.13.4.47 scene1

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

Referenced by [OnImGuiRender\(\)](#), and [OnUpdate\(\)](#).

10.13.4.48 scene2

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

Referenced by [OnImGuiRender\(\)](#), and [OnUpdate\(\)](#).

10.13.4.49 scene3

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

Referenced by [OnImGuiRender\(\)](#), and [OnUpdate\(\)](#).

10.13.4.50 scene4

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

Referenced by [OnImGuiRender\(\)](#), and [OnUpdate\(\)](#).

10.13.4.51 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](#)

10.14 Vesper::Entity Class Reference

Represents an entity in a scene.

```
#include <Entity.h>
```

Public Member Functions

- `Entity ()=default`
- `Entity (entt::entity handle, Scene *scene)`
Constructs an Entity with the given handle and scene.
- `Entity (const Entity &other)=default`
Copy constructor for Entity.
- 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}`
The unique identifier of the entity within the scene.
- `Scene * m_Scene = nullptr`
Pointer to the scene that contains the entity.

10.14.1 Detailed Description

Represents an entity in a scene.

Contains a handle to an entity (entt::entity) in a scene and provides methods to manipulate its components.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 Entity() [1/3]

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

10.14.2.2 Entity() [2/3]

```
Vesper::Entity::Entity (
    entt::entity handle,
    Scene * scene)
```

Constructs an [Entity](#) with the given handle and scene.

Parameters

| | |
|---------------|--|
| <i>handle</i> | The handle to the entity. |
| <i>scene</i> | The scene to which the entity belongs. |

```
00007      : m_EntityID(handle), m_Scene(scene)
00008  {
00009 }
```

References [Entity\(\)](#), and [m_Scene](#).

Referenced by [Entity\(\)](#).

10.14.2.3 Entity() [3/3]

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

Copy constructor for [Entity](#).

10.14.3 Member Function Documentation

10.14.3.1 AddComponent()

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

Adds a component of type T to the entity with the provided arguments Otherwise, asserts if the entity already has the component.

Template Parameters

| | |
|-------------|---|
| <i>T</i> | The type of component to add. |
| <i>Args</i> | The types of arguments to forward to the component's constructor. |

Parameters

| | |
|-------------|--|
| <i>args</i> | The arguments to forward to the component's constructor. |
|-------------|--|

Returns

A reference to the newly added component.

```
00046      {
00047          VZ_CORE_ASSERT(!HasComponent<T>(), "Entity already has component!");
00048          T& component = m_Scene->m_Registry.emplace<T>(m_EntityID, std::forward<Args>(args)...);
00049          m_Scene->OnComponentAdded<T>(*this, component);
00050          return component;
00051      }
```

10.14.3.2 AddOrReplaceComponent()

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

Adds or replaces a component of type T to the entity with the provided arguments.

Template Parameters

| | |
|-------------|---|
| <i>T</i> | The type of component to add or replace. |
| <i>Args</i> | The types of arguments to forward to the component's constructor. |

Parameters

| | |
|-------------|--|
| <i>args</i> | The arguments to forward to the component's constructor. |
|-------------|--|

Returns

A reference to the newly added or replaced component.

```
00061      {
00062          return m_Scene->m_Registry.emplace_or_replace<T>(m_EntityID, std::forward<Args>(args)...);
00063      }
```

10.14.3.3 GetComponent()

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

Retrieves a reference to the component of type T attached to the entity if it exists. Otherwise, asserts.

Template Parameters

| | |
|----------|------------------------------------|
| <i>T</i> | The type of component to retrieve. |
|----------|------------------------------------|

Returns

A reference to the component.

```
00071      {
00072          VZ_CORE_ASSERT(HasComponent<T>(), "Entity does not have component!");
00073          return m_Scene->m_Registry.get<T>(m_EntityID);
00074      }
```

10.14.3.4 GetID()

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

Retrieves a const reference to the [UUID](#) of the entity.

Returns

A const reference to the [UUID](#) of the entity.

```
00105      {
00106          return GetComponent<UUIDComponent>().ID;
00107      }
```

10.14.3.5 GetName()

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

Retrieves a const reference to the name of the entity.

Returns

A const reference to the name of the entity.

```
00113     {
00114         return GetComponent<NameComponent>().Name;
00115     }
```

10.14.3.6 GetOrAddComponent()

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

Retrieves a reference to the component of type T attached to the entity if it exists. Otherwise, adds the component with the provided arguments and returns it.

Template Parameters

| | |
|-------------|---|
| <i>T</i> | The type of component to add. |
| <i>Args</i> | The types of arguments to forward to the component's constructor. |

Parameters

| | |
|-------------|--|
| <i>args</i> | The arguments to forward to the component's constructor. |
|-------------|--|

Returns

A reference to the newly added component.

```
00084     {
00085         if (HasComponent<T>())
00086             return GetComponent<T>();
00087         else
00088             return AddComponent<T>(std::forward<Args>(args)...);
00089     }
```

10.14.3.7 HasComponent()

```
template<typename T>
bool Vesper::Entity::HasComponent () const [inline]
```

Checks if the entity has a component of type T.

Template Parameters

| | |
|----------|-------------------------------------|
| <i>T</i> | The type of component to check for. |
|----------|-------------------------------------|

Returns

true if the entity has the component, false otherwise.

```
00034     {  
00035         return m_Scene->m_Registry.all_of<T>(m_EntityID);  
00036     }
```

10.14.3.8 operator bool()

```
Vesper::Entity::operator bool () const [inline]  
00118 { return m_EntityID != entt::null; }
```

10.14.3.9 operator entt::entity()

```
Vesper::Entity::operator entt::entity () const [inline]  
00119 { return m_EntityID; }
```

10.14.3.10 operator uint32_t()

```
Vesper::Entity::operator uint32_t () const [inline]  
00120 { return (uint32_t)m_EntityID; }
```

10.14.3.11 operator"!=()

```
bool Vesper::Entity::operator!= (  
    const Entity & other) const [inline]  
00122 { return !(this == other); }
```

10.14.3.12 operator==()

```
bool Vesper::Entity::operator== (  
    const Entity & other) const [inline]  
00121 { return m_EntityID == other.m_EntityID && m_Scene == other.m_Scene; }
```

10.14.3.13 RemoveComponent()

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

Removes the component of type T from the entity if it exists. Otherwise, asserts.

Template Parameters

| | |
|----------|----------------------------------|
| <i>T</i> | The type of component to remove. |
|----------|----------------------------------|

```
00096      {
00097          VZ_CORE_ASSERT(HasComponent<T>(), "Entity does not have component!");
00098          m_Scene->m_Registry.remove<T>(m_EntityID);
00099      }
```

10.14.4 Member Data Documentation

10.14.4.1 m_EntityID

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

The unique identifier of the entity within the scene.

```
00126 {entt::null};
```

10.14.4.2 m_Scene

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

Pointer to the scene that contains the entity.

Referenced by [Entity\(\)](#).

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

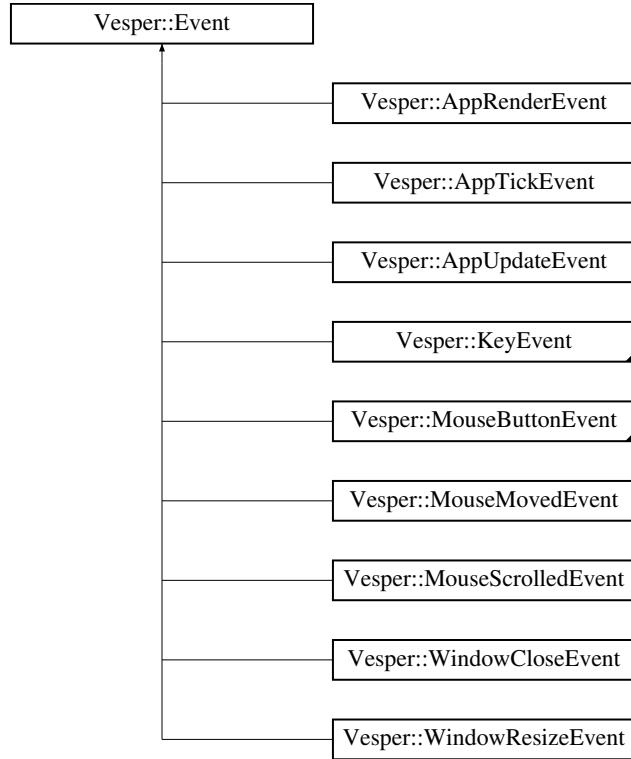
- Vesper/src/Vesper/Scene/[Entity.h](#)
- Vesper/src/Vesper/Scene/[Entity.cpp](#)

10.15 Vesper::Event Class Reference

Abstract base class for all events.

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

Inheritance diagram for Vesper::Event:



Public Member Functions

- virtual `~Event ()=default`
- virtual `EventType GetEventType () const =0`
Get the type of the event.
- virtual const char * `GetName () const =0`
Get the name of the event.
- virtual int `GetCategoryFlags () const =0`
Get the category flags of the event.
- virtual std::string `ToString () const`
Convert the event to a string representation.
- bool `IsInCategory (EventCategory category)`
Check if the event is in a specific category.

Public Attributes

- bool `Handled = false`
Indicates whether the event has been handled.

Friends

- class `EventDispatcher`

10.15.1 Detailed Description

Abstract base class for all events.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 ~Event()

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

10.15.3 Member Function Documentation

10.15.3.1 GetCategoryFlags()

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

Get the category flags of the event.

Referenced by [IsInCategory\(\)](#).

10.15.3.2 GetEventType()

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

Get the type of the event.

10.15.3.3 GetName()

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

Get the name of the event.

Referenced by [ToString\(\)](#).

10.15.3.4 IsInCategory()

```
bool Vesper::Event::IsInCategory (
    EventType category) [inline]
```

Check if the event is in a specific category.

```
00056     {
00057         return GetCategoryFlags() & category;
00058     }
```

References [GetCategoryFlags\(\)](#).

Referenced by [Vesper::ImGuiLayer::OnEvent\(\)](#).

10.15.3.5 ToString()

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

Convert the event to a string representation.

Reimplemented in [Vesper::KeyPressedEvent](#), [Vesper::KeyReleasedEvent](#), [Vesper::KeyTypedEvent](#), [Vesper::MouseButtonPressedEvent](#), [Vesper::MouseButtonReleasedEvent](#), [Vesper::MouseMovedEvent](#), [Vesper::MouseScrolledEvent](#), and [Vesper::WindowResizeEvent](#).
00052 { return GetName(); }

References [GetName\(\)](#).

10.15.4 Friends And Related Symbol Documentation

10.15.4.1 EventDispatcher

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

10.15.5 Member Data Documentation

10.15.5.1 Handled

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

Indicates whether the event has been handled.

Referenced by [Vesper::ImGuiLayer::OnEvent\(\)](#).

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

- Vesper/src/Vesper/Events/[Event.h](#)

10.16 Vesper::EventDispatcher Class Reference

Stack-based templated event dispatcher.

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

Public Member Functions

- [EventDispatcher \(Event &event\)](#)
Construct an [EventDispatcher](#) for a specific event.
- [template<typename T> bool Dispatch \(EventFn< T > func\)](#)
Dispatch the event to the appropriate handler if the types match.

Private Types

- template<typename T>
using [EventFn](#) = std::function<bool(T&)>
Type alias for event handling functions.

Private Attributes

- [Event](#) & [m_Event](#)
The event to be dispatched.

10.16.1 Detailed Description

Stack-based templated event dispatcher.

10.16.2 Member Typedef Documentation

10.16.2.1 [EventFn](#)

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

Type alias for event handling functions.

10.16.3 Constructor & Destructor Documentation

10.16.3.1 [EventDispatcher\(\)](#)

```
Vesper::EventDispatcher::EventDispatcher (
    Event & event) [inline]
```

Construct an [EventDispatcher](#) for a specific event.

Parameters

| | |
|-----------------------|------------------------|
| event | The event to dispatch. |
|-----------------------|------------------------|

```
00079      : m\_Event(event)
00080      {
00081      }
```

References [m_Event](#).

Referenced by [Vesper::Application::OnEvent\(\)](#), [Vesper::EditorCamera::OnEvent\(\)](#), [Vesper::EditorLayer::OnEvent\(\)](#), and [Vesper::OrthographicCameraController::OnEvent\(\)](#).

10.16.4 Member Function Documentation

10.16.4.1 Dispatch()

```
template<typename T>
bool Vesper::EventDispatcher::Dispatch (
    EventFn< T > func) [inline]
```

Dispatch the event to the appropriate handler if the types match.

Template Parameters

| | |
|----------|------------------------------------|
| <i>T</i> | The type of the event to dispatch. |
|----------|------------------------------------|

Parameters

| | |
|-------------|-----------------------------------|
| <i>func</i> | The function to handle the event. |
|-------------|-----------------------------------|

Returns

True if the event was handled, false otherwise.

```
00090     {
00091         if (m_Event.GetEventType() == T::GetStaticType())
00092         {
00093             m_Event.Handled = func(* (T*) &m_Event);
00094             return true;
00095         }
00096         return false;
00097     }
```

10.16.5 Member Data Documentation

10.16.5.1 m_Event

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

The event to be dispatched.

Referenced by [EventDispatcher\(\)](#).

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

- Vesper/src/Vesper/Events/[Event.h](#)

10.17 Vesper::FileDialogs Class Reference

Cross-platform file dialog utilities.

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

Static Public Member Functions

- static std::string [OpenFile](#) (const char *filter)
Opens a file dialog to select a file to open.
- static std::string [SaveFile](#) (const char *filter)
Opens a file dialog to select a location to save a file.

10.17.1 Detailed Description

Cross-platform file dialog utilities.

10.17.2 Member Function Documentation

10.17.2.1 OpenFile()

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

Opens a file dialog to select a file to open.

Parameters

| | |
|---------------|--------------------------------------|
| <i>filter</i> | The file type filter for the dialog. |
|---------------|--------------------------------------|

Returns

The selected file path or an empty string if cancelled.

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

10.17.2.2 SaveFile()

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

Opens a file dialog to select a location to save a file.

Parameters

| | |
|---------------|--------------------------------------|
| <i>filter</i> | The file type filter for the dialog. |
|---------------|--------------------------------------|

Returns

The selected file path or an empty string if cancelled.

```

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 =
00037         glfwGetWin32Window((GLFWwindow*)Application::Get().GetWindow().GetNativeWindow());
00038     ofn.lpstrFile = szFile;
00039     ofn.nMaxFile = sizeof(szFile);
00040     if (GetCurrentDirectoryA(256, currentDir))
00041         ofn.lpstrInitialDir = currentDir;
00042     ofn.lpstrFilter = filter;
00043     ofn.nFilterIndex = 1;
00044     ofn.Flags = OFN_PATHMUSTEXIST | OFN_OVERWRITEPROMPT | OFN_NOCHANGEDIR;
00045
00046     // Sets the default extension by extracting it from the filter
00047     ofn.lpstrDefExt = strchr(filter, '\0') + 1;
00048
00049     if (GetSaveFileNameA(&ofn) == TRUE)
00050         return ofn.lpstrFile;
00051
00052     return std::string();
00053 }
```

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

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

10.18 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](#) = ""

10.18.1 Member Function Documentation

10.18.1.1 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\(\)](#).

10.18.1.2 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 }
```

10.18.1.3 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\(\)](#).

10.18.1.4 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\(\)](#).

10.18.1.5 IsInitialized()

```

bool Vesper::FileSystem::IsInitialized () [inline], [static]
00034 { return m_Initialized; }
```

References [m_Initialized](#).

10.18.2 Member Data Documentation

10.18.2.1 m_AssetsDirectory

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

10.18.2.2 m_CurrentProjectDirectory

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

10.18.2.3 m_Initialized

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

Referenced by [Initialize\(\)](#), and [IsInitialized\(\)](#).

10.18.2.4 m_ProjectsDirectory

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

10.18.2.5 m_ResourcesDirectory

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

10.18.2.6 m_RootEditorDirectory

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

10.18.2.7 m_RootEngineDirectory

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

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

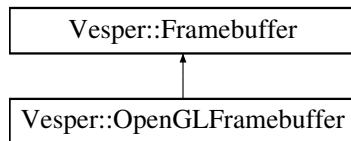
- Vesper/src/Vesper/Utils/[PlatformUtils.h](#)
- Vesper/src/Platform/Windows/[WindowsPlatformUtils.cpp](#)

10.19 Vesper::Framebuffer Class Reference

Abstract class representing a framebuffer.

```
#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`
Resizes the framebuffer to the given width and height.
- `virtual uint32_t GetColorAttachmentRendererID () const =0`
Returns the renderer ID of the color attachment texture.
- `virtual const FramebufferSpecification & GetSpecification () const =0`
Returns the specification used to create the framebuffer.

Static Public Member Functions

- `static Ref< Framebuffer > Create (const FramebufferSpecification &spec)`
Creates a framebuffer with the given specification.

10.19.1 Detailed Description

Abstract class representing a framebuffer.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 ~Framebuffer()

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

10.19.3 Member Function Documentation

10.19.3.1 Bind()

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

Implemented in [Vesper::OpenGLFramebuffer](#).

10.19.3.2 Create()

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

Creates a framebuffer with the given specification.

Parameters

| | |
|-------------|--|
| <i>spec</i> | The specification for the framebuffer. |
|-------------|--|

Returns

A reference-counted pointer to the created framebuffer.

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

References [Vesper::Renderer::GetAPI\(\)](#), [Vesper::RendererAPI::None](#), and [Vesper::RendererAPI::OpenGL](#).

10.19.3.3 GetColorAttachmentRendererID()

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

Returns the renderer ID of the color attachment texture.

Implemented in [Vesper::OpenGLFramebuffer](#).

10.19.3.4 GetSpecification()

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

Returns the specification used to create the framebuffer.

Implemented in [Vesper::OpenGLFramebuffer](#).

10.19.3.5 Resize()

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

Resizes the framebuffer to the given width and height.

Implemented in [Vesper::OpenGLFramebuffer](#).

10.19.3.6 Unbind()

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

Implemented in [Vesper::OpenGLFramebuffer](#).

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

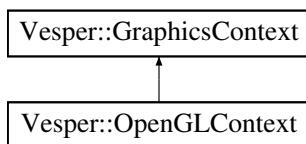
- [Vesper/src/Vesper/Renderer/Framebuffer.h](#)
- [Vesper/src/Vesper/Renderer/Framebuffer.cpp](#)

10.20 Vesper::GraphicsContext Class Reference

Abstract class representing a graphics context.

```
#include <GraphicsContext.h>
```

Inheritance diagram for Vesper::GraphicsContext:



Public Member Functions

- virtual [~GraphicsContext](#) ()
- virtual void [Init](#) ()=0
Initializes the graphics context.
- virtual void [SwapBuffers](#) ()=0
Swaps the front and back buffers.

10.20.1 Detailed Description

Abstract class representing a graphics context.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 ~GraphicsContext()

```
virtual Vesper::GraphicsContext::~GraphicsContext () [inline], [virtual]  
00012 {}
```

10.20.3 Member Function Documentation

10.20.3.1 Init()

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

Initializes the graphics context.

Implemented in [Vesper::OpenGLContext](#).

Referenced by [Vesper::WindowsWindow::Init\(\)](#).

10.20.3.2 SwapBuffers()

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

Swaps the front and back buffers.

Implemented in [Vesper::OpenGLContext](#).

Referenced by [Vesper::WindowsWindow::OnUpdate\(\)](#).

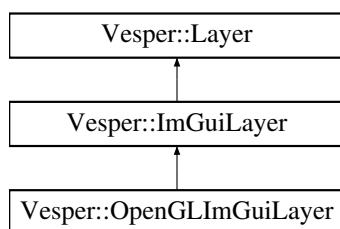
The documentation for this class was generated from the following file:

- [Vesper/src/Vesper/Renderer/GraphicsContext.h](#)

10.21 Vesper::ImGuiLayer Class Reference

```
#include <ImGuiLayer.h>
```

Inheritance diagram for Vesper::ImGuiLayer:



Public Member Functions

- `ImGuiLayer()`
- `~ImGuiLayer()`
- virtual void `OnAttach()` override
Called when the layer is attached to the application.
- virtual void `OnDetach()` override
Called when the layer is detached from the application.
- virtual void `OnImGuiRender()` override
Called when the layer should render its ImGui components.
- virtual void `OnEvent(Event &e)` override
Called when an event is dispatched to the layer.
- 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")`
Constructs a `Layer` with an optional name for debugging purposes.
- virtual `~Layer()`
- virtual void `OnUpdate(Timestep ts)`
Called every frame to update the layer with the given timestep.
- virtual void `OnRender()`
Called when the layer should render its contents.
- const std::string & `GetName()` const
Retrieves the name of the layer for debugging purposes.

Protected Attributes

- bool `m_BlockEvents` = true
- float `m_Time` = 0.0f

Protected Attributes inherited from `Vesper::Layer`

- std::string `m_DebugName`
The name of the layer assigned at creation, used for debugging.

10.21.1 Constructor & Destructor Documentation

10.21.1.1 `ImGuiLayer()`

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

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

10.21.1.2 ~ImGuiLayer()

```
Vesper::ImGuiLayer::~ImGuiLayer ()  
00026 {  
00027 }
```

10.21.2 Member Function Documentation

10.21.2.1 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\(\)](#).

10.21.2.2 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_VideoPortsEnable)  
     {  
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::GetWindow\(\)](#).

Referenced by [Vesper::OpenGLImGuiLayer::End\(\)](#), and [Vesper::Application::Run\(\)](#).

10.21.2.3 OnAttach()

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

Called when the layer is attached to the application.

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
00039          Windows
00039          //io.ConfigFlags |= ImGuiConfigFlags_ViewportsNoTaskBarIcons; // Disable Platform Windows task
00040          bar icons
00040          //io.ConfigFlags |= ImGuiConfigFlags_ViewportsNoMerge;        // Disable Platform Windows
00041          merging into host window
00042
00042          io.Fonts->AddFontFromFileTTF("../Vesper-Editor/assets/fonts/RedHatMono/static/RedHatMono-Bold.ttf",
00042          18.0f);
00043          io.FontDefault =
00043          io.Fonts->AddFontFromFileTTF("../Vesper-Editor/assets/fonts/RedHatMono/static/RedHatMono-Light.ttf",
00043          18.0f);
00044
00045
00046          ImGui::StyleColorsDark();
00047
00048          // When viewports are enabled we tweak WindowRounding/WindowBg so platform windows can look
00048          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
00059          {
00060              Application& app = Application::Get();
00061              GLFWwindow* window = static_cast<GLFWwindow*>(app.GetWindow().GetNativeWindow());
00062              ImGui_ImplGlfw_InitForOpenGL(window, true);
00063              ImGui_ImplOpenGL3_Init("#version 410");
00064
00065          }
00066
00067      }
```

References [Vesper::Application::Get\(\)](#), and [SetDarkThemeColors\(\)](#).

Referenced by [Vesper::OpenGLImGuiLayer::OnAttach\(\)](#).

10.21.2.4 OnDetach()

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

Called when the layer is detached from the application.

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\(\)](#).

10.21.2.5 OnEvent()

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

Called when an event is dispatched to the layer.

Parameters

| | |
|--------------|-----------------------------|
| <i>event</i> | The event being dispatched. |
|--------------|-----------------------------|

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\(\)](#).

10.21.2.6 OnImGuiRender()

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

Called when the layer should render its ImGui components.

Reimplemented from [Vesper::Layer](#).

Reimplemented in [Vesper::OpenGLImGuiLayer](#).

```
00082      {
00083
00084  }
```

Referenced by [Vesper::OpenGLImGuiLayer::OnImGuiRender\(\)](#).

10.21.2.7 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\(\)](#).

10.21.2.8 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\(\)](#).

10.21.3 Member Data Documentation

10.21.3.1 m_BlockEvents

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

Referenced by [OnEvent\(\)](#), [SetBlockEvents\(\)](#), and [Vesper::OpenGLImGuiLayer::SetBlockEvents\(\)](#).

10.21.3.2 m_Time

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

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

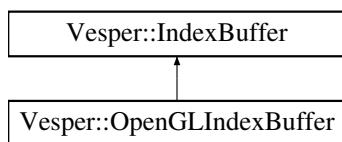
- [Vesper/src/Vesper/ImGui/ImGuiLayer.h](#)
- [Vesper/src/Vesper/ImGui/ImGuiLayer.cpp](#)

10.22 Vesper::IndexBuffer Class Reference

Abstract base class for an index buffer.

```
#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\(uint32_t *indices, uint32_t count\)](#)

10.22.1 Detailed Description

Abstract base class for an index buffer.

Currently only supports uint32_t indices

10.22.2 Constructor & Destructor Documentation

10.22.2.1 ~IndexBuffer()

```
virtual Vesper::IndexBuffer::~IndexBuffer () [inline], [virtual]  
00156 {}
```

10.22.3 Member Function Documentation

10.22.3.1 Bind()

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

Implemented in [Vesper::OpenGLIndexBuffer](#).

10.22.3.2 Create()

```
Ref< IndexBuffer > Vesper::IndexBuffer::Create (
    uint32_t * indices,
    uint32_t count) [static]

00034 {
00035     switch (Renderer::GetAPI())
00036     {
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](#).

10.22.3.3 GetCount()

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

Implemented in [Vesper::OpenGLIndexBuffer](#).

10.22.3.4 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](#)

10.23 Vesper::Input Class Reference

Base input class for querying input states.

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

Public Member Functions

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

Static Public Member Functions

- static bool `IsKeyPressed (int keycode)`
Checks if the specified key is currently pressed.
- static bool `IsMouseButtonPressed (int button)`
Checks if the specified mouse button is currently pressed.
- static float `GetMouseX ()`
Gets the current X position of the mouse cursor.
- static float `GetMouseY ()`
Gets the current Y position of the mouse cursor.
- static glm::vec2 `GetMousePosition ()`
Gets the current position of the mouse cursor as a 2D vector.

Protected Member Functions

- `Input ()=default`

10.23.1 Detailed Description

Base input class for querying input states.

10.23.2 Constructor & Destructor Documentation**10.23.2.1 Input() [1/2]**

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

10.23.2.2 Input() [2/2]

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

10.23.3 Member Function Documentation**10.23.3.1 GetMousePosition()**

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

Gets the current position of the mouse cursor as a 2D vector.

```
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      }
```

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

10.23.3.2 GetMouseX()

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

Gets the current X position of the mouse cursor.

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

10.23.3.3 GetMouseY()

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

Gets the current Y position of the mouse cursor.

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

10.23.3.4 IsKeyPressed()

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

Checks if the specified key is currently pressed.

Parameters

| | |
|----------------------|----------------------------------|
| <code>keycode</code> | The keycode of the key to check. |
|----------------------|----------------------------------|

Returns

True if the key is pressed, false otherwise.

```
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\(\)](#).

10.23.3.5 IsMouseButtonPressed()

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

Checks if the specified mouse button is currently pressed.

Parameters

| | |
|---------------|----------------------------|
| <i>button</i> | The mouse button to check. |
|---------------|----------------------------|

Returns

True if the mouse button is pressed, false otherwise.

```
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\(\)](#).

10.23.3.6 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](#)

10.24 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](#)

10.24.1 Constructor & Destructor Documentation

10.24.1.1 InstrumentationTimer()

```
Vesper::InstrumentationTimer::InstrumentationTimer (
    const char * name) [inline]
00149         : m_Name(name), m_Stopped(false)
00150     {
00151         m_StartTimepoint = std::chrono::high_resolution_clock::now();
00152     }
```

10.24.1.2 ~InstrumentationTimer()

```
Vesper::InstrumentationTimer::~InstrumentationTimer () [inline]
00155     {
00156         if (!m_Stopped)
00157             Stop();
00158     }
```

10.24.2 Member Function Documentation

10.24.2.1 Stop()

```
void Vesper::InstrumentationTimer::Stop () [inline]
00161     {
00162         auto endTimepoint = std::chrono::high_resolution_clock::now();
00163
00164         long long start =
00165             std::chrono::time_point_cast<std::chrono::microseconds>(m_StartTimepoint).time_since_epoch().count();
00166         long long end =
00167             std::chrono::time_point_cast<std::chrono::microseconds>(endTimepoint).time_since_epoch().count();
00168
00169         uint32_t threadID = std::hash<std::thread::id>{}(std::this_thread::get_id());
00170         Instrumentor::Get().WriteProfile({ m_Name, start, end, threadID });
00171         m_Stopped = true;
00171     }
```

10.24.3 Member Data Documentation

10.24.3.1 m_Name

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

10.24.3.2 m_StartTimepoint

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

10.24.3.3 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](#)

10.25 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`

10.25.1 Constructor & Destructor Documentation

10.25.1.1 Instrumentor()

```
Vesper::Instrumentor::Instrumentor () [inline]
00054     : m_CurrentSession(nullptr)
00055     {
00056 }
```

10.25.2 Member Function Documentation

10.25.2.1 BeginSession()

```
void Vesper::Instrumentor::BeginSession (
    const std::string & name,
    const std::string & filepath = "results.json") [inline]
00059     {
00060         std::lock_guard lock(m_Mutex);
00061         if (m_CurrentSession) {
00062             // If there is already a current session, end it and start new one
00063             if (Log::GetCoreLogger())
00064                 {
00065                     VZ_CORE_ERROR("Instrumentor::BeginSession('{0}') when session '{1}' already
open.", name, m_CurrentSession->Name);
00066                 }
00067             InternalEndSession();
00068         }
00069         m_OutputStream.open(filepath);
00070         if (!m_OutputStream.is_open()) {
00071             m_CurrentSession = new InstrumentationSession{ name };
00072             WriteHeader();
00073         }
00074         else {
00075             if (Log::GetCoreLogger())
00076             {
00077                 VZ_CORE_ERROR("Instrumentor could not open results file '{0}'.", filepath);
00078             }
00079         }
00080         WriteHeader();
00081         m_CurrentSession = new InstrumentationSession{ name };
00082     }
```

10.25.2.2 EndSession()

```
void Vesper::Instrumentor::EndSession () [inline]
00085     {
00086         std::lock_guard lock(m_Mutex);
00087         InternalEndSession();
00088     }
```

10.25.2.3 Get()

```
Instrumentor & Vesper::Instrumentor::Get () [inline], [static]
00139     {
00140         static Instrumentor instance;
00141         return instance;
00142     }
```

10.25.2.4 InternalEndSession()

```
void Vesper::Instrumentor::InternalEndSession () [inline]
00128     {
00129         if (m_CurrentSession)
00130         {
00131             WriteFooter();
00132             m_OutputStream.close();
00133             delete m_CurrentSession;
00134             m_CurrentSession = nullptr;
00135         }
00136     }
```

10.25.2.5 WriteFooter()

```
void Vesper::Instrumentor::WriteFooter () [inline]
00122     {
00123         m_OutputStream << "}]";
00124         m_OutputStream.flush();
00125     }
```

10.25.2.6 WriteHeader()

```
void Vesper::Instrumentor::WriteHeader () [inline]
00116     {
00117         m_OutputStream << "{\"otherData\": {}, \"traceEvents\": [";
00118         m_OutputStream.flush();
00119     }
```

10.25.2.7 WriteProfile()

```
void Vesper::Instrumentor::WriteProfile (
    const ProfileResult & result) [inline]
00091     {
00092         m_OutputStream << ",";
00093         std::string name = result.Name;
00094         std::replace(name.begin(), name.end(), '\"', '\\\"');
00095         m_OutputStream << "{";
00096         m_OutputStream << "\"cat\":\"function\",";
00097         m_OutputStream << "\"dur\":(" << (result.End - result.Start) << ",";
00098         m_OutputStream << "\"name\":\"" << name << "\",";
00099     }
```

```

00101     m_OutputStream << "\\"ph\":\"X\", ";
00102     m_OutputStream << "\\"pid\":0, ";
00103     m_OutputStream << "\\"tid\":" << result.ThreadID << ",";
00104     m_OutputStream << "\\"ts\":" << result.Start;
00105     m_OutputStream << "}";
00106
00107     std::lock_guard lock(m_Mutex);
00108     if (m_CurrentSession)
00109     {
00110         //m_OutputStream << json.str();
00111         m_OutputStream.flush();
00112     }
00113 }
```

10.25.3 Member Data Documentation

10.25.3.1 m_CurrentSession

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

10.25.3.2 m_Mutex

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

10.25.3.3 m_OutputStream

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

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

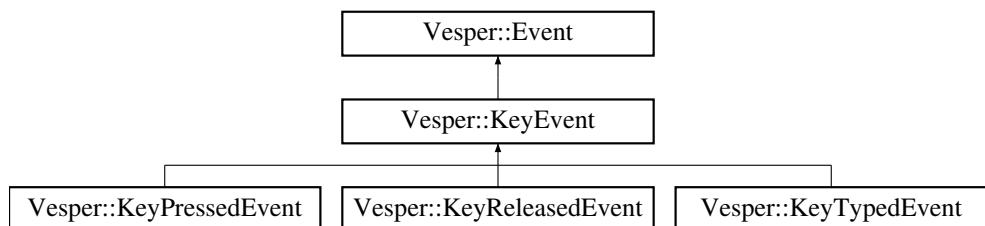
- `Vesper/src/Vesper/Debug/Instrumentor.h`

10.26 Vesper::KeyEvent Class Reference

Base class for keyboard events.

`#include <KeyEvent.h>`

Inheritance diagram for Vesper::KeyEvent:



Public Member Functions

- `int GetKeyCode () const`
Get the key code associated with the event.

Public Member Functions inherited from [Vesper::Event](#)

- virtual [~Event](#) ()=default
- virtual [EventType GetEventType](#) () const =0
Get the type of the event.
- virtual const char * [GetName](#) () const =0
Get the name of the event.
- virtual int [GetCategoryFlags](#) () const =0
Get the category flags of the event.
- virtual std::string [ToString](#) () const
Convert the event to a string representation.
- bool [IsInCategory](#) ([EventCategory](#) category)
Check if the event is in a specific category.

Protected Member Functions

- [KeyEvent](#) (int keycode)
Construct a [KeyEvent](#) with the specified key code.

Protected Attributes

- int [m_KeyCode](#)

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled](#) = false
Indicates whether the event has been handled.

10.26.1 Detailed Description

Base class for keyboard events.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 [KeyEvent\(\)](#)

```
Vesper::KeyEvent::KeyEvent (
    int keycode) [inline], [protected]
```

Construct a [KeyEvent](#) with the specified key code.

Parameters

| | |
|----------------------|--|
| <code>keycode</code> | The key code associated with the event. Restricted specific construction to derived classes. |
|----------------------|--|

00030 : `m_KeyCode (keycode) { }`

References [m_KeyCode](#).

Referenced by [Vesper::KeyPressedEvent::KeyPressedEvent\(\)](#), [Vesper::KeyReleasedEvent::KeyReleasedEvent\(\)](#), and [Vesper::KeyTypedEvent::KeyTypedEvent\(\)](#).

10.26.3 Member Function Documentation

10.26.3.1 GetKeyCode()

`int Vesper::KeyEvent::GetKeyCode () const [inline]`

Get the key code associated with the event.

00020 { `return m_KeyCode;` }

References [m_KeyCode](#).

Referenced by [Vesper::EditorLayer::OnKeyPressed\(\)](#).

10.26.4 Member Data Documentation

10.26.4.1 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:

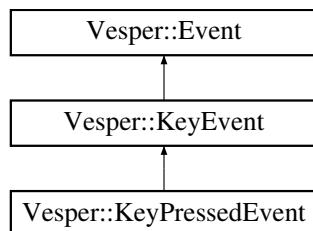
- [Vesper/src/Vesper/Events/KeyEvent.h](#)

10.27 Vesper::KeyPressedEvent Class Reference

[Event](#) for registering key press.

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

Inheritance diagram for Vesper::KeyPressedEvent:



Public Member Functions

- `KeyPressedEvent` (int keycode, int repeatCount)
Construct a `KeyPressedEvent` with the specified key code and repeat count.
- int `GetRepeatCount` () const
Get the repeat count of the key press event.
- std::string `ToString` () const override
Convert the event to a string representation.

Public Member Functions inherited from `Vesper::KeyEvent`

- int `GetKeyCode` () const
Get the key code associated with the event.

Public Member Functions inherited from `Vesper::Event`

- virtual ~`Event` ()=default
- virtual `EventType GetEventType` () const =0
Get the type of the event.
- virtual const char * `GetName` () const =0
Get the name of the event.
- virtual int `GetCategoryFlags` () const =0
Get the category flags of the event.
- bool `IsInCategory` (`EventCategory` category)
Check if the event is in a specific category.

Private Attributes

- int `m_RepeatCount`

Additional Inherited Members

Public Attributes inherited from `Vesper::Event`

- bool `Handled` = false
Indicates whether the event has been handled.

Protected Member Functions inherited from `Vesper::KeyEvent`

- `KeyEvent` (int keycode)
Construct a `KeyEvent` with the specified key code.

Protected Attributes inherited from `Vesper::KeyEvent`

- int `m_KeyCode`

10.27.1 Detailed Description

[Event](#) for registering key press.

10.27.2 Constructor & Destructor Documentation

10.27.2.1 KeyPressedEvent()

```
Vesper::KeyPressedEvent::KeyPressedEvent (
    int keycode,
    int repeatCount) [inline]
```

Construct a [KeyPressedEvent](#) with the specified key code and repeat count.

Parameters

| | |
|--------------------|--|
| <i>keycode</i> | The key code associated with the event. |
| <i>repeatCount</i> | The number of times the key press is repeated. |

```
00045 : KeyEvent(keycode), m_RepeatCount(repeatCount) {}
```

References [Vesper::KeyEvent::KeyEvent\(\)](#), and [m_RepeatCount](#).

10.27.3 Member Function Documentation

10.27.3.1 GetRepeatCount()

```
int Vesper::KeyPressedEvent::GetRepeatCount () const [inline]
```

Get the repeat count of the key press event.

```
00048 { return m_RepeatCount; }
```

References [m_RepeatCount](#).

Referenced by [Vesper::EditorLayer::OnKeyPressed\(\)](#).

10.27.3.2 ToString()

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

Convert the event to a string representation.

Reimplemented from [Vesper::Event](#).

```
00052 {
00053     std::stringstream ss;
00054     ss << "KeyPressedEvent: " << m_KeyCode << " (" << m_RepeatCount << " repeats)";
00055     return ss.str();
00056 }
```

References [Vesper::KeyEvent::m_KeyCode](#), and [m_RepeatCount](#).

10.27.4 Member Data Documentation

10.27.4.1 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:

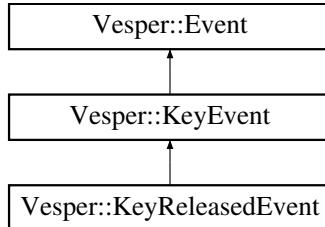
- Vesper/src/Vesper/Events/[KeyEvent.h](#)

10.28 Vesper::KeyReleasedEvent Class Reference

[Event](#) for registering key release.

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

Inheritance diagram for Vesper::KeyReleasedEvent:



Public Member Functions

- [KeyReleasedEvent](#) (int keycode)
Construct a [KeyReleasedEvent](#) with the specified key code.
- std::string [ToString](#) () const override
Convert the event to a string representation.

Public Member Functions inherited from [Vesper::KeyEvent](#)

- int [GetKeyCode](#) () const
Get the key code associated with the event.

Public Member Functions inherited from [Vesper::Event](#)

- virtual ~[Event](#) ()=default
- virtual [EventType](#) [GetEventType](#) () const =0
Get the type of the event.
- virtual const char * [GetName](#) () const =0
Get the name of the event.
- virtual int [GetCategoryFlags](#) () const =0
Get the category flags of the event.
- bool [IsInCategory](#) ([EventCategory](#) category)
Check if the event is in a specific category.

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled](#) = false
Indicates whether the event has been handled.

Protected Member Functions inherited from [Vesper::KeyEvent](#)

- [KeyEvent](#) (int keycode)
Construct a [KeyEvent](#) with the specified key code.

Protected Attributes inherited from [Vesper::KeyEvent](#)

- int [m_KeyCode](#)

10.28.1 Detailed Description

[Event](#) for registering key release.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 KeyReleasedEvent()

```
Vesper::KeyReleasedEvent::KeyReleasedEvent (
    int keycode) [inline]
```

Construct a [KeyReleasedEvent](#) with the specified key code.

Parameters

| | |
|----------------------|---|
| <code>keycode</code> | The key code associated with the event. |
|----------------------|---|

```
00072     : KeyEvent(keycode) {}
```

References [Vesper::KeyEvent::KeyEvent\(\)](#).

10.28.3 Member Function Documentation

10.28.3.1 ToString()

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

Convert the event to a string representation.

Reimplemented from [Vesper::Event](#).

```
00076     {
00077         std::stringstream ss;
00078         ss << "KeyReleasedEvent: " << m\_KeyCode;
00079         return ss.str();
00080     }
```

References [Vesper::KeyEvent::m_KeyCode](#).

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

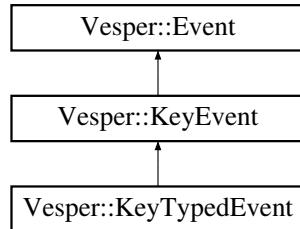
- [Vesper/src/Vesper/Events/KeyEvent.h](#)

10.29 Vesper::KeyTypedEvent Class Reference

[Event](#) for registering key typing.

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

Inheritance diagram for Vesper::KeyTypedEvent:



Public Member Functions

- [`KeyTypedEvent`](#) (int keycode)
Construct a `KeyTypedEvent` with the specified key code.
- std::string [`ToString`](#) () const override
Convert the event to a string representation.

Public Member Functions inherited from [Vesper::KeyEvent](#)

- int [`GetKeyCode`](#) () const
Get the key code associated with the event.

Public Member Functions inherited from [Vesper::Event](#)

- virtual [`~Event`](#) ()=default
- virtual [`EventType GetEventType`](#) () const =0
Get the type of the event.
- virtual const char * [`GetName`](#) () const =0
Get the name of the event.
- virtual int [`GetCategoryFlags`](#) () const =0
Get the category flags of the event.
- bool [`IsInCategory`](#) ([EventCategory](#) category)
Check if the event is in a specific category.

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [`Handled`](#) = false
Indicates whether the event has been handled.

Protected Member Functions inherited from [Vesper::KeyEvent](#)

- [KeyEvent](#) (int keycode)
Construct a [KeyEvent](#) with the specified key code.

Protected Attributes inherited from [Vesper::KeyEvent](#)

- int [m_KeyCode](#)

10.29.1 Detailed Description

[Event](#) for registering key typing.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 KeyTypedEvent()

```
Vesper::KeyTypedEvent::KeyTypedEvent (
    int keycode) [inline]
```

Construct a [KeyTypedEvent](#) with the specified key code.

Parameters

| | |
|----------------------|---|
| <code>keycode</code> | The key code associated with the event. |
|----------------------|---|

```
00094     : KeyEvent(keycode) {
00095 }
```

References [Vesper::KeyEvent::KeyEvent\(\)](#).

10.29.3 Member Function Documentation

10.29.3.1 ToString()

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

Convert the event to a string representation.

Reimplemented from [Vesper::Event](#).

```
00099     {
00100         std::stringstream ss;
00101         ss << "KeyTypedEvent: " << m\_KeyCode;
00102         return ss.str();
00103     }
```

References [Vesper::KeyEvent::m_KeyCode](#).

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

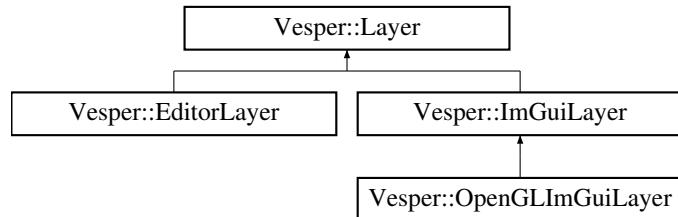
- Vesper/src/Vesper/Events/[KeyEvent.h](#)

10.30 Vesper::Layer Class Reference

Represents a reusable application layer that receives lifecycle callbacks (attach, detach, update, events, render, and ImGui render). Intended as a base class for concrete layers.

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

Inheritance diagram for Vesper::Layer:



Public Member Functions

- [Layer](#) (const std::string &name="Layer")
Constructs a [Layer](#) with an optional name for debugging purposes.
- virtual [~Layer](#) ()
- virtual void [OnAttach](#) ()
Called when the layer is attached to the application.
- virtual void [OnDetach](#) ()
Called when the layer is detached from the application.
- virtual void [OnUpdate](#) (Timestep ts)
Called every frame to update the layer with the given timestep.
- virtual void [OnEvent](#) (Event &event)
Called when an event is dispatched to the layer.
- virtual void [OnRender](#) ()
Called when the layer should render its contents.
- virtual void [OnImGuiRender](#) ()
Called when the layer should render its ImGui components.
- const std::string & [GetName](#) () const
Retrieves the name of the layer for debugging purposes.

Protected Attributes

- std::string [m_DebugName](#)
The name of the layer assigned at creation, used for debugging.

10.30.1 Detailed Description

Represents a reusable application layer that receives lifecycle callbacks (attach, detach, update, events, render, and ImGui render). Intended as a base class for concrete layers.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 Layer()

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

Constructs a [Layer](#) with an optional name for debugging purposes.

Parameters

| | |
|-------------|--|
| <i>name</i> | The name of the layer, used for debugging, defaulted to "Layer". |
|-------------|--|

```
00007      : m_DebugName(name)
00008  {
00009 }
```

References [Layer\(\)](#).

Referenced by [Layer\(\)](#).

10.30.2.2 ~Layer()

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

10.30.3 Member Function Documentation

10.30.3.1 GetName()

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

Retrieves the name of the layer for debugging purposes.

```
00053 { return m_DebugName; }
```

10.30.3.2 OnAttach()

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

Called when the layer is attached to the application.

Reimplemented in [Vesper::EditorLayer](#), [Vesper::ImGuiLayer](#), and [Vesper::OpenGLImGuiLayer](#).

```
00028 {};
```

Referenced by [Vesper::Application::PushLayer\(\)](#), and [Vesper::Application::PushOverlay\(\)](#).

10.30.3.3 OnDetach()

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

Called when the layer is detached from the application.

Reimplemented in [Vesper::EditorLayer](#), [Vesper::ImGuiLayer](#), and [Vesper::OpenGLImGuiLayer](#).

```
00031 {};
```

10.30.3.4 OnEvent()

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

Called when an event is dispatched to the layer.

Parameters

| | |
|--------------|-----------------------------|
| <i>event</i> | The event being dispatched. |
|--------------|-----------------------------|

Reimplemented in [Vesper::EditorLayer](#), [Vesper::ImGuiLayer](#), and [Vesper::OpenGLImGuiLayer](#).

```
00041 {};
```

10.30.3.5 OnImGuiRender()

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

Called when the layer should render its ImGui components.

Reimplemented in [Vesper::EditorLayer](#), [Vesper::ImGuiLayer](#), and [Vesper::OpenGLImGuiLayer](#).

```
00050 {};
```

10.30.3.6 OnRender()

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

Called when the layer should render its contents.

Note

Layers are responsible for their own rendering needs.

Todo add layer rendering into the application's render loop

```
00047 {};
```

10.30.3.7 OnUpdate()

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

Called every frame to update the layer with the given timestep.

Parameters

| | |
|-----------------|---|
| <code>ts</code> | The timestep representing the time elapsed since the last update. |
|-----------------|---|

Reimplemented in [Vesper::EditorLayer](#).

00036 {};

10.30.4 Member Data Documentation

10.30.4.1 m_DebugName

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

The name of the layer assigned at creation, used for debugging.

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

- [Vesper/src/Vesper/App/Layer.h](#)
- [Vesper/src/Vesper/App/Layer.cpp](#)

10.31 Vesper::LayerStack Class Reference

Manages an ordered stack of [Layer](#) pointers. Layers can be pushed or popped and the stack can be iterated in forward or reverse order.

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

Public Member Functions

- [LayerStack \(\)](#)
Adds the layer to the stack at the position before the first overlay.
- [~LayerStack \(\)](#)
- [void PushLayer \(Layer *layer\)](#)
Pushes the overlay layer on top of all other layers.
- [void PopLayer \(Layer *layer\)](#)
Removes the specified layer from the stack.
- [void PopOverlay \(Layer *overlay\)](#)
Removes the specified overlay layer from the stack.
- [std::vector< Layer * >::iterator begin \(\)](#)
Returns an iterator to the beginning of the layer stack.
- [std::vector< Layer * >::iterator end \(\)](#)
Returns an iterator to the end of the layer stack.
- [std::vector< Layer * >::reverse_iterator rbegin \(\)](#)
Returns a reverse iterator to the beginning of the layer stack.
- [std::vector< Layer * >::reverse_iterator rend \(\)](#)
Returns a reverse iterator to the end of the layer stack.

Private Attributes

- std::vector< Layer * > m_Layers
Vector holding pointers to the layers in the stack.
- unsigned int m_LayerInsertIndex = 0
Index indicating where to insert new layers (before overlays).

10.31.1 Detailed Description

Manages an ordered stack of [Layer](#) pointers. Layers can be pushed or popped and the stack can be iterated in forward or reverse order.

Todo Add layer priority system (maybe?)

10.31.2 Constructor & Destructor Documentation

10.31.2.1 LayerStack()

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

10.31.2.2 ~LayerStack()

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

10.31.3 Member Function Documentation

10.31.3.1 begin()

```
std::vector< Layer * >::iterator Vesper::LayerStack::begin () [inline]
```

Returns an iterator to the beginning of the layer stack.

```
00042 { return m_Layers.begin(); }
```

10.31.3.2 end()

```
std::vector< Layer * >::iterator Vesper::LayerStack::end () [inline]
```

Returns an iterator to the end of the layer stack.

```
00044 { return m_Layers.end(); }
```

10.31.3.3 PopLayer()

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

Removes the specified layer from the stack.

Parameters

| | |
|--------------|-------------------------------------|
| <i>layer</i> | Pointer to the layer to be removed. |
|--------------|-------------------------------------|

```
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](#).

10.31.3.4 PopOverlay()

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

Removes the specified overlay layer from the stack.

Parameters

| | |
|----------------|---|
| <i>overlay</i> | Pointer to the overlay layer to be removed. |
|----------------|---|

```
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 }
```

10.31.3.5 PushLayer()

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

Adds the layer to the stack at the position before the first overlay.

Parameters

| | |
|--------------|-----------------------------------|
| <i>layer</i> | Pointer to the layer to be added. |
|--------------|-----------------------------------|

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

References [m_LayerInsertIndex](#).

10.31.3.6 PushOverlay()

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

Pushes the overlay layer on top of all other layers.

Parameters

| | |
|----------------|---|
| <i>overlay</i> | Pointer to the overlay layer to be added. |
|----------------|---|

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

10.31.3.7 rbegin()

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

Returns a reverse iterator to the beginning of the layer stack.

```
00046 { return m_Layers.rbegin(); }
```

10.31.3.8 rend()

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

Returns a reverse iterator to the end of the layer stack.

```
00048 { return m_Layers.rend(); }
```

10.31.4 Member Data Documentation

10.31.4.1 m_LayerInsertIndex

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

Index indicating where to insert new layers (before overlays).

Referenced by [PopLayer\(\)](#), and [PushLayer\(\)](#).

10.31.4.2 m_Layers

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

Vector holding pointers to the layers in the stack.

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

- Vesper/src/Vesper/App/[LayerStack.h](#)
- Vesper/src/Vesper/App/[LayerStack.cpp](#)

10.32 Vesper::Log Class Reference

A logging utility class for the [Vesper](#) engine.

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

Static Public Member Functions

- static void [Init \(\)](#)
Initializes the logging system.
- static std::shared_ptr< spdlog::logger > & [GetCoreLogger \(\)](#)
Returns the core logger instance.
- static std::shared_ptr< spdlog::logger > & [GetClientLogger \(\)](#)
Returns the client logger instance.

Static Private Attributes

- static std::shared_ptr< spdlog::logger > [s_CoreLogger](#)
- static std::shared_ptr< spdlog::logger > [s_ClientLogger](#)

10.32.1 Detailed Description

A logging utility class for the [Vesper](#) engine.

Todo Rethink logging flow with Macros and possibly implement different loggers for different modules.

10.32.2 Member Function Documentation

10.32.2.1 GetClientLogger()

```
std::shared_ptr< spdlog::logger > & Vesper::Log::GetClientLogger () [inline], [static]
```

Returns the client logger instance.

```
00025 { return s_ClientLogger; }
```

10.32.2.2 GetCoreLogger()

```
std::shared_ptr< spdlog::logger > & Vesper::Log::GetCoreLogger () [inline], [static]
```

Returns the core logger instance.

```
00023 { return s_CoreLogger; }
```

10.32.2.3 `Init()`

```
void Vesper::Log::Init () [static]
```

Initializes the logging system.

```
00017  {
00018      spdlog::set_pattern("%^[%T] %n: %v%$");
00019      s_CoreLogger = spdlog::stdout_color_mt("VESPER");
00020      s_CoreLogger->set_level(spdlog::level::trace);
00021
00022      s_ClientLogger = spdlog::stdout_color_mt("APP");
00023      s_ClientLogger->set_level(spdlog::level::trace);
00024 }
```

10.32.3 Member Data Documentation

10.32.3.1 `s_ClientLogger`

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

10.32.3.2 `s_CoreLogger`

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

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

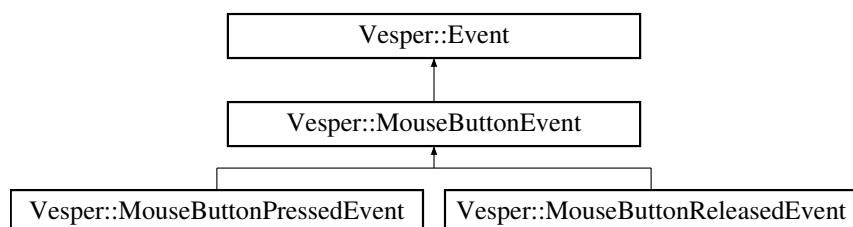
- Vesper/src/Vesper/Core/[Log.h](#)
- Vesper/src/Vesper/Core/[Log.cpp](#)

10.33 Vesper::MouseButtonEvent Class Reference

Base class for mouse button events.

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

Inheritance diagram for Vesper::MouseButtonEvent:



Public Member Functions

- int `GetMouseButton () const`

Get the mouse button associated with the event.

Public Member Functions inherited from [Vesper::Event](#)

- virtual [~Event](#) ()=default
- virtual [EventType GetEventType](#) () const =0

Get the type of the event.
- virtual const char * [GetName](#) () const =0

Get the name of the event.
- virtual int [GetCategoryFlags](#) () const =0

Get the category flags of the event.
- virtual std::string [ToString](#) () const

Convert the event to a string representation.
- bool [IsInCategory](#) ([EventCategory](#) category)

Check if the event is in a specific category.

Protected Member Functions

- [MouseEvent](#) (int button)

Construct a [MouseEvent](#) with the specified button.

Protected Attributes

- int [m_Button](#)

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled](#) = false

Indicates whether the event has been handled.

10.33.1 Detailed Description

Base class for mouse button events.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 [MouseEvent\(\)](#)

```
Vesper::MouseEvent::MouseEvent (
    int button) [inline], [protected]
```

Construct a [MouseEvent](#) with the specified button.

Parameters

| | |
|---------------------|--|
| <code>button</code> | The mouse button associated with the event. Restricted specific construction to derived classes. |
|---------------------|--|

```
00093     : m\_Button(button) {  
00094 }
```

References [m_Button](#).

Referenced by [Vesper::MouseButtonPressedEvent::MouseButtonPressedEvent\(\)](#), and [Vesper::MouseButtonReleasedEvent::MouseButtonReleasedEvent\(\)](#).

10.33.3 Member Function Documentation

10.33.3.1 GetMouseButton()

```
int Vesper::MouseButtonEvent::GetMouseButton () const [inline]
```

Get the mouse button associated with the event.

```
00084 { return m\_Button; }
```

References [m_Button](#).

Referenced by [Vesper::MouseButtonPressedEvent::ToString\(\)](#), and [Vesper::MouseButtonReleasedEvent::ToString\(\)](#).

10.33.4 Member Data Documentation

10.33.4.1 m_Button

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

Referenced by [GetMouseButton\(\)](#), and [MouseButtonEvent\(\)](#).

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

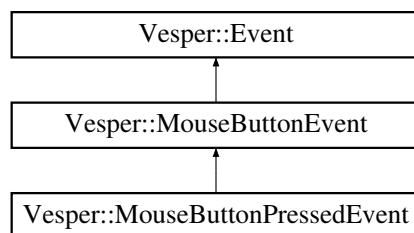
- [Vesper/src/Vesper/Events/MouseEvent.h](#)

10.34 Vesper::MouseButtonPressedEvent Class Reference

[Event](#) for registering mouse button presses.

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

Inheritance diagram for Vesper::MouseButtonPressedEvent:



Public Member Functions

- `MouseButtonPressedEvent (int button)`
Construct a `MouseButtonPressedEvent` with the specified button.
- `std::string ToString () const override`
Convert the event to a string representation.

Public Member Functions inherited from `Vesper::MouseButtonEvent`

- `int GetMouseButton () const`
Get the mouse button associated with the event.

Public Member Functions inherited from `Vesper::Event`

- `virtual ~Event ()=default`
- `virtual EventType GetEventType () const =0`
Get the type of the event.
- `virtual const char * GetName () const =0`
Get the name of the event.
- `virtual int GetCategoryFlags () const =0`
Get the category flags of the event.
- `bool IsInCategory (EventCategory category)`
Check if the event is in a specific category.

Additional Inherited Members

Public Attributes inherited from `Vesper::Event`

- `bool Handled = false`
Indicates whether the event has been handled.

Protected Member Functions inherited from `Vesper::MouseButtonEvent`

- `MouseButtonEvent (int button)`
Construct a `MouseButtonEvent` with the specified button.

Protected Attributes inherited from `Vesper::MouseButtonEvent`

- `int m_Button`

10.34.1 Detailed Description

`Event` for registering mouse button presses.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 MouseButtonPressedEvent()

```
Vesper::MouseButtonPressedEvent::MouseButtonPressedEvent (
    int button) [inline]
```

Construct a [MouseButtonPressedEvent](#) with the specified button.

Parameters

| | |
|---------------|---|
| <i>button</i> | The mouse button associated with the event. |
|---------------|---|

```
00107     : MouseButtonEvent(button) {
00108 }
```

References [Vesper::MouseButtonEvent::MouseButtonEvent\(\)](#).

10.34.3 Member Function Documentation

10.34.3.1 ToString()

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

Convert the event to a string representation.

Reimplemented from [Vesper::Event](#).

```
00112     {
00113         std::stringstream ss;
00114         ss << "MouseButtonPressedEvent: " << GetMouseButton();
00115         return ss.str();
00116     }
```

References [Vesper::MouseButtonEvent::GetMouseButton\(\)](#).

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

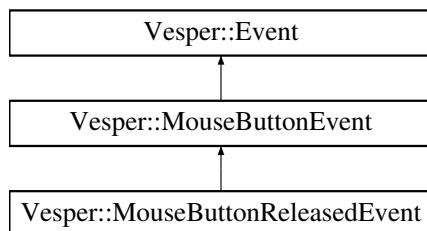
- Vesper/src/Vesper/Events/[MouseEvent.h](#)

10.35 Vesper::MouseButtonReleasedEvent Class Reference

[Event](#) for registering mouse button releases.

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

Inheritance diagram for Vesper::MouseButtonReleasedEvent:



Public Member Functions

- `MouseButtonReleasedEvent (int button)`
Construct a `MouseButtonReleasedEvent` with the specified button.
- `std::string ToString () const override`
Convert the event to a string representation.

Public Member Functions inherited from `Vesper::MouseButtonEvent`

- `int GetMouseButton () const`
Get the mouse button associated with the event.

Public Member Functions inherited from `Vesper::Event`

- `virtual ~Event ()=default`
- `virtual EventType GetEventType () const =0`
Get the type of the event.
- `virtual const char * GetName () const =0`
Get the name of the event.
- `virtual int GetCategoryFlags () const =0`
Get the category flags of the event.
- `bool IsInCategory (EventCategory category)`
Check if the event is in a specific category.

Additional Inherited Members

Public Attributes inherited from `Vesper::Event`

- `bool Handled = false`
Indicates whether the event has been handled.

Protected Member Functions inherited from `Vesper::MouseButtonEvent`

- `MouseButtonEvent (int button)`
Construct a `MouseButtonEvent` with the specified button.

Protected Attributes inherited from `Vesper::MouseButtonEvent`

- `int m_Button`

10.35.1 Detailed Description

`Event` for registering mouse button releases.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 MouseButtonReleasedEvent()

```
Vesper::MouseButtonReleasedEvent::MouseButtonReleasedEvent (
    int button) [inline]
```

Construct a [MouseButtonReleasedEvent](#) with the specified button.

Parameters

| | |
|---------------|---|
| <i>button</i> | The mouse button associated with the event. |
|---------------|---|

```
00131     : MouseButtonEvent(button) {
00132 }
```

References [Vesper::MouseEvent::MouseButtonEvent\(\)](#).

10.35.3 Member Function Documentation

10.35.3.1 ToString()

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

Convert the event to a string representation.

Reimplemented from [Vesper::Event](#).

```
00136     {
00137         std::stringstream ss;
00138         ss << "MouseButtonReleasedEvent: " << GetMouseButton();
00139         return ss.str();
00140     }
```

References [Vesper::MouseEvent::GetMouseButton\(\)](#).

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

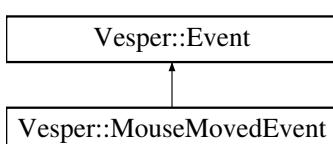
- Vesper/src/Vesper/Events/[MouseEvent.h](#)

10.36 Vesper::MouseMovedEvent Class Reference

[Event](#) for registering mouse movement.

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

Inheritance diagram for Vesper::MouseMovedEvent:



Public Member Functions

- `MouseMovedEvent (float x, float y)`
Construct a `MouseMovedEvent` with the specified x and y coordinates.
- `float GetX () const`
Get the x coordinate of the mouse.
- `float GetY () const`
Get the y coordinate of the mouse.
- `std::string ToString () const override`
Convert the event to a string representation.

Public Member Functions inherited from `Vesper::Event`

- `virtual ~Event ()=default`
- `virtual EventType GetEventType () const =0`
Get the type of the event.
- `virtual const char * GetName () const =0`
Get the name of the event.
- `virtual int GetCategoryFlags () const =0`
Get the category flags of the event.
- `bool IsInCategory (EventCategory category)`
Check if the event is in a specific category.

Private Attributes

- `float m_MouseX`
- `float m_MouseY`

Additional Inherited Members

Public Attributes inherited from `Vesper::Event`

- `bool Handled = false`
Indicates whether the event has been handled.

10.36.1 Detailed Description

`Event` for registering mouse movement.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 MouseMovedEvent()

```
Vesper::MouseMovedEvent::MouseMovedEvent (
    float x,
    float y) [inline]
```

Construct a [MouseMovedEvent](#) with the specified x and y coordinates.

Parameters

| | |
|---|--------------------------------|
| x | The x coordinate of the mouse. |
| y | The y coordinate of the mouse. |

```
00024     : m_MouseX(x), m_MouseY(y) {
00025 }
```

References [m_MouseX](#), and [m_MouseY](#).

10.36.3 Member Function Documentation

10.36.3.1 GetX()

```
float Vesper::MouseMovedEvent::GetX () const [inline]
```

Get the x coordinate of the mouse.

```
00028 { return m_MouseX; }
```

References [m_MouseX](#).

10.36.3.2 GetY()

```
float Vesper::MouseMovedEvent::GetY () const [inline]
```

Get the y coordinate of the mouse.

```
00030 { return m_MouseY; }
```

References [m_MouseY](#).

10.36.3.3 ToString()

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

Convert the event to a string representation.

Reimplemented from [Vesper::Event](#).

```
00034 {
00035     std::stringstream ss;
00036     ss << "MouseMovedEvent: " << m_MouseX << ", " << m_MouseY;
00037     return ss.str();
00038 }
```

References [m_MouseX](#), and [m_MouseY](#).

10.36.4 Member Data Documentation

10.36.4.1 m_MouseX

```
float Vesper::MouseMovedEvent::m_MouseX [private]
```

Referenced by [GetX\(\)](#), [MouseMovedEvent\(\)](#), and [ToString\(\)](#).

10.36.4.2 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:

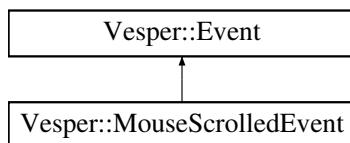
- Vesper/src/Vesper/Events/[MouseEvent.h](#)

10.37 Vesper::MouseScrolledEvent Class Reference

[Event](#) for registering mouse scroll wheel movement.

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

Inheritance diagram for Vesper::MouseScrolledEvent:



Public Member Functions

- [MouseScrolledEvent](#) (float xOffset, float yOffset)
Construct a [MouseScrolledEvent](#) with the specified x and y offsets.
- float [GetXOffset](#) () const
Get the x offset of the mouse scroll.
- float [GetYOffset](#) () const
Get the y offset of the mouse scroll.
- std::string [ToString](#) () const override
Convert the event to a string representation.

Public Member Functions inherited from [Vesper::Event](#)

- virtual [~Event](#) ()=default
- virtual [EventType GetEventType](#) () const =0
Get the type of the event.
- virtual const char * [GetName](#) () const =0
Get the name of the event.
- virtual int [GetCategoryFlags](#) () const =0
Get the category flags of the event.
- bool [IsInCategory](#) ([EventCategory](#) category)
Check if the event is in a specific category.

Private Attributes

- float [m_XOffset](#)
- float [m_YOffset](#)

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled](#) = false
Indicates whether the event has been handled.

10.37.1 Detailed Description

[Event](#) for registering mouse scroll wheel movement.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 [MouseScrolledEvent\(\)](#)

```
Vesper::MouseScrolledEvent::MouseScrolledEvent (
    float xOffset,
    float yOffset) [inline]
```

Construct a [MouseScrolledEvent](#) with the specified x and y offsets.

Parameters

| | |
|-------------------------|--|
| xOffset | The offset of the mouse scroll in the x direction. |
| yOffset | The offset of the mouse scroll in the y direction. |

```
00056           : m\_XOffset(xOffset), m\_YOffset(yOffset) {
00057 }
```

References [m_XOffset](#), and [m_YOffset](#).

10.37.3 Member Function Documentation

10.37.3.1 GetXOffset()

```
float Vesper::MouseScrolledEvent::GetXOffset () const [inline]
```

Get the x offset of the mouse scroll.

```
00060 { return m_XOffset; }
```

References [m_XOffset](#).

Referenced by [ToString\(\)](#).

10.37.3.2 GetYOffset()

```
float Vesper::MouseScrolledEvent::GetYOffset () const [inline]
```

Get the y offset of the mouse scroll.

```
00062 { return m_YOffset; }
```

References [m_YOffset](#).

Referenced by [Vesper::EditorCamera::OnMouseScroll\(\)](#), [Vesper::OrthographicCameraController::OnMouseScrolled\(\)](#), and [ToString\(\)](#).

10.37.3.3 ToString()

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

Convert the event to a string representation.

Reimplemented from [Vesper::Event](#).

```
00066     {
00067         std::stringstream ss;
00068         ss << "MouseScrolledEvent: " << GetXOffset() << ", " << GetYOffset();
00069         return ss.str();
00070     }
```

References [GetXOffset\(\)](#), and [GetYOffset\(\)](#).

10.37.4 Member Data Documentation

10.37.4.1 m_XOffset

```
float Vesper::MouseScrolledEvent::m_XOffset [private]
```

Referenced by [GetXOffset\(\)](#), and [MouseScrolledEvent\(\)](#).

10.37.4.2 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](#)

10.38 Vesper::NameComponent Struct Reference

Component that holds the name of an entity.

```
#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](#)
The name of the owning entity.

10.38.1 Detailed Description

Component that holds the name of an entity.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 NameComponent() [1/3]

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

10.38.2.2 NameComponent() [2/3]

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

10.38.2.3 NameComponent() [3/3]

```
Vesper::NameComponent::NameComponent (
    const std::string & name) [inline]
00053         : Name(name) {
00054 }
```

10.38.3 Member Function Documentation

10.38.3.1 GetName()

```
std::string & Vesper::NameComponent::GetName () [inline]
00057 { return Name; }
```

10.38.3.2 operator const std::string &()

```
Vesper::NameComponent::operator const std::string & () const [inline]
00056 { return Name; }
```

10.38.3.3 operator std::string &()

```
Vesper::NameComponent::operator std::string & () [inline]
00055 { return Name; }
```

10.38.4 Member Data Documentation

10.38.4.1 Name

`std::string Vesper::NameComponent::Name`

The name of the owning entity.

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

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

10.39 Vesper::NativeScriptComponent Struct Reference

Component that holds scripting data for an entity.

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

Public Member Functions

- `template<typename T>
void Bind ()`
- Binds a script type to this component.*

Public Attributes

- `ScriptableEntity * Instance = nullptr`
Pointer to the instance of the scriptable entity.
- `ScriptableEntity *(* InstantiateScript)()`
Function pointer to instantiate the script.
- `void(* DestroyScript)(NativeScriptComponent *)`
Function pointer to destroy the script.

10.39.1 Detailed Description

Component that holds scripting data for an entity.

10.39.2 Member Function Documentation

10.39.2.1 Bind()

```
template<typename T>
void Vesper::NativeScriptComponent::Bind () [inline]
```

Binds a script type to this component.

Template Parameters

| | |
|----------|---------------------------------|
| <i>T</i> | The type of the script to bind. |
|----------|---------------------------------|

```
00245      {
00246          InstantiateScript = []() { return static_cast<ScriptableEntity*> (new T()); };
00247          DestroyScript = [](NativeScriptComponent* nsc) { delete nsc->Instance; nsc->Instance =
00248              nullptr; }
```

10.39.3 Member Data Documentation

10.39.3.1 DestroyScript

```
void(* Vesper::NativeScriptComponent::DestroyScript) (NativeScriptComponent *)
```

Function pointer to destroy the script.

10.39.3.2 Instance

```
ScriptableEntity* Vesper::NativeScriptComponent::Instance = nullptr
```

Pointer to the instance of the scriptable entity.

10.39.3.3 InstantiateScript

```
ScriptableEntity *(* Vesper::NativeScriptComponent::InstantiateScript) ()
```

Function pointer to instantiate the script.

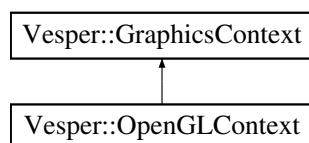
The documentation for this struct was generated from the following file:

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

10.40 Vesper::OpenGLContext Class Reference

```
#include <OpenGLContext.h>
```

Inheritance diagram for Vesper::OpenGLContext:



Public Member Functions

- [OpenGLContext](#) (GLFWwindow *windowHandle)
- virtual [~OpenGLContext](#) ()
- void [Init](#) () override
Initializes the graphics context.
- void [SwapBuffers](#) () override
Swaps the front and back buffers.

Public Member Functions inherited from [Vesper::GraphicsContext](#)

- virtual [~GraphicsContext](#) ()

Private Attributes

- GLFWwindow * [m_WindowHandle](#)

10.40.1 Constructor & Destructor Documentation

10.40.1.1 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](#).

10.40.1.2 ~OpenGLContext()

```
Vesper::OpenGLContext::~OpenGLContext () [virtual]
00019     {
00020 }
```

10.40.2 Member Function Documentation

10.40.2.1 Init()

```
void Vesper::OpenGLContext::Init () [override], [virtual]
```

Initializes the graphics context.

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](#).

10.40.2.2 SwapBuffers()

```
void Vesper::OpenGLContext::SwapBuffers () [override], [virtual]
```

Swaps the front and back buffers.

Implements [Vesper::GraphicsContext](#).

```
00045     {
00046         VZ_PROFILE_FUNCTION();
00047         glfwSwapBuffers(m_WindowHandle);
00048
00049 }
```

References [m_WindowHandle](#).

10.40.3 Member Data Documentation

10.40.3.1 m_WindowHandle

```
GLFWwindow* Vesper::OpenGLContext::m_WindowHandle [private]
```

Referenced by [Init\(\)](#), [OpenGLContext\(\)](#), and [SwapBuffers\(\)](#).

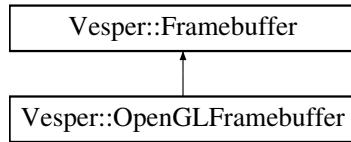
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/[OpenGLContext.h](#)
- Vesper/src/RenderAPI/OpenGL/[OpenGLContext.cpp](#)

10.41 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
Resizes the framebuffer to the given width and height.
- virtual uint32_t [GetColorAttachmentRendererID](#) () const override
Returns the renderer ID of the color attachment texture.
- virtual const FramebufferSpecification & [GetSpecification](#) () const
Returns the specification used to create the framebuffer.

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)
Creates a framebuffer with the given specification.

10.41.1 Constructor & Destructor Documentation

10.41.1.1 OpenGLFramebuffer()

```
Vesper::OpenGLFramebuffer::OpenGLFramebuffer (
    const FramebufferSpecification & spec)
00012     : m_Specification(spec)
00013     {
00014         Invalidate();
00015     }
```

References [Invalidate\(\)](#), and [m_Specification](#).

10.41.1.2 ~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](#).

10.41.2 Member Function Documentation

10.41.2.1 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](#).

10.41.2.2 GetColorAttachmentRendererID()

```
virtual uint32_t Vesper::OpenGLFramebuffer::GetColorAttachmentRendererID () const [inline],
[override], [virtual]
```

Returns the renderer ID of the color attachment texture.

Implements [Vesper::Framebuffer](#).

```
00019 { return m_ColorAttachment; }
```

References [m_ColorAttachment](#).

10.41.2.3 GetSpecification()

```
virtual const FramebufferSpecification & Vesper::OpenGLFramebuffer::GetSpecification () const
[inline], [virtual]
```

Returns the specification used to create the framebuffer.

Implements [Vesper::Framebuffer](#).

```
00020 { return m_Specification; }
```

References [m_Specification](#).

10.41.2.4 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,
00039                 GL_RGBA, GL_UNSIGNED_BYTE, nullptr);
00040     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
00041     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
00042     glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, m_ColorAttachment,
00043                           0);
00044
00045     glCreateTextures(GL_TEXTURE_2D, 1, &m_DepthAttachment);
00046     glBindTexture(GL_TEXTURE_2D, m_DepthAttachment);
00047     glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH24_STENCIL8, m_Specification.Width,
00048                 m_Specification.Height, 0, GL_DEPTH_STENCIL, GL_UNSIGNED_INT_24_8, nullptr);
00049     glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_STENCIL_ATTACHMENT, GL_TEXTURE_2D,
00050                           m_DepthAttachment, 0);
00051     VZ_CORE_ASSERT(glCheckFramebufferStatus(GL_FRAMEBUFFER) == GL_FRAMEBUFFER_COMPLETE,
00052                    "Framebuffer is complete!");
00053     glBindFramebuffer(GL_FRAMEBUFFER, 0);
00054 }
```

References [m_ColorAttachment](#), [m_DepthAttachment](#), and [m_RendererID](#).

Referenced by [OpenGLFramebuffer\(\)](#), and [Resize\(\)](#).

10.41.2.5 Resize()

```
void Vesper::OpenGLFramebuffer::Resize (
    uint32_t width,
    uint32_t height) [override], [virtual]
```

Resizes the framebuffer to the given width and height.

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](#).

10.41.2.6 Unbind()

```
void Vesper::OpenGLFramebuffer::Unbind () [override], [virtual]
```

Implements [Vesper::Framebuffer](#).

```
00063     {
00064         glBindFramebuffer(GL_FRAMEBUFFER, 0);
00065     }
```

10.41.3 Member Data Documentation

10.41.3.1 m_ColorAttachment

```
uint32_t Vesper::OpenGLFramebuffer::m_ColorAttachment [private]
```

Referenced by [GetColorAttachmentRendererID\(\)](#), [Invalidate\(\)](#), and [~OpenGLFramebuffer\(\)](#).

10.41.3.2 m_DepthAttachment

```
uint32_t Vesper::OpenGLFramebuffer::m_DepthAttachment [private]
```

Referenced by [Invalidate\(\)](#), and [~OpenGLFramebuffer\(\)](#).

10.41.3.3 m_RendererID

```
uint32_t Vesper::OpenGLFramebuffer::m_RendererID [private]
```

Referenced by [Invalidate\(\)](#), and [~OpenGLFramebuffer\(\)](#).

10.41.3.4 m_Specification

```
FramebufferSpecification Vesper::OpenGLFramebuffer::m_Specification [private]
```

Referenced by [Bind\(\)](#), [GetSpecification\(\)](#), [OpenGLFramebuffer\(\)](#), and [Resize\(\)](#).

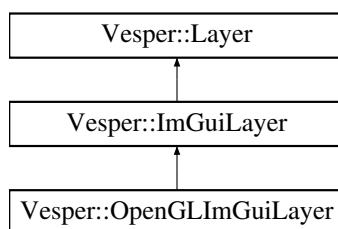
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/[OpenGLFramebuffer.h](#)
- Vesper/src/RenderAPI/OpenGL/[OpenGLFramebuffer.cpp](#)

10.42 Vesper::OpenGLImGuiLayer Class Reference

```
#include <OpenGLImGuiLayer.h>
```

Inheritance diagram for Vesper::OpenGLImGuiLayer:



Public Member Functions

- [OpenGLGuiLayer \(\)](#)
- [~OpenGLGuiLayer \(\)](#)
- virtual void [OnAttach \(\)](#) override

Called when the layer is attached to the application.
- virtual void [OnDetach \(\)](#) override

Called when the layer is detached from the application.
- virtual void [OnImGuiRender \(\)](#) override

Called when the layer should render its ImGui components.
- virtual void [OnEvent \(Event &e\)](#) override

Called when an event is dispatched to the layer.
- 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 \(const std::string &name="Layer"\)](#)

Constructs a [Layer](#) with an optional name for debugging purposes.
- virtual [~Layer \(\)](#)
- virtual void [OnUpdate \(Timestep ts\)](#)

Called every frame to update the layer with the given timestep.
- virtual void [OnRender \(\)](#)

Called when the layer should render its contents.
- const std::string & [GetName \(\)](#) const

Retrieves the name of the layer for debugging purposes.

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](#)

The name of the layer assigned at creation, used for debugging.

10.42.1 Constructor & Destructor Documentation

10.42.1.1 OpenGLImGuiLayer()

```
Vesper::OpenGLImGuiLayer::OpenGLImGuiLayer ()
00020          : ImGuiLayer\(\)
00021      {
00022 }
```

References [Vesper::ImGuiLayer::ImGuiLayer\(\)](#).

10.42.1.2 ~OpenGLImGuiLayer()

```
Vesper::OpenGLImGuiLayer::~OpenGLImGuiLayer ()
00025      {
00026 }
```

10.42.2 Member Function Documentation

10.42.2.1 Begin()

```
void Vesper::OpenGLImGuiLayer::Begin () [override], [virtual]
```

Reimplemented from [Vesper::ImGuiLayer](#).

```
00059      {
00060          ImGuiLayer::Begin\(\);
00061 }
```

References [Vesper::ImGuiLayer::Begin\(\)](#).

10.42.2.2 End()

```
void Vesper::OpenGLImGuiLayer::End () [override], [virtual]
```

Reimplemented from [Vesper::ImGuiLayer](#).

```
00064      {
00065          ImGuiLayer::End\(\);
00066 }
```

References [Vesper::ImGuiLayer::End\(\)](#).

10.42.2.3 OnAttach()

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

Called when the layer is attached to the application.

Reimplemented from [Vesper::ImGuiLayer](#).

```
00029      {
00030          ImGuiLayer::OnAttach\(\);
00031
00032
00033
00034 }
```

References [Vesper::ImGuiLayer::OnAttach\(\)](#).

10.42.2.4 OnDetach()

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

Called when the layer is detached from the application.

Reimplemented from [Vesper::ImGuiLayer](#).

```
00037  {
00038      ImGuiLayer::OnDetach ();
00039
00040
00041
00042 }
```

References [Vesper::ImGuiLayer::OnDetach\(\)](#).

10.42.2.5 OnEvent()

```
void Vesper::OpenGLImGuiLayer::OnEvent (
    Event & event) [override], [virtual]
```

Called when an event is dispatched to the layer.

Parameters

| | |
|--------------|-----------------------------|
| <i>event</i> | The event being dispatched. |
|--------------|-----------------------------|

Reimplemented from [Vesper::ImGuiLayer](#).

```
00052  {
00053      ImGuiLayer::OnEvent (e);
00054
00055
00056 }
```

References [Vesper::ImGuiLayer::OnEvent\(\)](#).

10.42.2.6 OnImGuiRender()

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

Called when the layer should render its ImGui components.

Reimplemented from [Vesper::ImGuiLayer](#).

```
00045  {
00046      ImGuiLayer::OnImGuiRender ();
00047
00048
00049 }
```

References [Vesper::ImGuiLayer::OnImGuiRender\(\)](#).

10.42.2.7 SetBlockEvents()

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

Reimplemented from [Vesper::ImGuiLayer](#).

```
00027 { m_BlockEvents = block; }
```

References [Vesper::ImGuiLayer::m_BlockEvents](#).

10.42.2.8 SetDarkThemeColors()

```
void Vesper::OpenGLImGuiLayer::SetDarkThemeColors () [override], [virtual]
```

Reimplemented from [Vesper::ImGuiLayer](#).

```
00069     {
00070         ImGuiLayer::SetDarkThemeColors ();
00071     }
```

References [Vesper::ImGuiLayer::SetDarkThemeColors\(\)](#).

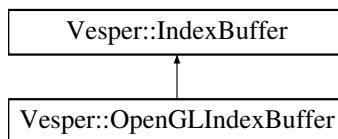
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/[OpenGLImGuiLayer.h](#)
- Vesper/src/RenderAPI/OpenGL/[OpenGLImGuiLayer.cpp](#)

10.43 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)

10.43.1 Constructor & Destructor Documentation

10.43.1.1 OpenGLIndexBuffer()

```
Vesper::OpenGLIndexBuffer::OpenGLIndexBuffer (
    uint32_t * indices,
    uint32_t count)
00063     : m_Count(count)
00064 {
00065     VZ_PROFILE_FUNCTION();
00066
00067     glGenBuffers(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](#).

10.43.1.2 ~OpenGLIndexBuffer()

```
Vesper::OpenGLIndexBuffer::~OpenGLIndexBuffer () [virtual]
00073 {
00074     VZ_PROFILE_FUNCTION();
00075
00076     glDeleteBuffers(1, &m_RendererID);
00077 }
```

References [m_RendererID](#).

10.43.2 Member Function Documentation

10.43.2.1 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 }
```

10.43.2.2 GetCount()

```
virtual uint32_t Vesper::OpenGLIndexBuffer::GetCount () const [inline], [override], [virtual]
```

Implements [Vesper::IndexBuffer](#).

```
00034 { return m_Count; }
```

References [m_Count](#).

10.43.2.3 Unbind()

```
void Vesper::OpenGLIndexBuffer::Unbind () const [override], [virtual]
```

Implements [Vesper::IndexBuffer](#).

```
00087     {
00088         VZ_PROFILE_FUNCTION();
00089         glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
00090     }
00091 }
```

10.43.3 Member Data Documentation

10.43.3.1 m_Count

```
uint32_t Vesper::OpenGLIndexBuffer::m_Count [private]
```

Referenced by [GetCount\(\)](#), and [OpenGLIndexBuffer\(\)](#).

10.43.3.2 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:

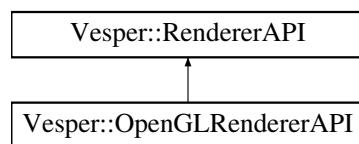
- Vesper/src/RenderAPI/OpenGL/[OpenGLBuffer.h](#)
- Vesper/src/RenderAPI/OpenGL/[OpenGLBuffer.cpp](#)

10.44 Vesper::OpenGLRendererAPI Class Reference

An implementation of the [RendererAPI](#) for OpenGL.

```
#include <OpenGLRendererAPI.h>
```

Inheritance diagram for Vesper::OpenGLRendererAPI:



Public Member Functions

- virtual void [Init \(\)](#) override
Initializes the OpenGL rendering API.
- virtual void [SetViewport \(uint32_t x, uint32_t y, uint32_t width, uint32_t height\)](#) override
Sets the viewport dimensions for OpenGL.
- virtual void [SetClearColor \(const glm::vec4 &color\)](#) override
Sets the clear color for OpenGL.
- virtual void [Clear \(\)](#) override
Clears the OpenGL rendering buffers.
- virtual void [DrawIndexed \(const Ref< VertexArray > &vertexArray, uint32_t indexCount=0\)](#) override
Draws indexed geometry using the provided vertex array in OpenGL.

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 }](#)
API.

Static Public Member Functions inherited from [Vesper::RendererAPI](#)

- static [API GetAPI \(\)](#)
Returns the current rendering API.

10.44.1 Detailed Description

An implementation of the [RendererAPI](#) for OpenGL.

Note

Should only be called by the [RenderCommand](#) class.

10.44.2 Member Function Documentation

10.44.2.1 [Clear\(\)](#)

```
void Vesper::OpenGLRendererAPI::Clear () [override], [virtual]
```

Clears the OpenGL rendering buffers.

Implements [Vesper::RendererAPI](#).

```
00030      {
00031          VZ_PROFILE_FUNCTION();
00032          glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
00034      }
```

10.44.2.2 DrawIndexed()

```
void Vesper::OpenGLRendererAPI::DrawIndexed (
    const Ref< VertexArray > & vertexArray,
    uint32_t indexCount = 0) [override], [virtual]
```

Draws indexed geometry using the provided vertex array in OpenGL.

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 }
```

10.44.2.3 Init()

```
void Vesper::OpenGLRendererAPI::Init () [override], [virtual]
```

Initializes the OpenGL rendering [API](#).

Implements [Vesper::RendererAPI](#).

```
00009  {
00010      glEnable(GL_BLEND);
00011      glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
00012      glEnable(GL_DEPTH_TEST);
00013 }
```

10.44.2.4 SetClearColor()

```
void Vesper::OpenGLRendererAPI::SetClearColor (
    const glm::vec4 & color) [override], [virtual]
```

Sets the clear color for OpenGL.

Implements [Vesper::RendererAPI](#).

```
00023  {
00024      VZ_PROFILE_FUNCTION();
00025
00026      glClearColor(color.r, color.g, color.b, color.a);
00027 }
```

10.44.2.5 SetViewport()

```
void Vesper::OpenGLRendererAPI::SetViewport (
    uint32_t x,
    uint32_t y,
    uint32_t width,
    uint32_t height) [override], [virtual]
```

Sets the viewport dimensions for OpenGL.

Implements [Vesper::RendererAPI](#).

```
00016  {
00017      VZ_PROFILE_FUNCTION();
00018
00019      glViewport(x, y, width, height);
00020 }
```

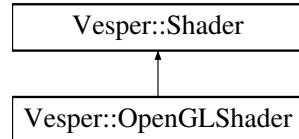
The documentation for this class was generated from the following files:

- [Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.h](#)
- [Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.cpp](#)

10.45 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
connects the shader program for use.
- void [Unbind](#) () const override
disconnects the shader program.
- virtual void [SetMat4](#) (const std::string &name, const glm::mat4 &value) override
Sets a 4x4 matrix uniform in the shader.
- virtual void [SetFloat4](#) (const std::string &name, const glm::vec4 &value) override
Sets a 4-component float vector uniform in the shader.
- virtual void [SetFloat3](#) (const std::string &name, const glm::vec3 &value) override
Sets a 3-component float vector uniform in the shader.
- virtual void [SetFloat](#) (const std::string &name, float value) override
Sets a single float uniform in the shader.
- virtual void [SetInt](#) (const std::string &name, int value) override
Sets a single integer uniform in the shader.
- virtual void [SetIntArray](#) (const std::string &name, int *values, uint32_t count) override
Sets an array of integers uniform in the shader.
- 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)

10.45.1 Constructor & Destructor Documentation**10.45.1.1 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 }
```

10.45.1.2 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\(\)](#).

10.45.1.3 ~OpenGLShader()

```
Vesper::OpenGLShader::~OpenGLShader ()
00047 {
00048     VZ_PROFILE_FUNCTION();
00049     glDeleteProgram(m_RendererID);
00050 }
```

References [m_RendererID](#).

10.45.2 Member Function Documentation

10.45.2.1 Bind()

```
void Vesper::OpenGLShader::Bind () const [override], [virtual]
```

connects the shader program for use.

Implements [Vesper::Shader](#).

```
00176     {
00177         VZ_PROFILE_FUNCTION();
00178         glUseProgram(m_RendererID);
00179     }
```

References [m_RendererID](#).

10.45.2.2 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         int glShaderIDIndex = 0;
00106
00107         for (auto& kv : shaderSources)
00108         {
00109             GLenum type = kv.first;
00110             const std::string& source = kv.second;
00111
00112             GLuint shader = glCreateShader(type);
00113
00114             const GLchar* shaderSource = source.c_str();
00115             glShaderSource(shader, 1, &shaderSource, 0);
00116
00117             glCompileShader(shader);
00118
00119             GLint isCompiled = 0;
00120             glGetShaderiv(shader, GL_COMPILE_STATUS, &isCompiled);
00121             if (isCompiled == GL_FALSE)
00122             {
00123                 GLint maxLength = 0;
00124                 glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &maxLength);
00125
00126                 std::vector<GLchar> infoLog(maxLength);
00127                 glGetShaderInfoLog(shader, maxLength, &maxLength, &infoLog[0]);
00128
00129                 glDeleteShader(shader);
00130
00131                 VZ_CORE_ERROR("{0}", infoLog.data());
00132                 VZ_CORE_ASSERT(false, "Shader compilation failure!");
00133                 break;
00134             }
00135             glAttachShader(program, shader);
00136             shaderIDs[glShaderIDIndex++] = shader;
00137         }
00138
00139         // Link our program
00140         glLinkProgram(program);
00141
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](#).

10.45.2.3 GetName()

```
virtual const std::string & Vesper::OpenGLShader::GetName () const [inline], [override],
[virtual]
```

Implements [Vesper::Shader](#).

```
00028 { return m_Name; }
```

10.45.2.4 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"
00086         keyword)
00087         std::string type = source.substr(begin, eol - begin);
00088         VZ_CORE_ASSERT(ShaderTypeFromString(type), "Invalid shader type specified");
00089
00090         size_t nextLinePos = source.find_first_not_of("\r\n", eol); // Start of shader code after
00091         shader type declaration line
00092         VZ_CORE_ASSERT(nextLinePos != std::string::npos, "Syntax error");
00093
00094         pos = source.find(typeToken, nextLinePos); // Start of next shader type declaration line
00095         shaderSources[ShaderTypeFromString(type)] = (pos == std::string::npos) ?
00096             source.substr(nextLinePos) : source.substr(nextLinePos, pos - nextLinePos);
00097     }
00098
00099     return shaderSources;
00100 }
```

10.45.2.5 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}', {1}", filepath);
00067         VZ_CORE_ASSERT(false, "Failed to open file!");
00068     }
00069     return result;
00070 }
```

10.45.2.6 SetFloat()

```
void Vesper::OpenGLShader::SetFloat (
    const std::string & name,
    float value) [override], [virtual]
```

Sets a single float uniform in the shader.

Implements [Vesper::Shader](#).

```
00201 {
00202     VZ_PROFILE_FUNCTION();
00203     UploadUniformFloat(name, value);
00204 }
```

References [UploadUniformFloat\(\)](#).

10.45.2.7 SetFloat3()

```
void Vesper::OpenGLShader::SetFloat3 (
    const std::string & name,
    const glm::vec3 & value) [override], [virtual]
```

Sets a 3-component float vector uniform in the shader.

Implements [Vesper::Shader](#).

```
00195 {
00196     VZ_PROFILE_FUNCTION();
00197     UploadUniformFloat3(name, value);
00198 }
```

10.45.2.8 SetFloat4()

```
void Vesper::OpenGLShader::SetFloat4 (
    const std::string & name,
    const glm::vec4 & value) [override], [virtual]
```

Sets a 4-component float vector uniform in the shader.

Implements [Vesper::Shader](#).

```
00189 {
00190     VZ_PROFILE_FUNCTION();
00191     UploadUniformFloat4(name, value);
00192 }
```

10.45.2.9 SetInt()

```
void Vesper::OpenGLShader::SetInt (
    const std::string & name,
    int value) [override], [virtual]
```

Sets a single integer uniform in the shader.

Implements [Vesper::Shader](#).

```
00207  {
00208      VZ_PROFILE_FUNCTION();
00209      UploadUniformInt(name, value);
00210 }
```

References [UploadUniformInt\(\)](#).

10.45.2.10 SetIntArray()

```
void Vesper::OpenGLShader::SetIntArray (
    const std::string & name,
    int * values,
    uint32_t count) [override], [virtual]
```

Sets an array of integers uniform in the shader.

Implements [Vesper::Shader](#).

```
00213  {
00214      VZ_PROFILE_FUNCTION();
00215      GLint location = glGetUniformLocation(m_RendererID, name.c_str());
00216      glUniform1iv(location, count, values);
00217 }
```

10.45.2.11 SetMat4()

```
void Vesper::OpenGLShader::SetMat4 (
    const std::string & name,
    const glm::mat4 & value) [override], [virtual]
```

Sets a 4x4 matrix uniform in the shader.

Implements [Vesper::Shader](#).

```
00220  {
00221      VZ_PROFILE_FUNCTION();
00222      UploadUniformMat4(name, value);
00223 }
```

10.45.2.12 Unbind()

```
void Vesper::OpenGLShader::Unbind () const [override], [virtual]
```

disconnects the shader program.

Implements [Vesper::Shader](#).

```
00182  {
00183      VZ_PROFILE_FUNCTION();
00184      glUseProgram(0);
00185 }
```

10.45.2.13 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\(\)](#).

10.45.2.14 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 }
```

10.45.2.15 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 }
```

10.45.2.16 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 }
```

10.45.2.17 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\(\)](#).

10.45.2.18 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 }
```

10.45.2.19 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 }
```

10.45.3 Member Data Documentation

10.45.3.1 m_Name

```
std::string Vesper::OpenGLShader::m_Name [private]
```

10.45.3.2 m_RendererID

```
unsigned int Vesper::OpenGLShader::m_RendererID [private]
```

Referenced by [Bind\(\)](#), [Compile\(\)](#), and [~OpenGLShader\(\)](#).

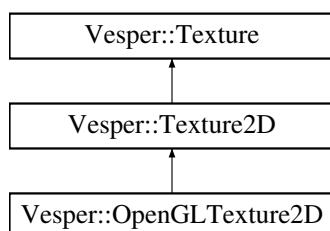
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/[OpenGLShader.h](#)
- Vesper/src/RenderAPI/OpenGL/[OpenGLShader.cpp](#)

10.46 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`
Returns the width of the texture.
- virtual `uint32_t GetHeight () const override`
Returns the height of the texture.
- virtual `uint32_t GetRendererID () const override`
Returns the renderer ID of the texture.
- virtual `void Bind (uint32_t slot) const override`
Binds the texture to the specified slot for use.
- virtual `void SetData (void *data, uint32_t size) override`
Sets the data of the texture.
- 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)`

10.46.1 Constructor & Destructor Documentation**10.46.1.1 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     glGenTextures(GL_TEXTURE_2D, 1, &m_RendererID);
00017     glBindTexture(GL_TEXTURE_2D, m_RendererID);
00018     glTexImage2D(m_RendererID, 0, m_InternalFormat, width, height, 0, m_DataFormat, GL_UNSIGNED_BYTE);
00019     glTexParameteri(m_RendererID, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
00020     glTexParameteri(m_RendererID, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
00021
00022     glTexParameteri(m_RendererID, GL_TEXTURE_WRAP_S, GL_REPEAT);
00023     glTexParameteri(m_RendererID, GL_TEXTURE_WRAP_T, GL_REPEAT);
00024 }
```

References `m_Height`, and `m_Width`.

10.46.1.2 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     m_Width = width;
00039     m_Height = height;
00040
00041     GLenum internalFormat = 0, dataFormat = 0;
00042     if (channels == 4)
00043     {
00044         internalFormat = GL_RGBA8;
00045         dataFormat = GL_RGBA;
00046     }
00047     else if (channels == 3)
00048     {
00049         internalFormat = GL_RGB8;
00050         dataFormat = GL_RGB;
00051     }
00052
00053     m_InternalFormat = internalFormat;
00054     m_DataFormat = dataFormat;
00055
00056
00057     VZ_CORE_ASSERT(internalFormat & dataFormat, "Format not supported!");
00058
00059     glGenTextures(GL_TEXTURE_2D, 1, &m_RendererID);
00060     glBindTexture(GL_TEXTURE_2D, m_RendererID);
00061     glTextureStorage2D(m_RendererID, 1, internalFormat, m_Width, m_Height);
00062
00063     glTexParameteri(m_RendererID, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
00064     glTexParameteri(m_RendererID, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
00065
00066     glTexParameteri(m_RendererID, GL_TEXTURE_WRAP_S, GL_REPEAT);
00067     glTexParameteri(m_RendererID, GL_TEXTURE_WRAP_T, GL_REPEAT);
00068
00069     glTextureSubImage2D(m_RendererID, 0, 0, 0, m_Width, m_Height, dataFormat, GL_UNSIGNED_BYTE,
00070     data);
00071     stbi_image_free(data);
00072 }
```

References [m_Height](#), [m_RendererID](#), [m_Width](#), and [OpenGLTexture2D\(\)](#).

Referenced by [OpenGLTexture2D\(\)](#).

10.46.1.3 ~OpenGLTexture2D()

```
Vesper::OpenGLTexture2D::~OpenGLTexture2D () [virtual]
00075     {
00076         VZ_PROFILE_FUNCTION();
00077         glDeleteTextures(1, &m_RendererID);
00078     }
```

References [m_RendererID](#).

10.46.2 Member Function Documentation

10.46.2.1 Bind()

```
void Vesper::OpenGLTexture2D::Bind (
    uint32_t slot) const [override], [virtual]
```

Binds the texture to the specified slot for use.

Implements [Vesper::Texture](#).

```
00081     {
00082         VZ_PROFILE_FUNCTION();
00083         glBindTextureUnit(slot, m_RendererID);
00084     }
```

References [m_RendererID](#).

10.46.2.2 GetHeight()

```
virtual uint32_t Vesper::OpenGLTexture2D::GetHeight () const [inline], [override], [virtual]
```

Returns the height of the texture.

Implements [Vesper::Texture](#).

```
00016 { return m_Height; }
```

References [m_Height](#).

10.46.2.3 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 }
```

10.46.2.4 GetRendererID()

```
virtual uint32_t Vesper::OpenGLTexture2D::GetRendererID () const [inline], [override], [virtual]
```

Returns the renderer ID of the texture.

Implements [Vesper::Texture](#).

```
00017 { return m_RendererID; }
```

References [m_RendererID](#).

10.46.2.5 GetWidth()

```
virtual uint32_t Vesper::OpenGLTexture2D::GetWidth () const [inline], [override], [virtual]
```

Returns the width of the texture.

Implements [Vesper::Texture](#).

```
00015 { return m_Width; }
```

References [m_Width](#).

10.46.2.6 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](#).

10.46.2.7 SetData()

```
void Vesper::OpenGLTexture2D::SetData (
    void * data,
    uint32_t size) [override], [virtual]
```

Sets the data of the texture.

Parameters

| | |
|-------------|--------------------------------|
| <i>data</i> | Pointer to the data to be set. |
| <i>size</i> | Size of the data in bytes. |

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,
00092     data);
00092 }
```

10.46.3 Member Data Documentation

10.46.3.1 m_DataFormat

```
Glenum Vesper::OpenGLTexture2D::m_DataFormat [private]
```

10.46.3.2 m_Height

```
uint32_t Vesper::OpenGLTexture2D::m_Height [private]
```

Referenced by [GetHeight\(\)](#), [OpenGLTexture2D\(\)](#), and [OpenGLTexture2D\(\)](#).

10.46.3.3 m_InternalFormat

```
Glenum Vesper::OpenGLTexture2D::m_InternalFormat [private]
```

10.46.3.4 m_Path

```
std::string Vesper::OpenGLTexture2D::m_Path [private]
```

10.46.3.5 m_RendererID

```
uint32_t Vesper::OpenGLTexture2D::m_RendererID [private]
```

Referenced by [Bind\(\)](#), [GetRendererID\(\)](#), [OpenGLTexture2D\(\)](#), [operator==\(\)](#), and [~OpenGLTexture2D\(\)](#).

10.46.3.6 m_Width

```
uint32_t Vesper::OpenGLTexture2D::m_Width [private]
```

Referenced by [GetWidth\(\)](#), [OpenGLTexture2D\(\)](#), and [OpenGLTexture2D\(\)](#).

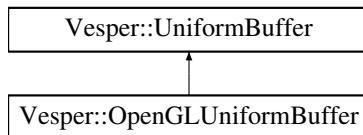
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/[OpenGLTexture.h](#)
- Vesper/src/RenderAPI/OpenGL/[OpenGLTexture.cpp](#)

10.47 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)

10.47.1 Constructor & Destructor Documentation

10.47.1.1 [OpenGLUniformBuffer\(\)](#)

```
Vesper::OpenGLUniformBuffer::OpenGLUniformBuffer (
    uint32_t size,
    uint32_t binding)

00009     {
00010         glGenBuffers(1, &m_RendererID);
00011         glBindBuffer(GL_UNIFORM_BUFFER, binding, m_RendererID); // TODO: investigate usage
        hint
00012         glBindBufferBase(GL_UNIFORM_BUFFER, binding, m_RendererID);
00013     }
```

References [m_RendererID](#).

10.47.1.2 [~OpenGLUniformBuffer\(\)](#)

```
Vesper::OpenGLUniformBuffer::~OpenGLUniformBuffer () [virtual]
00016     {
00017         glDeleteBuffers(1, &m_RendererID);
00018     }
```

References [m_RendererID](#).

10.47.2 Member Function Documentation

10.47.2.1 [SetData\(\)](#)

```
void Vesper::OpenGLUniformBuffer::SetData (
    const void * data,
    uint32_t size,
    uint32_t offset = 0) [override], [virtual]
```

Implements [Vesper::UniformBuffer](#).

```
00022     {
00023         glBindBufferSubData(m_RendererID, offset, size, data);
00024     }
```

References [m_RendererID](#).

10.47.3 Member Data Documentation

10.47.3.1 m_RendererID

```
uint32_t Vesper::OpenGLUniformBuffer::m_RendererID = 0 [private]
```

Referenced by [OpenGLUniformBuffer\(\)](#), [SetData\(\)](#), and [~OpenGLUniformBuffer\(\)](#).

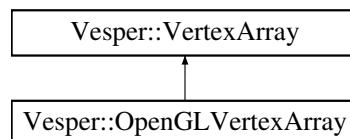
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/[OpenGLUniformBuffer.h](#)
- Vesper/src/RenderAPI/OpenGL/[OpenGLUniformBuffer.cpp](#)

10.48 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](#)
Adds a vertex buffer to the vertex array.
- void [SetIndexBuffer \(const Ref< IndexBuffer > &indexBuffer\) override](#)
Sets the index buffer for the vertex array.
- 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 \(\)](#)

10.48.1 Constructor & Destructor Documentation

10.48.1.1 [OpenGLVertexArray\(\)](#)

```
Vesper::OpenGLVertexArray::OpenGLVertexArray ()
00031     {
00032         VZ_PROFILE_FUNCTION();
00033         glGenVertexArrays(1, &m_RendererID);
00035     }
```

References [m_RendererID](#).

10.48.1.2 [~OpenGLVertexArray\(\)](#)

```
Vesper::OpenGLVertexArray::~OpenGLVertexArray ()
00038     {
00039         VZ_PROFILE_FUNCTION();
00040         glDeleteVertexArrays(1, &m_RendererID);
00042     }
```

References [m_RendererID](#).

10.48.2 Member Function Documentation

10.48.2.1 [AddVertexBuffer\(\)](#)

```
void Vesper::OpenGLVertexArray::AddVertexBuffer (
    const Ref< VertexBuffer > & vertexBuffer) [override], [virtual]
```

Adds a vertex buffer to the vertex array.

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         uint32_t index = 0;
00067         const auto& layout = vertexBuffer->GetLayout();
00068         for (const auto& element : layout)
00069         {
00070             VZ_PROFILE_SCOPE("VertexBufferElement");
00071             glEnableVertexAttribArray(index);
00072             glVertexAttribPointer(index, element.GetComponentCount(),
00073                 ShaderDataTypeToOpenGLBaseType(element.Type),
00074                 element.Normalized ? GL_TRUE : GL_FALSE,
00075                 layout.GetStride(), (const void*)element.Offset);
00076             index++;
00077         }
00078     }
00079     m_VertexBuffers.push_back(vertexBuffer);
00080 }
```

References [m_RendererID](#).

10.48.2.2 Bind()

```
void Vesper::OpenGLVertexArray::Bind () const [override], [virtual]
```

Implements [Vesper::VertexArray](#).

```
00045     {
00046         VZ_PROFILE_FUNCTION();
00047         glBindVertexArray(m_RendererID);
00049     }
```

References [m_RendererID](#).

10.48.2.3 GetIndexBuffer()

```
const Ref< IndexBuffer > & Vesper::OpenGLVertexArray::GetIndexBuffer () const [inline], [override], [virtual]
```

Implements [Vesper::VertexArray](#).

```
00018 { return m_IndexBuffer; }
```

10.48.2.4 GetVertexBuffers()

```
const std::vector< Ref< VertexBuffer > > & Vesper::OpenGLVertexArray::GetVertexBuffers () [inline], [override], [virtual]
```

Implements [Vesper::VertexArray](#).

```
00017 { return m_VertexBuffers; }
```

10.48.2.5 SetIndexBuffer()

```
void Vesper::OpenGLVertexArray::SetIndexBuffer (
    const Ref< IndexBuffer > & indexBuffer) [override], [virtual]
```

Sets the index buffer for the vertex array.

Implements [Vesper::VertexArray](#).

```
00083     {
00084         VZ_PROFILE_FUNCTION();
00085         glBindVertexArray(m_RendererID);
00086         indexBuffer->Bind();
00087         m_IndexBuffer = indexBuffer;
00089     }
```

References [m_RendererID](#).

10.48.2.6 Unbind()

```
void Vesper::OpenGLVertexArray::Unbind () const [override], [virtual]
```

Implements [Vesper::VertexArray](#).

```
00052     {
00053         VZ_PROFILE_FUNCTION();
00054         glBindVertexArray(0);
00056     }
```

10.48.3 Member Data Documentation

10.48.3.1 m_IndexBuffer

```
Ref<IndexBuffer> Vesper::OpenGLVertexArray::m_IndexBuffer [private]
```

10.48.3.2 m_RendererID

```
uint32_t Vesper::OpenGLVertexArray::m_RendererID [private]
```

Referenced by [AddVertexBuffer\(\)](#), [Bind\(\)](#), [OpenGLVertexArray\(\)](#), [SetIndexBuffer\(\)](#), and [~OpenGLVertexArray\(\)](#).

10.48.3.3 m_VertexBufferIndex

```
uint32_t Vesper::OpenGLVertexArray::m_VertexBufferIndex = 0 [private]
```

10.48.3.4 m_VertexBuffers

```
std::vector<Ref<VertexBuffer>> Vesper::OpenGLVertexArray::m_VertexBuffers [private]
```

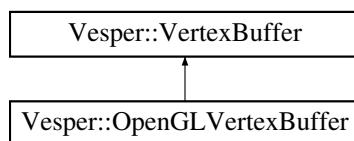
The documentation for this class was generated from the following files:

- Vesper/src/RenderAPI/OpenGL/[OpenGLVertexArray.h](#)
- Vesper/src/RenderAPI/OpenGL/[OpenGLVertexArray.cpp](#)

10.49 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 \(float *vertices, uint32_t size\)](#)

10.49.1 Constructor & Destructor Documentation

10.49.1.1 OpenGLVertexBuffer() [1/2]

```
Vesper::OpenGLVertexBuffer::OpenGLVertexBuffer (
    uint32_t size)
00013     {
00014         VZ_PROFILE_FUNCTION();
00015         glGenBuffers(1, &m_RendererID);
00016         glBindBuffer(GL_ARRAY_BUFFER, m_RendererID);
00017         glBufferData(GL_ARRAY_BUFFER, size, nullptr, GL_DYNAMIC_DRAW);
00018     }
00019 }
```

References [m_RendererID](#).

10.49.1.2 OpenGLVertexBuffer() [2/2]

```
Vesper::OpenGLVertexBuffer::OpenGLVertexBuffer (
    float * vertices,
    uint32_t size)
00022     {
00023         VZ_PROFILE_FUNCTION();
00024         glGenBuffers(1, &m_RendererID);
00025         glBindBuffer(GL_ARRAY_BUFFER, m_RendererID);
00026         glBufferData(GL_ARRAY_BUFFER, size, vertices, GL_STATIC_DRAW);
00027     }
00028 }
```

References [m_RendererID](#).

10.49.1.3 ~OpenGLVertexBuffer()

```
Vesper::OpenGLVertexBuffer::~OpenGLVertexBuffer () [virtual]
00031     {
00032         VZ_PROFILE_FUNCTION();
00033         glDeleteBuffers(1, &m_RendererID);
00034     }
00035 }
```

References [m_RendererID](#).

10.49.2 Member Function Documentation

10.49.2.1 Bind()

```
void Vesper::OpenGLVertexBuffer::Bind () const [override], [virtual]
```

Implements [Vesper::VertexBuffer](#).

```
00038     {
00039         VZ_PROFILE_FUNCTION();
00040         glBindBuffer(GL_ARRAY_BUFFER, m_RendererID);
00042     }
```

10.49.2.2 GetLayout()

```
virtual const BufferLayout & Vesper::OpenGLVertexBuffer::GetLayout () const [inline], [override], [virtual]
```

Implements [Vesper::VertexBuffer](#).

```
00018 { return m_Layout; }
```

10.49.2.3 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     }
```

10.49.2.4 SetLayout()

```
virtual void Vesper::OpenGLVertexBuffer::setLayout (
    const BufferLayout & layout) [inline], [override], [virtual]
```

Implements [Vesper::VertexBuffer](#).

```
00017 { m_Layout = layout; }
```

10.49.2.5 Unbind()

```
void Vesper::OpenGLVertexBuffer::Unbind () const [override], [virtual]
```

Implements [Vesper::VertexBuffer](#).

```
00045     {
00046         VZ_PROFILE_FUNCTION();
00047         glBindBuffer(GL_ARRAY_BUFFER, 0);
00049     }
```

10.49.3 Member Data Documentation

10.49.3.1 m_Layout

```
BufferLayout Vesper::OpenGLVertexBuffer::m_Layout [private]
```

10.49.3.2 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](#)

10.50 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

10.50.1 Constructor & Destructor Documentation

10.50.1.1 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\(\)](#).

10.50.2 Member Function Documentation

10.50.2.1 GetPosition()

```
const glm::vec3 & Vesper::OrthographicCamera::GetPosition () const [inline]
00014 { return m_Position; }
```

10.50.2.2 GetProjectionMatrix()

```
const glm::mat4 & Vesper::OrthographicCamera::GetProjectionMatrix () const [inline]
00020 { return m_ProjectionMatrix; }
```

10.50.2.3 GetRotation()

```
const float Vesper::OrthographicCamera::GetRotation () const [inline]
00017 { return m_Rotation; }
```

References [m_Rotation](#).

10.50.2.4 GetViewMatrix()

```
const glm::mat4 & Vesper::OrthographicCamera::GetViewMatrix () const [inline]
00021 { return m_ViewMatrix; }
```

10.50.2.5 GetViewProjectionMatrix()

```
const glm::mat4 & Vesper::OrthographicCamera::GetViewProjectionMatrix () const [inline]
00022 { return m_ViewProjectionMatrix; }
```

10.50.2.6 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\(\)](#).

10.50.2.7 SetPosition()

```

void Vesper::OrthographicCamera::SetPosition (
    const glm::vec3 & position) [inline]
00013 { m_Position = position; RecalculateViewMatrix(); }
```

References [RecalculateViewMatrix\(\)](#).

10.50.2.8 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 }
```

10.50.2.9 SetRotation()

```

void Vesper::OrthographicCamera::SetRotation (
    float rotation) [inline]
00016 { m_Rotation = rotation; RecalculateViewMatrix(); }
```

References [m_Rotation](#), and [RecalculateViewMatrix\(\)](#).

10.50.3 Member Data Documentation

10.50.3.1 m_Position

```

glm::vec3 Vesper::OrthographicCamera::m_Position = { 0.0f, 0.0f, 0.0f } [private]
00030 { 0.0f, 0.0f, 0.0f };
```

10.50.3.2 m_ProjectionMatrix

```

glm::mat4 Vesper::OrthographicCamera::m_ProjectionMatrix [private]
```

10.50.3.3 m_Rotation

```
float Vesper::OrthographicCamera::m_Rotation = 0.0f [private]
```

Referenced by [GetRotation\(\)](#), and [SetRotation\(\)](#).

10.50.3.4 m_ViewMatrix

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

10.50.3.5 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](#)

10.51 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](#)

10.51.1 Member Function Documentation

10.51.1.1 GetHeight()

```
float Vesper::OrthographicCameraBounds::GetHeight () const [inline]
00020 { return Top - Bottom; }
```

References [Bottom](#), and [Top](#).

10.51.1.2 GetWidth()

```
float Vesper::OrthographicCameraBounds::GetWidth () const [inline]  
00019 { return Right - Left; }
```

References [Left](#), and [Right](#).

10.51.2 Member Data Documentation

10.51.2.1 Bottom

```
float Vesper::OrthographicCameraBounds::Bottom
```

Referenced by [GetHeight\(\)](#), and [Vesper::OrthographicCameraController::UpdateCameraBounds\(\)](#).

10.51.2.2 Left

```
float Vesper::OrthographicCameraBounds::Left
```

Referenced by [GetWidth\(\)](#), and [Vesper::OrthographicCameraController::UpdateCameraBounds\(\)](#).

10.51.2.3 Right

```
float Vesper::OrthographicCameraBounds::Right
```

Referenced by [GetWidth\(\)](#), and [Vesper::OrthographicCameraController::UpdateCameraBounds\(\)](#).

10.51.2.4 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](#)

10.52 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`

10.52.1 Constructor & Destructor Documentation

10.52.1.1 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\(\)](#).

10.52.2 Member Function Documentation

10.52.2.1 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\(\)](#).

10.52.2.2 CanRotate()

```
bool Vesper::OrthographicCameraController::CanRotate () const [inline]
00053 { return m_Rotation; }
```

References [m_Rotation](#).

Referenced by [OnImGuiRender\(\)](#).

10.52.2.3 GetAspectRatio()

```
float Vesper::OrthographicCameraController::GetAspectRatio () const
00124 {
00125     return m_AspectRatio;
00126 }
```

References [m_AspectRatio](#).

10.52.2.4 GetBounds()

```
OrthographicCameraBounds Vesper::OrthographicCameraController::GetBounds () const [inline]  
00036 { return m_Bounds; }
```

References [m_Bounds](#).

10.52.2.5 GetCamera() [1/2]

```
OrthographicCamera & Vesper::OrthographicCameraController::GetCamera () [inline]  
00034 { return camera; }
```

10.52.2.6 GetCamera() [2/2]

```
const OrthographicCamera & Vesper::OrthographicCameraController::GetCamera () const [inline]  
00035 { return camera; }
```

10.52.2.7 GetMoveSpeed()

```
float Vesper::OrthographicCameraController::GetMoveSpeed () const [inline]  
00042 { return m_CameraMoveSpeed; }
```

References [m_CameraMoveSpeed](#).

Referenced by [OnImGuiRender\(\)](#).

10.52.2.8 GetPosition()

```
glm::vec3 Vesper::OrthographicCameraController::GetPosition () const [inline]  
00038 { return m_CameraPosition; }
```

10.52.2.9 GetRotation()

```
float Vesper::OrthographicCameraController::GetRotation () const [inline]  
00045 { return m_CameraRotation; }
```

References [m_CameraRotation](#).

Referenced by [OnImGuiRender\(\)](#).

10.52.2.10 GetRotationSpeed()

```
float Vesper::OrthographicCameraController::GetRotationSpeed () const [inline]  
00048 { return m_CameraRotationSpeed; }
```

References [m_CameraRotationSpeed](#).

Referenced by [OnImGuiRender\(\)](#).

10.52.2.11 GetZoomLevel()

```
float Vesper::OrthographicCameraController::GetZoomLevel () const [inline]
00057 { return m_ZoomLevel; }
```

References [m_ZoomLevel](#).

10.52.2.12 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\(\)](#).

10.52.2.13 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\(\)](#).

10.52.2.14 OnMouseScrolled()

```

00065     bool Vesper::OrthographicCameraController::OnMouseScrolled (
00066         MouseScrolledEvent & e) [private]
00067     {
00068         VZ_PROFILE_FUNCTION();
00069         m_ZoomLevel -= e.GetYOffset() * 0.25f;
00070         m_ZoomLevel = std::max(m_ZoomLevel, 0.25f);
00071         CalculateView();
00072         return false;
00073     }

```

References [CalculateView\(\)](#), [Vesper::MouseScrolledEvent::GetYOffset\(\)](#), and [m_ZoomLevel](#).

Referenced by [OnEvent\(\)](#).

10.52.2.15 OnResize()

```

00054     void Vesper::OrthographicCameraController::OnResize (
00055         float width,
00056         float height)
00057     {
00058         VZ_PROFILE_FUNCTION();
00059         m_AspectRatio = width / height;
00060         CalculateView();
00061     }
00062 }

```

References [CalculateView\(\)](#), and [m_AspectRatio](#).

Referenced by [OnWindowResized\(\)](#).

10.52.2.16 OnUpdate()

```

00021     void Vesper::OrthographicCameraController::OnUpdate (
00022         Timestep ts)
00023     {
00024         VZ_PROFILE_FUNCTION();
00025         if (Input::IsKeyPressed(Key::A))
00026             m_CameraPosition.x -= m_CameraMoveSpeed * ts;
00027         else if (Input::IsKeyPressed(Key::D))
00028             m_CameraPosition.x += m_CameraMoveSpeed * ts;
00029         if (Input::IsKeyPressed(Key::W))
00030             m_CameraPosition.y += m_CameraMoveSpeed * ts;
00031         else if (Input::IsKeyPressed(Key::S))
00032             m_CameraPosition.y -= m_CameraMoveSpeed * ts;
00033         if (m_Rotation)
00034         {
00035             if (Input::IsKeyPressed(Key::Q))
00036                 m_CameraRotation -= m_CameraRotationSpeed * ts;
00037             else if (Input::IsKeyPressed(Key::E))
00038                 m_CameraRotation += m_CameraRotationSpeed * ts;
00039             camera.SetRotation(m_CameraRotation);
00040         }
00041         camera.SetPosition(m_CameraPosition);
00042     }
00043 }
00044
00045 }

```

References [Vesper::Key::E](#), [Vesper::Input::IsKeyPressed\(\)](#), [m_CameraRotation](#), [m_CameraRotationSpeed](#), [m_Rotation](#), and [Vesper::Key::Q](#).

10.52.2.17 OnUpdateBounds()

```
void Vesper::OrthographicCameraController::OnUpdateBounds () [private]
00091 {
00092     // Nothing for now
00093 }
```

Referenced by [UpdateCameraBounds\(\)](#).

10.52.2.18 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\(\)](#).

10.52.2.19 SetAspectRatio()

```
void Vesper::OrthographicCameraController::SetAspectRatio (
    float aspectRatio)
00129 {
00130     m_AspectRatio = aspectRatio;
00131     CalculateView();
00132 }
```

References [CalculateView\(\)](#), and [m_AspectRatio](#).

10.52.2.20 SetCanRotate()

```
void Vesper::OrthographicCameraController::SetCanRotate (
    bool canRotate) [inline]
00054 { m_Rotation = canRotate; }
```

References [m_Rotation](#).

Referenced by [OnImGuiRender\(\)](#).

10.52.2.21 SetMoveSpeed()

```
void Vesper::OrthographicCameraController::SetMoveSpeed (
    float speed)
00108 {
00109     m_CameraMoveSpeed = speed;
00110 }
```

References [m_CameraMoveSpeed](#).

Referenced by [OnImGuiRender\(\)](#).

10.52.2.22 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\(\)](#).

10.52.2.23 SetRotation()

```

void Vesper::OrthographicCameraController::SetRotation (
    float rotation)
00113    {
00114        m_CameraRotation = rotation;
00115        camera.SetRotation(m_CameraRotation);
00116    }

```

References [m_CameraRotation](#).

Referenced by [OnImGuiRender\(\)](#).

10.52.2.24 SetRotationSpeed()

```

void Vesper::OrthographicCameraController::SetRotationSpeed (
    float speed)
00119    {
00120        m_CameraRotationSpeed = speed;
00121    }

```

References [m_CameraRotationSpeed](#).

Referenced by [OnImGuiRender\(\)](#).

10.52.2.25 SetZoomLevel()

```

void Vesper::OrthographicCameraController::SetZoomLevel (
    float level) [inline]
00056 { m_ZoomLevel = level; CalculateView(); }

```

References [CalculateView\(\)](#), and [m_ZoomLevel](#).

10.52.2.26 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::OrthographicCameraBounds::Top](#).

Referenced by [CalculateView\(\)](#).

10.52.3 Member Data Documentation

10.52.3.1 camera

```
OrthographicCamera Vesper::OrthographicCameraController::camera [private]
```

10.52.3.2 m_AspectRatio

```
float Vesper::OrthographicCameraController::m_AspectRatio [private]
```

Referenced by [GetAspectRatio\(\)](#), [OnResize\(\)](#), [OrthographicCameraController\(\)](#), [SetAspectRatio\(\)](#), and [UpdateCameraBounds\(\)](#).

10.52.3.3 m_Bounds

```
OrthographicCameraBounds Vesper::OrthographicCameraController::m_Bounds [private]
```

Referenced by [GetBounds\(\)](#), [OrthographicCameraController\(\)](#), and [UpdateCameraBounds\(\)](#).

10.52.3.4 m_CameraMoveSpeed

```
float Vesper::OrthographicCameraController::m_CameraMoveSpeed = 5.0f [private]
```

Referenced by [GetMoveSpeed\(\)](#), and [SetMoveSpeed\(\)](#).

10.52.3.5 m_CameraPosition

```
glm::vec3 Vesper::OrthographicCameraController::m_CameraPosition = { 0.0f, 0.0f, 0.0f } [private]  
00074 { 0.0f, 0.0f, 0.0f };
```

10.52.3.6 m_CameraRotation

```
float Vesper::OrthographicCameraController::m_CameraRotation = 0.0f [private]
```

Referenced by [GetRotation\(\)](#), [OnUpdate\(\)](#), and [SetRotation\(\)](#).

10.52.3.7 m_CameraRotationSpeed

```
float Vesper::OrthographicCameraController::m_CameraRotationSpeed = 180.0f [private]
```

Referenced by [GetRotationSpeed\(\)](#), [OnUpdate\(\)](#), and [SetRotationSpeed\(\)](#).

10.52.3.8 m_Rotation

```
bool Vesper::OrthographicCameraController::m_Rotation = true [private]
```

Referenced by [CanRotate\(\)](#), [OnUpdate\(\)](#), [OrthographicCameraController\(\)](#), and [SetCanRotate\(\)](#).

10.52.3.9 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](#)

10.53 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](#)

10.53.1 Class Documentation

10.53.1.1 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 | |

10.53.2 Constructor & Destructor Documentation

10.53.2.1 ParticleSystem() [1/2]

```
Vesper::ParticleSystem::ParticleSystem ()
00013     {
00014         m_PoolIndex = 999;
00015         m_ParticlePool.resize(1000);
00016     }
```

References [m_PoolIndex](#).

10.53.2.2 ParticleSystem() [2/2]

```
Vesper::ParticleSystem::ParticleSystem (
    uint32_t maxParticles)
00019     : m_PoolIndex(maxParticles - 1)
00020     {
00021         m_ParticlePool.resize(maxParticles);
00022     }
```

References [m_PoolIndex](#).

10.53.3 Member Function Documentation

10.53.3.1 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 *
00069     (Vesper::Random::Float1() - 0.5f);
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 *
00078     (Vesper::Random::Float1() - 0.5f);
00079     particle.SizeEnd = particleProps.SizeEnd;
00080
00081     particle.LifeTime = particleProps.LifeTime + particleProps.LifetimeVariation *
00082     (Vesper::Random::Float1() - 0.5f);
00083     particle.LifeRemaining = particle.LifeTime;
00084     m_PoolIndex = --m_PoolIndex % m_ParticlePool.size();
00085 }
```

References [Vesper::ParticleSystem::Particle::Active](#), [Vesper::Random::Float1\(\)](#), [Vesper::ParticleSystem::Particle::LifeRemaining](#), [Vesper::ParticleSystem::Particle::LifeTime](#), [Vesper::ParticleSystem::Particle::Rotation](#), [Vesper::ParticleSystem::Particle::SizeBegin](#), [Vesper::ParticleSystem::Particle::SizeEnd](#), [Vesper::ParticleSystem::Particle::SizeVariation](#), [Vesper::ParticleProps::LifetimeVariation](#), [Vesper::ParticleProps::RotationVariation](#), [Vesper::ParticleProps::SizeEnd](#), and [Vesper::ParticleProps::SizeVariation](#).

10.53.3.2 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 }
```

10.53.3.3 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 }
```

10.53.3.4 SetParticleProps()

```
void Vesper::ParticleSystem::SetParticleProps (
    const ParticleProps & particleProps) [inline]
00033 { m_Props = particleProps; }
```

10.53.4 Member Data Documentation

10.53.4.1 m_ParticlePool

```
std::vector<Particle> Vesper::ParticleSystem::m_ParticlePool [private]
```

10.53.4.2 m_PoolIndex

```
uint32_t Vesper::ParticleSystem::m_PoolIndex = 999 [private]
```

Referenced by [ParticleSystem\(\)](#), and [ParticleSystem\(\)](#).

10.53.4.3 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](#)

10.54 Vesper::RenderCommand Class Reference

A static class that provides an interface for issuing rendering commands.

```
#include <RenderCommand.h>
```

Static Public Member Functions

- static void [Init \(\)](#)
Initializes the rendering API.
- static void [SetViewport \(uint32_t x, uint32_t y, uint32_t width, uint32_t height\)](#)
Sets the viewport dimensions for the renderer.
- static void [SetClearColor \(const glm::vec4 &color\)](#)
Sets the clear color for the renderer.
- static void [Clear \(\)](#)
Clears the rendering buffers.
- static void [DrawIndexed \(const Ref<VertexArray> &vertexArray, uint32_t indexCount=0\)](#)
Draws indexed geometry using the specified vertex array and index count.

Static Private Attributes

- static RendererAPI * s_RendererAPI = new OpenGLRendererAPI()

10.54.1 Detailed Description

A static class that provides an interface for issuing rendering commands.

10.54.2 Member Function Documentation

10.54.2.1 Clear()

```
void Vesper::RenderCommand::Clear () [inline], [static]
```

Clears the rendering buffers.

```
00033     {
00034         s_RendererAPI->Clear();
00035     }
```

References [Vesper::RendererAPI::Clear\(\)](#), and [s_RendererAPI](#).

Referenced by [Vesper::EditorLayer::OnUpdate\(\)](#).

10.54.2.2 DrawIndexed()

```
void Vesper::RenderCommand::DrawIndexed (
    const Ref< VertexArray > & vertexArray,
    uint32_t indexCount = 0) [inline], [static]
```

Draws indexed geometry using the specified vertex array and index count.

```
00039     {
00040         s_RendererAPI->DrawIndexed(vertexArray, indexCount);
00041     }
```

References [s_RendererAPI](#).

10.54.2.3 Init()

```
void Vesper::RenderCommand::Init () [inline], [static]
```

Initializes the rendering API.

```
00015     {
00016         s_RendererAPI->Init();
00017     }
```

References [Vesper::RendererAPI::Init\(\)](#), and [s_RendererAPI](#).

Referenced by [Vesper::Renderer::Init\(\)](#).

10.54.2.4 SetClearColor()

```
void Vesper::RenderCommand::SetClearColor (
    const glm::vec4 & color) [inline], [static]
```

Sets the clear color for the renderer.

```
00027      {
00028          s_RendererAPI->SetClearColor(color);
00029      }
```

References [s_RendererAPI](#).

10.54.2.5 SetViewport()

```
void Vesper::RenderCommand::SetViewport (
    uint32_t x,
    uint32_t y,
    uint32_t width,
    uint32_t height) [inline], [static]
```

Sets the viewport dimensions for the renderer.

```
00021      {
00022          s_RendererAPI->SetViewport(x, y, width, height);
00023      }
```

References [s_RendererAPI](#), and [Vesper::RendererAPI::SetViewport\(\)](#).

Referenced by [Vesper::Renderer::OnWindowResize\(\)](#).

10.54.3 Member Data Documentation

10.54.3.1 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

10.55 Vesper::Renderer Class Reference

The main renderer class responsible for managing rendering operations.

```
#include <Renderer.h>
```

Classes

- struct [SceneData](#)
Scene data structure containing view-projection matrix. [More...](#)

Static Public Member Functions

- static void [Init \(\)](#)
Initializes the renderer.
- static void [OnWindowResize \(uint32_t width, uint32_t height\)](#)
Handles window resize events by resizing the viewport.
- static void [BeginScene \(OrthographicCamera &camera\)](#)
Begins a new scene with the given orthographic camera.
- static void [EndScene \(\)](#)
Ends the current scene.
- static void [Submit \(const Ref< Shader > &shader, const Ref< VertexArray > &vertexArray, const glm::mat4 &transform=glm::mat4\(1.0f\)\)](#)
Submits a draw call with the specified shader, vertex array, and transform.
- static [RendererAPI::API GetAPI \(\)](#)
Retrieves the current rendering API.

Static Private Attributes

- static [SceneData * s_SceneData = new Renderer::SceneData](#)
Pointer to the current scene data.

10.55.1 Detailed Description

The main renderer class responsible for managing rendering operations.

10.55.2 Class Documentation

10.55.2.1 struct Vesper::Renderer::SceneData

[Scene](#) data structure containing view-projection matrix.

Class Members

| | | |
|------|--------------------------------------|--|
| mat4 | ViewProjectionMatrix | The combined view-projection matrix for the scene. |
|------|--------------------------------------|--|

10.55.3 Member Function Documentation

10.55.3.1 [BeginScene\(\)](#)

```
void Vesper::Renderer::BeginScene (
    OrthographicCamera & camera) [static]
```

Begins a new scene with the given orthographic camera.

Parameters

| | |
|---------------------|---|
| <code>camera</code> | The orthographic camera defining the view and projection for the scene. |
|---------------------|---|

Todo Support for other camera types.

```
00028     {
00029         VZ_PROFILE_FUNCTION();
00030         s_SceneData->ViewProjectionMatrix = camera.GetViewProjectionMatrix();
00031     }
```

References [s_SceneData](#).

10.55.3.2 EndScene()

```
void Vesper::Renderer::EndScene () [static]
```

Ends the current scene.

Note

Currently does nothing as scene closure is automatic, but could be useful for a render interface

```
00034     {
00035         VZ_PROFILE_FUNCTION();
00036     }
00037 }
```

10.55.3.3 GetAPI()

```
RendererAPI::API Vesper::Renderer::GetAPI () [inline], [static]
```

Retrieves the current rendering API.

```
00040 { 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\(\)](#).

10.55.3.4 Init()

```
void Vesper::Renderer::Init () [static]
```

Initializes the renderer.

Note

This initializes both the [RenderCommand](#) and [Renderer2D](#) systems.

```
00014     {
00015         VZ_PROFILE_FUNCTION();
00016         RenderCommand::Init();
00017         Renderer2D::Init();
00018     }
00019 }
```

References [Vesper::RenderCommand::Init\(\)](#), and [Vesper::Renderer2D::Init\(\)](#).

Referenced by [Vesper::Application::Application\(\)](#).

10.55.3.5 OnWindowResize()

```
void Vesper::Renderer::OnWindowResize (
    uint32_t width,
    uint32_t height) [static]
```

Handles window resize events by resizing the viewport.

```
00022 {
00023     VZ_PROFILE_FUNCTION();
00024     RenderCommand::SetViewport(0, 0, width, height);
00025 }
```

References [Vesper::RenderCommand::SetViewport\(\)](#).

Referenced by [Vesper::Application::OnWindowResize\(\)](#).

10.55.3.6 Submit()

```
void Vesper::Renderer::Submit (
    const Ref< Shader > & shader,
    const Ref< VertexArray > & vertexArray,
    const glm::mat4 & transform = glm::mat4(1.0f)) [static]
```

Submits a draw call with the specified shader, vertex array, and transform.

Parameters

| | |
|--------------------|-------------------------------------|
| <i>shader</i> | The shader to use for rendering. |
| <i>vertexArray</i> | The vertex array to draw. |
| <i>transform</i> | The transformation matrix to apply. |

Todo Set up a command list to encapsulate this for draw order control

```
00040 {
00041     VZ_PROFILE_FUNCTION();
00042     shader->Bind();
00043     std::dynamic_pointer_cast<OpenGLShader>(shader)->UploadUniformMat4("u_ViewProjection",
        s_SceneData->ViewProjectionMatrix);
00044     std::dynamic_pointer_cast<OpenGLShader>(shader)->UploadUniformMat4("u_Transform", transform);
00045
00046     vertexArray->Bind();
00047     RenderCommand::DrawIndexed(vertexArray);
00048 }
```

References [s_SceneData](#).

10.55.4 Member Data Documentation

10.55.4.1 s_SceneData

```
Renderer::SceneData * Vesper::Renderer::s_SceneData = new Renderer::SceneData [static], [private]
```

Pointer to the current scene data.

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](#)

10.56 Vesper::Renderer2D Class Reference

A 2D renderer for drawing quads and sprites.

```
#include <Renderer2D.h>
```

Classes

- struct [Statistics](#)
2D Renderer Statistics

Static Public Member Functions

- static void [Init \(\)](#)
Initializes the 2D renderer.
- static void [Shutdown \(\)](#)
Shuts down the 2D renderer.
- static void [BeginScene \(const Camera &camera, const glm::mat4 &transform\)](#)
Begins a new scene with the given camera and transform.
- static void [BeginScene \(const EditorCamera &camera\)](#)
Begins a new scene with the given editor camera.
- static void [BeginScene \(const OrthographicCamera &camera\)](#)
Begins a new scene with the given orthographic camera.
- static void [EndScene \(\)](#)
Ends the current scene.
- static void [Flush \(\)](#)
Flushes the current batch of rendering commands.
- static void [DrawQuad \(const glm::mat4 &transform, const glm::vec4 &color\)](#)
Draws a colored quad with the given transform and color.
- static void [DrawQuad \(const glm::vec2 &position, const glm::vec2 &size, const glm::vec4 &color\)](#)
Draws a colored quad at the specified position and size.
- static void [DrawQuad \(const glm::vec3 &position, const glm::vec2 &size, const glm::vec4 &color\)](#)
Draws a colored quad at the specified position and size.
- static void [DrawQuadWithTexture \(const glm::mat4 &transform, const Ref< Texture2D > &texture, float tilingFactor, const glm::vec4 tintColor\)](#)
Draws a textured quad with the given transform, texture, tiling factor, and tint color.
- static void [DrawQuadWithTexture \(const glm::vec2 &position, const glm::vec2 &size, const Ref< Texture2D > &texture, float tilingFactor, const glm::vec4 tintColor\)](#)
Draws a textured quad at the specified position and size.
- static void [DrawQuadWithTexture \(const glm::vec3 &position, const glm::vec2 &size, const Ref< Texture2D > &texture, float tilingFactor, const glm::vec4 tintColor\)](#)
Draws a textured quad at the specified position and size.
- static void [DrawQuadWithTexture \(const glm::mat4 &transform, const Ref< SubTexture2D > &subtexture, float tilingFactor, const glm::vec4 tintColor\)](#)
Draws a textured quad with the given transform, subtexture, tiling factor, and tint color.
- static void [DrawQuadWithTexture \(const glm::vec2 &position, const glm::vec2 &size, const Ref< SubTexture2D > &subtexture, float tilingFactor, const glm::vec4 tintColor\)](#)
Draws a textured quad at the specified position and size.
- static void [DrawQuadWithTexture \(const glm::vec3 &position, const glm::vec2 &size, const Ref< SubTexture2D > &subtexture, float tilingFactor, const glm::vec4 tintColor\)](#)
Draws a textured quad at the specified position and size.

- static void [DrawQuadRotated](#) (const `glm::mat4 &transform`, const `glm::vec4 &color`)

Draws a textured quad at the specified position and size.
- static void [DrawQuadRotated](#) (const `glm::vec2 &position`, const `glm::vec2 &size`, float `rotationRads`, const `glm::vec4 &color`)

Draws a rotated colored quad with the given transform and color.
- static void [DrawQuadRotated](#) (const `glm::vec3 &position`, const `glm::vec2 &size`, float `rotationRads`, const `glm::vec4 &color`)

Draws a rotated colored quad at the specified position, size, and rotation.
- static void [DrawQuadRotatedWithTexture](#) (const `glm::mat4 &transform`, const `Ref< Texture2D > &texture`, float `tilingFactor`, const `glm::vec4 tintColor`)

Draws a rotated textured quad with the given transform, texture, tiling factor, and tint color.
- static void [DrawQuadRotatedWithTexture](#) (const `glm::vec2 &position`, const `glm::vec2 &size`, const `Ref< Texture2D > &texture`, float `rotationRads`, float `tilingFactor`, const `glm::vec4 tintColor`)

Draws a rotated textured quad at the specified position, size, and rotation.
- static void [DrawQuadRotatedWithTexture](#) (const `glm::vec3 &position`, const `glm::vec2 &size`, const `Ref< Texture2D > &texture`, float `rotationRads`, float `tilingFactor`, const `glm::vec4 tintColor`)

Draws a rotated textured quad at the specified position, size, and rotation.
- static void [DrawQuadRotatedWithTexture](#) (const `glm::mat4 &transform`, const `Ref< SubTexture2D > &subtexture`, float `tilingFactor`, const `glm::vec4 tintColor`)

Draws a rotated textured quad with the given transform, subtexture, tiling factor, and tint color.
- static void [DrawQuadRotatedWithTexture](#) (const `glm::vec2 &position`, const `glm::vec2 &size`, const `Ref< SubTexture2D > &subtexture`, float `rotationRads`, float `tilingFactor`, const `glm::vec4 tintColor`)

Draws a rotated textured quad at the specified position, size, and rotation.
- static void [DrawQuadRotatedWithTexture](#) (const `glm::vec3 &position`, const `glm::vec2 &size`, const `Ref< SubTexture2D > &subtexture`, float `rotationRads`, float `tilingFactor`, const `glm::vec4 tintColor`)

Draws a rotated textured quad at the specified position, size, and rotation.
- static `Ref< Texture2D > GetWhiteTexture` ()

Returns a reference to the default white texture that allows for coloring.
- static void [ResetStats](#) ()

Resets the rendering statistics.
- static `Statistics GetStats` ()

Retrieves the current rendering statistics.

Static Private Member Functions

- static void [FlushAndReset](#) ()
- static void [StartBatch](#) ()

10.56.1 Detailed Description

A 2D renderer for drawing quads and sprites.

10.56.2 Member Function Documentation

10.56.2.1 BeginScene() [1/3]

```
void Vesper::Renderer2D::BeginScene (
    const Camera & camera,
    const glm::mat4 & transform) [static]
```

Begins a new scene with the given camera and transform.

Parameters

| | |
|------------------|--------------------------------------|
| <i>camera</i> | The camera to use for the scene. |
| <i>transform</i> | The transform matrix for the camera. |

```
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](#).

10.56.2.2 BeginScene() [2/3]

```
void Vesper::Renderer2D::BeginScene (
    const EditorCamera & camera) [static]
```

Begins a new scene with the given editor camera.

Parameters

| | |
|---------------|---|
| <i>camera</i> | The editor camera to use for the scene. |
|---------------|---|

```
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\(\)](#).

10.56.2.3 BeginScene() [3/3]

```
void Vesper::Renderer2D::BeginScene (
    const OrthographicCamera & camera) [static]
```

Begins a new scene with the given orthographic camera.

Parameters

| | |
|---------------|---|
| <i>camera</i> | The orthographic camera to use for the scene. |
|---------------|---|

Todo Remove once we have a proper scene system

```
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](#).

10.56.2.4 DrawQuad() [1/3]

```
void Vesper::Renderer2D::DrawQuad (
    const glm::mat4 & transform,
    const glm::vec4 & color) [static]
```

Draws a colored quad with the given transform and color.

Parameters

| | |
|------------------|------------------------------------|
| <i>transform</i> | The transform matrix for the quad. |
| <i>color</i> | The color of the quad. |

```
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     }
```

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](#).

10.56.2.5 DrawQuad() [2/3]

```
void Vesper::Renderer2D::DrawQuad (
    const glm::vec2 & position,
    const glm::vec2 & size,
    const glm::vec4 & color) [static]
```

Draws a colored quad at the specified position and size.

Parameters

| | |
|-----------------|---------------------------------------|
| <i>position</i> | The position of the quad in 2D space. |
| <i>size</i> | The size of the quad. |
| <i>color</i> | The color of the quad. |

```
00208     {
00209         DrawQuad({ position.x, position.y, 0.0f }, size, color);
00210     }
```

10.56.2.6 DrawQuad() [3/3]

```
void Vesper::Renderer2D::DrawQuad (
    const glm::vec3 & position,
    const glm::vec2 & size,
    const glm::vec4 & color) [static]
```

Draws a colored quad at the specified position and size.

Parameters

| | |
|-----------------|---------------------------------------|
| <i>position</i> | The position of the quad in 3D space. |
| <i>size</i> | The size of the quad. |
| <i>color</i> | The color of the quad. |

```
00212     {
00213         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position) * glm::scale(glm::mat4(1.0f),
00214             { size.x, size.y, 1.0f });
00215         DrawQuad(transform, color);
00216     }
```

10.56.2.7 DrawQuadRotated() [1/3]

```
void Vesper::Renderer2D::DrawQuadRotated (
    const glm::mat4 & transform,
    const glm::vec4 & color) [static]
```

Draws a rotated colored quad with the given transform and color.

Parameters

| | |
|------------------|------------------------------------|
| <i>transform</i> | The transform matrix for the quad. |
|------------------|------------------------------------|

| | |
|--------------|------------------------|
| <i>color</i> | The color of the quad. |
|--------------|------------------------|

```

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](#).

10.56.2.8 DrawQuadRotated() [2/3]

```

void Vesper::Renderer2D::DrawQuadRotated (
    const glm::vec2 & position,
    const glm::vec2 & size,
    float rotationRads,
    const glm::vec4 & color) [static]

```

Draws a rotated colored quad at the specified position, size, and rotation.

Parameters

| | |
|---------------------|---------------------------------------|
| <i>position</i> | The position of the quad in 2D space. |
| <i>size</i> | The size of the quad. |
| <i>rotationRads</i> | The rotation of the quad in radians. |
| <i>color</i> | The color of the quad. |

```

00357  {
00358      DrawQuadRotated({ position.x, position.y, 0.0f }, size, rotationRads, color);
00359  }

```

10.56.2.9 DrawQuadRotated() [3/3]

```

void Vesper::Renderer2D::DrawQuadRotated (
    const glm::vec3 & position,
    const glm::vec2 & size,
    float rotationRads,
    const glm::vec4 & color) [static]

```

Draws a rotated colored quad at the specified position, size, and rotation.

Parameters

| | |
|---------------------|---------------------------------------|
| <i>position</i> | The position of the quad in 3D space. |
| <i>size</i> | The size of the quad. |
| <i>rotationRads</i> | The rotation of the quad in radians. |
| <i>color</i> | The color of the quad. |

```
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      }
```

10.56.2.10 DrawQuadRotatedWithTexture() [1/6]

```
void Vesper::Renderer2D::DrawQuadRotatedWithTexture (
    const glm::mat4 & transform,
    const Ref< SubTexture2D > & subtexture,
    float tilingFactor,
    const glm::vec4 tintColor) [static]
```

Draws a rotated textured quad with the given transform, subtexture, tiling factor, and tint color.

Parameters

| | |
|---------------------|---|
| <i>transform</i> | The transform matrix for the quad. |
| <i>subtexture</i> | The subtexture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
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](#).

10.56.2.11 DrawQuadRotatedWithTexture() [2/6]

```
void Vesper::Renderer2D::DrawQuadRotatedWithTexture (
    const glm::mat4 & transform,
    const Ref< Texture2D > & texture,
    float tilingFactor,
    const glm::vec4 tintColor) [static]
```

Draws a rotated textured quad with the given transform, texture, tiling factor, and tint color.

Parameters

| | |
|---------------------|---|
| <i>transform</i> | The transform matrix for the quad. |
| <i>texture</i> | The texture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
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](#).

10.56.2.12 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]
```

Draws a rotated textured quad at the specified position, size, and rotation.

Parameters

| | |
|---------------------|---|
| <i>position</i> | The position of the quad in 2D space. |
| <i>size</i> | The size of the quad. |
| <i>rotationRads</i> | The rotation of the quad in radians. |
| <i>subtexture</i> | The subtexture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
00461      {
00462          DrawQuadRotatedWithTexture({ position.x, position.y, 0.0f }, size, subtexture, rotationRads,
00463          tilingFactor, tintColor);
00464      }
```

10.56.2.13 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]
```

Draws a rotated textured quad at the specified position, size, and rotation.

Parameters

| | |
|---------------------|---|
| <i>position</i> | The position of the quad in 2D space. |
| <i>size</i> | The size of the quad. |
| <i>rotationRads</i> | The rotation of the quad in radians. |
| <i>texture</i> | The texture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
00410      {
00411          DrawQuadRotatedWithTexture({ position.x, position.y, 0.0f }, size, texture, rotationRads,
00412          tilingFactor, tintColor);
00413      }
```

10.56.2.14 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]
```

Draws a rotated textured quad at the specified position, size, and rotation.

Parameters

| | |
|---------------------|---|
| <i>position</i> | The position of the quad in 3D space. |
| <i>size</i> | The size of the quad. |
| <i>rotationRads</i> | The rotation of the quad in radians. |
| <i>subtexture</i> | The subtexture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
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     }
```

10.56.2.15 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]
```

Draws a rotated textured quad at the specified position, size, and rotation.

Parameters

| | |
|---------------------|---|
| <i>position</i> | The position of the quad in 3D space. |
| <i>size</i> | The size of the quad. |
| <i>rotationRads</i> | The rotation of the quad in radians. |
| <i>texture</i> | The texture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
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     }
```

10.56.2.16 DrawQuadWithTexture() [1/6]

```
void Vesper::Renderer2D::DrawQuadWithTexture (
    const glm::mat4 & transform,
    const Ref< SubTexture2D > & subtexture,
    float tilingFactor,
    const glm::vec4 tintColor) [static]
```

Draws a textured quad with the given transform, subtexture, tiling factor, and tint color.

Parameters

| | |
|---------------------|---|
| <i>transform</i> | The transform matrix for the quad. |
| <i>subtexture</i> | The subtexture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
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::QuadVertexBuffer](#), [Vesper::s_Data](#), [Vesper::Renderer2DData::Stats](#), [Vesper::QuadVertex::TexIndex](#), [Vesper::Renderer2DData::TextureSlotIndex](#), and [Vesper::QuadVertex::TilingFactor](#).

10.56.2.17 DrawQuadWithTexture() [2/6]

```
void Vesper::Renderer2D::DrawQuadWithTexture (
    const glm::mat4 & transform,
```

```
const Ref< Texture2D > & texture,
float tilingFactor,
const glm::vec4 tintColor) [static]
```

Draws a textured quad with the given transform, texture, tiling factor, and tint color.

Parameters

| | |
|---------------------|---|
| <i>transform</i> | The transform matrix for the quad. |
| <i>texture</i> | The texture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
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      s_Data.QuadIndexCount += 6;
00258      s_Data.Stats.QuadCount++;
00259  }
```

References [FlushAndReset\(\)](#), [Vesper::Renderer2DData::MaxIndices](#), [Vesper::Renderer2DData::MaxTextureSlots](#), [Vesper::Renderer2D::Statistics::QuadCount](#), [Vesper::Renderer2DData::QuadIndexCount](#), [Vesper::Renderer2DData::QuadVertexBuffer](#), [Vesper::s_Data](#), [Vesper::Renderer2DData::Stats](#), [Vesper::QuadVertex::TexIndex](#), [Vesper::Renderer2DData::TextureSlotIndex](#), and [Vesper::QuadVertex::TilingFactor](#).

10.56.2.18 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]
```

Draws a textured quad at the specified position and size.

Parameters

| | |
|---------------------|---|
| <i>position</i> | The position of the quad in 2D space. |
| <i>size</i> | The size of the quad. |
| <i>subtexture</i> | The subtexture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
00319     {
00320         DrawQuadWithTexture({ position.x, position.y, 0.0f }, size, subtexture, tilingFactor,
00321         tintColor);
00321     }
```

10.56.2.19 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]
```

Draws a textured quad at the specified position and size.

Parameters

| | |
|---------------------|---|
| <i>position</i> | The position of the quad in 2D space. |
| <i>size</i> | The size of the quad. |
| <i>texture</i> | The texture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```
00263     {
00264         DrawQuadWithTexture({ position.x, position.y, 0.0f }, size, texture, tilingFactor, tintColor);
00265     }
```

10.56.2.20 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]
```

Draws a textured quad at the specified position and size.

Parameters

| | |
|---------------------|---|
| <i>position</i> | The position of the quad in 3D space. |
| <i>size</i> | The size of the quad. |
| <i>subtexture</i> | The subtexture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```

00323     {
00324         VZ_PROFILE_FUNCTION();
00325
00326         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position) * glm::scale(glm::mat4(1.0f),
00327             { size.x, size.y, 1.0f });
00328         DrawQuadWithTexture(transform, subtexture, tilingFactor, tintColor);
00329     }

```

10.56.2.21 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]

```

Draws a textured quad at the specified position and size.

Parameters

| | |
|---------------------|---|
| <i>position</i> | The position of the quad in 3D space. |
| <i>size</i> | The size of the quad. |
| <i>texture</i> | The texture to apply to the quad. |
| <i>tilingFactor</i> | The tiling factor for the texture. |
| <i>tintColor</i> | The tint color to apply to the texture. |

```

00267     {
00268         VZ_PROFILE_FUNCTION();
00269
00270         glm::mat4 transform = glm::translate(glm::mat4(1.0f), position) * glm::scale(glm::mat4(1.0f),
00271             { size.x, size.y, 1.0f });
00272         DrawQuadWithTexture(transform, texture, tilingFactor, tintColor);
00273     }

```

10.56.2.22 EndScene()

```
void Vesper::Renderer2D::EndScene () [static]
```

Ends the current scene.

```

00161     {
00162         VZ_PROFILE_FUNCTION();
00163         uint32_t dataSize = (uint32_t)((uint8_t*)s_Data.QuadVertexBufferPtr -
00164             (uint8_t*)s_Data.QuadVertexBufferBase);
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\(\)](#).

10.56.2.23 Flush()

```
void Vesper::Renderer2D::Flush () [static]
```

Flushes the current batch of rendering commands.

```
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\(\)](#).

10.56.2.24 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\(\)](#).

10.56.2.25 GetStats()

```
Renderer2D::Statistics Vesper::Renderer2D::GetStats () [static]
```

Retrieves the current rendering statistics.

```
00503 {
00504     return s_Data.Stats;
00505 }
```

References [Vesper::s_Data](#), and [Vesper::Renderer2DData::Stats](#).

Referenced by [Vesper::EditorLayer::OnImGuiRender\(\)](#).

10.56.2.26 GetWhiteTexture()

```
Ref< Texture2D > Vesper::Renderer2D::GetWhiteTexture () [static]
```

Returns a reference to the default white texture that allows for coloring.

```
00493 {
00494     return s_Data.WhiteTexture;
00495 }
```

References [Vesper::s_Data](#).

10.56.2.27 Init()

```
void Vesper::Renderer2D::Init () [static]
```

Initializes the 2D renderer.

```
00056     {
00057         VZ_PROFILE_FUNCTION();
00058
00059         s_Data.QuadVertexArray = (VertexArray::Create());
00060
00061         s_Data.QuadVertexBuffer = VertexBuffer::Create(s_Data.MaxVertices * sizeof(QuadVertex));
00062
00063         s_Data.QuadVertexBuffer->setLayout({
00064             { ShaderDataType::Float3, "a_Position" },
00065             { ShaderDataType::Float4, "a_Color" },
00066             { ShaderDataType::Float2, "a_TexCoord" },
00067             { ShaderDataType::Float, "a_TexIndex" },
00068             { ShaderDataType::Float, "a_TilingFactor" }
00069         });
00070         s_Data.QuadVertexArray->addVertexBuffer(s_Data.QuadVertexBuffer);
00071
00072         s_Data.QuadVertexBufferBase = new QuadVertex[s_Data.MaxVertices];
00073
00074         uint32_t* quadIndices = new uint32_t[s_Data.MaxIndices];
00075         uint32_t offset = 0;
00076
00077         for (uint32_t i = 0; i < s_Data.MaxIndices; i += 6) {
00078             quadIndices[i + 0] = offset + 0;
00079             quadIndices[i + 1] = offset + 1;
00080             quadIndices[i + 2] = offset + 2;
00081             quadIndices[i + 3] = offset + 2;
00082             quadIndices[i + 4] = offset + 3;
00083             quadIndices[i + 5] = offset + 0;
00084
00085             offset += 4;
00086         }
00087
00088         Ref<IndexBuffer> quadIB;
00089         quadIB = (IndexBuffer::Create(quadIndices, s_Data.MaxIndices));
00090         s_Data.QuadVertexArray->setIndexBuffer(quadIB);
00091         delete[] quadIndices;
00092
00093         s_Data.WhiteTexture = Texture2D::Create(1, 1);
00094         uint32_t whiteTextureData = 0xffffffff;
00095         s_Data.WhiteTexture->setData(&whiteTextureData, sizeof(uint32_t));
00096
00097         int32_t samplers[s_Data.MaxTextureSlots];
00098         for (uint32_t i = 0; i < s_Data.MaxTextureSlots; i++)
00099             samplers[i] = i;
00100
00101         s_Data.TextureShader = Shader::Create("../Vesper-Editor/assets/shaders/Texture.glsl");
00102         s_Data.TextureShader->bind();
00103         s_Data.TextureShader->setIntArray("u_Textures", samplers, s_Data.MaxTextureSlots);
00104
00105         s_Data.TextureSlots[0] = s_Data.WhiteTexture;
00106
00107         s_Data.QuadVertexPositions[0] = { -0.5f, -0.5f, 0.0f, 1.0f };
00108         s_Data.QuadVertexPositions[1] = { 0.5f, -0.5f, 0.0f, 1.0f };
00109         s_Data.QuadVertexPositions[2] = { 0.5f, 0.5f, 0.0f, 1.0f };
00110         s_Data.QuadVertexPositions[3] = { -0.5f, 0.5f, 0.0f, 1.0f };
00111
00112     }
00113 }
```

References [Vesper::Float](#), [Vesper::Float2](#), [Vesper::Float3](#), [Vesper::Float4](#), [Vesper::Renderer2DData::MaxIndices](#), [Vesper::Renderer2DData::MaxTextureSlots](#), [Vesper::Renderer2DData::MaxVertices](#), [Vesper::Renderer2DData::QuadVertexBufferBase](#) and [Vesper::s_Data](#).

Referenced by [Vesper::Renderer::Init\(\)](#).

10.56.2.28 ResetStats()

```
void Vesper::Renderer2D::ResetStats () [static]
```

Resets the rendering statistics.

```
00498     {
00499         memset(&s_Data.Stats, 0, sizeof(Statistics));
00500     }
```

References [Vesper::s_Data](#), and [Vesper::Renderer2DData::Stats](#).

Referenced by [Vesper::EditorLayer::OnUpdate\(\)](#).

10.56.2.29 Shutdown()

```
void Vesper::Renderer2D::Shutdown () [static]
```

Shuts down the 2D renderer.

```
00117     {
00118         VZ_PROFILE_FUNCTION();
00119         delete[] s_Data.QuadVertexBufferBase;
00121     }
```

References [Vesper::Renderer2DData::QuadVertexBufferBase](#), and [Vesper::s_Data](#).

10.56.2.30 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](#)

10.57 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`

10.57.1 Class Documentation

10.57.1.1 struct Vesper::Renderer2DData::CameraData

Class Members

| | | |
|-------------------|-----------------------------|--|
| <code>mat4</code> | <code>ViewProjection</code> | |
|-------------------|-----------------------------|--|

10.57.2 Member Data Documentation

10.57.2.1 CameraBuffer

`CameraData` `Vesper::Renderer2DData::CameraBuffer`

10.57.2.2 CameraUniformBuffer

`Ref< UniformBuffer >` `Vesper::Renderer2DData::CameraUniformBuffer`

10.57.2.3 MaxIndices

`const uint32_t Vesper::Renderer2DData::MaxIndices = MaxQuads * 6 [static]`

Referenced by `Vesper::Renderer2D::DrawQuad()`, `Vesper::Renderer2D::DrawQuadRotated()`, `Vesper::Renderer2D::DrawQuadRotatedWithTexture()`, `Vesper::Renderer2D::DrawQuadWithTexture()`, `Vesper::Renderer2D::DrawQuads()` and `Vesper::Renderer2D::Init()`.

10.57.2.4 MaxQuads

```
const uint32_t Vesper::Renderer2DData::MaxQuads = 30000 [static]
```

10.57.2.5 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\(\)](#).

10.57.2.6 MaxVertices

```
const uint32_t Vesper::Renderer2DData::MaxVertices = MaxQuads * 4 [static]
```

Referenced by [Vesper::Renderer2D::Init\(\)](#).

10.57.2.7 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::DrawQuadRotatedWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::Flush\(\)](#), [Vesper::Renderer2D::FlushAndReset\(\)](#), and [Vesper::Renderer2D::StartBatch\(\)](#).

10.57.2.8 QuadVertexArray

```
Ref< VertexArray > Vesper::Renderer2DData::QuadVertexArray
```

10.57.2.9 QuadVertexBuffer

```
Ref< VertexBuffer > Vesper::Renderer2DData::QuadVertexBuffer
```

10.57.2.10 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\(\)](#).

10.57.2.11 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::DrawQuadRotatedWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::EndScene\(\)](#), [Vesper::Renderer2D::FlushAndReset\(\)](#), and [Vesper::Renderer2D::StartBatch\(\)](#).

10.57.2.12 QuadVertexPositions

```
glm::vec4 Vesper::Renderer2DData::QuadVertexPositions
```

10.57.2.13 Stats

```
Renderer2D::Statistics Vesper::Renderer2DData::Stats
```

Referenced by [Vesper::Renderer2D::DrawQuad\(\)](#), [Vesper::Renderer2D::DrawQuadRotated\(\)](#), [Vesper::Renderer2D::DrawQuadRotatedWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::Flush\(\)](#), [Vesper::Renderer2D::GetStats\(\)](#), and [Vesper::Renderer2D::ResetStats\(\)](#).

10.57.2.14 TextureShader

```
Ref< Shader > Vesper::Renderer2DData::TextureShader
```

10.57.2.15 TextureSlotIndex

```
uint32_t Vesper::Renderer2DData::TextureSlotIndex = 1
```

Referenced by [Vesper::Renderer2D::BeginScene\(\)](#), [Vesper::Renderer2D::BeginScene\(\)](#), [Vesper::Renderer2D::DrawQuadRotatedWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadRotatedWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::Flush\(\)](#), [Vesper::Renderer2D::FlushAndReset\(\)](#), and [Vesper::Renderer2D::StartBatch\(\)](#).

10.57.2.16 TextureSlots

```
std::array< Ref< Texture2D >, MaxTextureSlots > Vesper::Renderer2DData::TextureSlots
```

10.57.2.17 WhiteTexture

```
Ref< Texture2D > Vesper::Renderer2DData::WhiteTexture
```

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

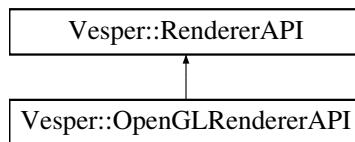
- [Vesper/src/Vesper/Renderer/Renderer2D.cpp](#)

10.58 Vesper::RendererAPI Class Reference

An abstract class defining the interface for a rendering [API](#).

```
#include <RendererAPI.h>
```

Inheritance diagram for Vesper::RendererAPI:



Public Types

- enum class [API](#) { [None](#) = 0 , [OpenGL](#) = 1 }
- API.*

Public Member Functions

- virtual [~RendererAPI](#) ()=default
- virtual void [Init](#) ()=0
Initializes the rendering API.
- virtual void [SetViewport](#) (uint32_t x, uint32_t y, uint32_t width, uint32_t height)=0
Sets the viewport dimensions.
- virtual void [SetClearColor](#) (const glm::vec4 &color)=0
Sets the clear color for the rendering API.
- virtual void [Clear](#) ()=0
Clears the rendering buffers.
- virtual void [DrawIndexed](#) (const Ref< [VertexArray](#) > &vertexArray, uint32_t indexCount=0)=0
Draws indexed geometry using the provided vertex array.

Static Public Member Functions

- static [API GetAPI](#) ()
Returns the current rendering API.

Static Private Attributes

- static [API s_API](#) = [RendererAPI::API::OpenGL](#)

10.58.1 Detailed Description

An abstract class defining the interface for a rendering [API](#).

10.58.2 Member Enumeration Documentation

10.58.2.1 API

```
enum class Vesper::RendererAPI::API [strong]
```

API.

Enumerator

| | |
|--------|--|
| None | |
| OpenGL | |

```
00014
00015     None = 0,
00016     OpenGL = 1,
00017 };
```

10.58.3 Constructor & Destructor Documentation

10.58.3.1 ~RendererAPI()

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

10.58.4 Member Function Documentation

10.58.4.1 Clear()

```
virtual void Vesper::RendererAPI::Clear () [pure virtual]
```

Clears the rendering buffers.

Implemented in [Vesper::OpenGLRendererAPI](#).

Referenced by [Vesper::RenderCommand::Clear\(\)](#).

10.58.4.2 DrawIndexed()

```
virtual void Vesper::RendererAPI::DrawIndexed (
    const Ref< VertexArray > & vertexArray,
    uint32_t indexCount = 0) [pure virtual]
```

Draws indexed geometry using the provided vertex array.

Implemented in [Vesper::OpenGLRendererAPI](#).

10.58.4.3 GetAPI()

```
API Vesper::RendererAPI::GetAPI () [inline], [static]
```

Returns the current rendering [API](#).

```
00034 { return s_API; }
```

References [s_API](#).

Referenced by [Vesper::Renderer::GetAPI\(\)](#).

10.58.4.4 Init()

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

Initializes the rendering [API](#).

Implemented in [Vesper::OpenGLRendererAPI](#).

Referenced by [Vesper::RenderCommand::Init\(\)](#).

10.58.4.5 SetClearColor()

```
virtual void Vesper::RendererAPI::SetClearColor (
    const glm::vec4 & color) [pure virtual]
```

Sets the clear color for the rendering [API](#).

Implemented in [Vesper::OpenGLRendererAPI](#).

10.58.4.6 SetViewport()

```
virtual void Vesper::RendererAPI::SetViewport (
    uint32_t x,
    uint32_t y,
    uint32_t width,
    uint32_t height) [pure virtual]
```

Sets the viewport dimensions.

Implemented in [Vesper::OpenGLRendererAPI](#).

Referenced by [Vesper::RenderCommand::SetViewport\(\)](#).

10.58.5 Member Data Documentation

10.58.5.1 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](#)

10.59 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_Visible = 160`
- `uint32_t m_VisibleHeight = 90`

Friends

- class `Entity`
- class `SceneSerializer`
- class `SceneHierarchyPanel`

10.59.1 Constructor & Destructor Documentation

10.59.1.1 Scene() [1/2]

```
Vesper::Scene::Scene ()
00012     : m_Name("Untitled Scene")
00013 {
00014     VZ_PROFILE_FUNCTION();
00015 }
00016 }
```

References [Scene\(\)](#).

Referenced by [Scene\(\)](#).

10.59.1.2 Scene() [2/2]

```
Vesper::Scene::Scene (
    const std::string & name)
00019     : m_Name(name)
00020 {
00021 }
```

References [Scene\(\)](#).

Referenced by [Scene\(\)](#).

10.59.1.3 ~Scene()

```
Vesper::Scene::~Scene ()
00024 {
00025 }
```

10.59.2 Member Function Documentation

10.59.2.1 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)) :
00044         name;
00045     entity.AddComponent<UUIDComponent>(uuid);
00046     return entity;
00047 }
```

10.59.2.2 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)) :
00033         name;
00034     entity.AddComponent<UUIDComponent>();
00035     return entity;
00036 }

```

10.59.2.3 DestroyEntity()

```

void Vesper::Scene::DestroyEntity (
    Entity entity)
00050 {
00051     m_Registry.destroy(entity);
00052 }

```

10.59.2.4 GetName()

```

const std::string & Vesper::Scene::GetName () const [inline], [private]
00043 { return m_Name; }

```

10.59.2.5 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 }

```

10.59.2.6 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](#).

10.59.2.7 OnComponentAdded() [2/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    CameraComponent & component) [private]
00244     component.Camera.SetViewportSize(m_ViewportWidth, m_ViewportHeight);
00245 }
00246 }
```

10.59.2.8 OnComponentAdded() [3/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    NameComponent & component) [private]
00236 }
00237 }
```

10.59.2.9 OnComponentAdded() [4/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    NameComponent & component) [private]
00236 }
00237 }
```

10.59.2.10 OnComponentAdded() [5/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    NativeScriptComponent & component) [private]
00265 {
00266 }
```

10.59.2.11 OnComponentAdded() [6/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    NativeScriptComponent & component) [private]
00265 {
00266 }
```

10.59.2.12 OnComponentAdded() [7/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    SpriteRendererComponent & component) [private]
00249
{
00250 }
```

10.59.2.13 OnComponentAdded() [8/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    SpriteRendererComponent & component) [private]
00249
{
00250 }
```

10.59.2.14 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 }
```

10.59.2.15 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 }
```

10.59.2.16 OnComponentAdded() [11/17]

```
template<typename T>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    T & component) [private]
00226
00227     static_assert(false);
00228 }
```

10.59.2.17 OnComponentAdded() [12/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    TextureAnimationComponent & component) [private]
00260
{
00261
00262 }
```

10.59.2.18 OnComponentAdded() [13/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    TextureAnimationComponent & component) [private]
00260
{
00261
00262 }
```

10.59.2.19 OnComponentAdded() [14/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    TransformComponent & component) [private]
00240
00241 }
```

10.59.2.20 OnComponentAdded() [15/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    TransformComponent & component) [private]
00240
00241 }
```

10.59.2.21 OnComponentAdded() [16/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    UUIDComponent & component) [private]
00231
00232     // TODO: search registry to ensure unique UUID
00233 }
```

10.59.2.22 OnComponentAdded() [17/17]

```
template<>
void Vesper::Scene::OnComponentAdded (
    Entity entity,
    UUIDComponent & component) [private]
00231
00232     // TODO: search registry to ensure unique UUID
00233 }
```

10.59.2.23 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         if (sprite.TextureEnabled && !sprite.Texture)
00164             Renderer2D::DrawQuadWithTexture(transform.GetTransform(), VZ_DEFAULT_TEXTURE,
00165             sprite.TilingFactor, sprite.Color);
00166         else if (sprite.TextureEnabled && sprite.Texture)
00167             Renderer2D::DrawQuadWithTexture(transform.GetTransform(), sprite.Texture,
00168             sprite.TilingFactor, sprite.Color);
00169         else
00170             Renderer2D::DrawQuad(transform.GetTransform(), sprite.Color);
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(),
00189     sprite.TilingFactor, sprite.Color);
00190     Renderer2D::EndScene();
00191 }
```

References [Vesper::Renderer2D::BeginScene\(\)](#), and [Vesper::Renderer2D::EndScene\(\)](#).

10.59.2.24 OnUpdateRuntime()

```
void Vesper::Scene::OnUpdateRuntime (
    Timestep ts)
```

TODO: Move to Scene::OnScenePlay()

```
00056 {
00057     VZ_PROFILE_FUNCTION();
```

```

00058     // Update scripts
00059     {
00060         m_Registry.view<NativeScriptComponent>().each([=] (auto entity, NativeScriptComponent& nsc)
00061         {
00062             if (!nsc.Instance)
00063             {
00064                 nsc.Instance = nsc.InstantiateScript();
00065                 nsc.Instance->m_Entity = Entity{ entity, this };
00066                 nsc.Instance->OnCreate();
00067             }
00068             nsc.Instance->OnUpdate(ts);
00069         });
00070     }
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,
00108             sprite.TilingFactor, sprite.Color);
00109         else if (sprite.TextureEnabled && sprite.Texture)
00110             Renderer2D::DrawQuadWithTexture(transform.GetTransform(), sprite.Texture,
00111             sprite.TilingFactor, sprite.Color);
00112         else
00113             Renderer2D::DrawQuad(transform.GetTransform(), sprite.Color);
00114     }
00115
00116     auto subTextureView = m_Registry.group<SubTextureComponent>();
00117     for (auto entity : subTextureView) {
00118         auto& transform = m_Registry.get<TransformComponent>(entity);
00119         auto& sprite = m_Registry.get<SpriteRendererComponent>(entity);
00120         auto& subTexture = m_Registry.get<SubTextureComponent>(entity);
00121
00122         if (!sprite.TextureEnabled) sprite.TextureEnabled = true;
00123         if (subTexture.SubTexture == nullptr)
00124         {
00125             subTexture.setTexture(VZ_DEFAULT_TEXTURE);
00126         }
00127         else {
00128             // Ensure the subtexture's texture matches the sprite's texture
00129             if (sprite.Texture && subTexture.SubTexture->GetTexture() != sprite.Texture) {
00130                 subTexture.setTexture(sprite.Texture);
00131             }
00132         }
00133         Renderer2D::DrawQuadWithTexture(transform.GetTransform(), subTexture.GetSubTexture(),
00134             sprite.TilingFactor, sprite.Color);
00135     }
00136
00137     //auto group1 = m_Registry.group<TextureAnimationComponent>();
00138     //for (auto entity : group1)
00139     //{
00140         // auto& transform = m_Registry.get<TransformComponent>(entity);
00141         // auto& texAnim = m_Registry.get<TextureAnimationComponent>(entity);
00142         // auto& sprite = m_Registry.get<SpriteRendererComponent>(entity);
00143         // texAnim.Update(ts.GetSeconds());
00144

```

```

00143     // Renderer2D::DrawQuadWithTexture(transform.GetTransform(),
00144     texAnim.SubTextures[texAnim.CurrentFrame], 1.0f, sprite.Color);
00145     //}
00146     Renderer2D::EndScene();
00147 }
00148 }
```

References [Vesper::Renderer2D::EndScene\(\)](#).

10.59.2.25 OnViewportResize()

```

void Vesper::Scene::OnViewportResize (
    uint32_t width,
    uint32_t height)
{
    m_ViewportWidth = width;
    m_ViewportHeight = height;

    // resize non fixed aspect ratio cameras
    auto view = m_Registry.view<CameraComponent>();
    for (auto entity : view)
    {
        auto& cameraComponent = view.get<CameraComponent>(entity);
        if (!cameraComponent.FixedAspectRatio)
            cameraComponent.Camera.SetViewportSize(width, height);
    }
}
```

References [m_ViewportHeight](#), and [m_ViewportWidth](#).

10.59.2.26 SetName()

```

void Vesper::Scene::SetName (
    const std::string & name) [inline], [private]

TODO: friend class SceneCamera; TODO: friend class SceneRenderer;
00042 { m_Name = name; }
```

10.59.3 Friends And Related Symbol Documentation

10.59.3.1 Entity

```
friend class Entity [friend]
```

10.59.3.2 SceneHierarchyPanel

```
friend class SceneHierarchyPanel [friend]
```

10.59.3.3 SceneSerializer

```
friend class SceneSerializer [friend]
```

10.59.4 Member Data Documentation

10.59.4.1 m_Name

```
std::string Vesper::Scene::m_Name [private]
```

10.59.4.2 m_Registry

```
entt::registry Vesper::Scene::m_Registry [private]
```

10.59.4.3 m_ViewportHeight

```
uint32_t Vesper::Scene::m_ViewportHeight = 90 [private]
```

Referenced by [OnComponentAdded\(\)](#), and [OnViewportResize\(\)](#).

10.59.4.4 m_ViewportWidth

```
uint32_t Vesper::Scene::m_ViewportWidth = 160 [private]
```

Referenced by [OnComponentAdded\(\)](#), and [OnViewportResize\(\)](#).

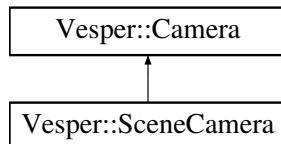
The documentation for this class was generated from the following files:

- Vesper/src/Vesper/Scene/[Scene.h](#)
- Vesper/src/Vesper/Scene/[Scene.cpp](#)

10.60 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)`

10.60.1 Member Enumeration Documentation

10.60.1.1 ProjectionType

```
enum class Vesper::SceneCamera::ProjectionType [strong]
```

Enumerator

| | |
|--------------|--|
| Perspective | |
| Orthographic | |

```
00009 { Perspective = 0, Orthographic = 1 };
```

10.60.2 Constructor & Destructor Documentation

10.60.2.1 SceneCamera()

```
Vesper::SceneCamera::SceneCamera ()
00010     {
00011         RecalculateProjection();
00012     }
```

References [RecalculateProjection\(\)](#).

10.60.2.2 ~SceneCamera()

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

10.60.3 Member Function Documentation

10.60.3.1 GetOrthographicFarClip()

```
float Vesper::SceneCamera::GetOrthographicFarClip () const [inline]
00030 { return m_OrthographicFar; }
```

References [m_OrthographicFar](#).

10.60.3.2 GetOrthographicNearClip()

```
float Vesper::SceneCamera::GetOrthographicNearClip () const [inline]
00028 { return m_OrthographicNear; }
```

References [m_OrthographicNear](#).

10.60.3.3 GetOrthographicSize()

```
float Vesper::SceneCamera::GetOrthographicSize () const [inline]
00026 { return m_OrthographicSize; }
```

References [m_OrthographicSize](#).

10.60.3.4 GetPerspectiveFarClip()

```
float Vesper::SceneCamera::GetPerspectiveFarClip () const [inline]
00023 { return m_PerspectiveFar; }
```

References [m_PerspectiveFar](#).

10.60.3.5 GetPerspectiveNearClip()

```
float Vesper::SceneCamera::GetPerspectiveNearClip () const [inline]
00021 { return m_PerspectiveNear; }
```

References [m_PerspectiveNear](#).

10.60.3.6 GetPerspectiveVerticalFOV()

```
float Vesper::SceneCamera::GetPerspectiveVerticalFOV () const [inline]
00019 { return m_PerspectiveFOV; }
```

10.60.3.7 GetProjectionType()

```
ProjectionType Vesper::SceneCamera::GetProjectionType () const [inline]
00033 { return m_ProjectionType; }
```

References [m_ProjectionType](#).

10.60.3.8 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,
00042                                             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     }
```

References [m_AspectRatio](#), [m_OrthographicSize](#), [m_ProjectionType](#), and [Perspective](#).

Referenced by [SceneCamera\(\)](#), [SetOrthographic\(\)](#), [SetOrthographicFarClip\(\)](#), [SetOrthographicNearClip\(\)](#), [SetOrthographicSize\(\)](#), [SetPerspective\(\)](#), [SetPerspectiveFarClip\(\)](#), [SetPerspectiveNearClip\(\)](#), [SetPerspectiveVerticalFOV\(\)](#), [SetProjectionType\(\)](#), and [SetViewportSize\(\)](#).

10.60.3.9 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\(\)](#).

10.60.3.10 SetOrthographicFarClip()

```
void Vesper::SceneCamera::SetOrthographicFarClip (
    float farClip) [inline]
00031 { m_OrthographicFar = farClip; RecalculateProjection(); }
```

References [m_OrthographicFar](#), and [RecalculateProjection\(\)](#).

10.60.3.11 SetOrthographicNearClip()

```
void Vesper::SceneCamera::SetOrthographicNearClip (
    float nearClip) [inline]
00029 { m_OrthographicNear = nearClip; RecalculateProjection(); }
```

References [m_OrthographicNear](#), and [RecalculateProjection\(\)](#).

10.60.3.12 SetOrthographicSize()

```
void Vesper::SceneCamera::SetOrthographicSize (
    float size) [inline]
00027 { m_OrthographicSize = size; RecalculateProjection(); }
```

References [m_OrthographicSize](#), and [RecalculateProjection\(\)](#).

10.60.3.13 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\(\)](#).

10.60.3.14 SetPerspectiveFarClip()

```
void Vesper::SceneCamera::SetPerspectiveFarClip (
    float farClip) [inline]
00024 { m_PerspectiveFar = farClip; RecalculateProjection(); }
```

References [m_PerspectiveFar](#), and [RecalculateProjection\(\)](#).

10.60.3.15 SetPerspectiveNearClip()

```
void Vesper::SceneCamera::SetPerspectiveNearClip (
    float nearClip) [inline]
00022 { m_PerspectiveNear = nearClip; RecalculateProjection(); }
```

References [m_PerspectiveNear](#), and [RecalculateProjection\(\)](#).

10.60.3.16 SetPerspectiveVerticalFOV()

```
void Vesper::SceneCamera::SetPerspectiveVerticalFOV (
    float verticalFov) [inline]
00020 { m_PerspectiveFOV = verticalFov; RecalculateProjection(); }
```

References [RecalculateProjection\(\)](#).

10.60.3.17 SetProjectionType()

```
void Vesper::SceneCamera::SetProjectionType (
    ProjectionType type) [inline]
00034 { m_ProjectionType = type; RecalculateProjection(); }
```

References [m_ProjectionType](#), and [RecalculateProjection\(\)](#).

10.60.3.18 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\(\)](#).

10.60.4 Member Data Documentation

10.60.4.1 m_AspectRatio

```
float Vesper::SceneCamera::m_AspectRatio = 0.0f [private]
```

Referenced by [RecalculateProjection\(\)](#), and [SetViewportSize\(\)](#).

10.60.4.2 m_OrthographicFar

```
float Vesper::SceneCamera::m_OrthographicFar = 1.0f [private]
```

Referenced by [GetOrthographicFarClip\(\)](#), [SetOrthographic\(\)](#), and [SetOrthographicFarClip\(\)](#).

10.60.4.3 m_OrthographicNear

```
float Vesper::SceneCamera::m_OrthographicNear = -1.0f [private]
```

Referenced by [GetOrthographicNearClip\(\)](#), [SetOrthographic\(\)](#), and [SetOrthographicNearClip\(\)](#).

10.60.4.4 m_OrthographicSize

```
float Vesper::SceneCamera::m_OrthographicSize = 10.0f [private]
```

Referenced by [GetOrthographicSize\(\)](#), [RecalculateProjection\(\)](#), [SetOrthographic\(\)](#), and [SetOrthographicSize\(\)](#).

10.60.4.5 m_PerspectiveFar

```
float Vesper::SceneCamera::m_PerspectiveFar = 1000.0f [private]
```

Referenced by [GetPerspectiveFarClip\(\)](#), [SetPerspective\(\)](#), and [SetPerspectiveFarClip\(\)](#).

10.60.4.6 m_PerspectiveFOV

```
float Vesper::SceneCamera::m_PerspectiveFOV = glm::radians(45.0f) [private]
```

10.60.4.7 m_PerspectiveNear

```
float Vesper::SceneCamera::m_PerspectiveNear = 0.01f [private]
```

Referenced by [GetPerspectiveNearClip\(\)](#), [SetPerspective\(\)](#), and [SetPerspectiveNearClip\(\)](#).

10.60.4.8 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](#)

10.61 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`

10.61.1 Constructor & Destructor Documentation

10.61.1.1 SceneHierarchyPanel() [1/2]

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

10.61.1.2 SceneHierarchyPanel() [2/2]

```
Vesper::SceneHierarchyPanel::SceneHierarchyPanel (
    const Ref< Scene > & context)
00017     {
00018         SetContext (context);
00019     }
```

10.61.2 Member Function Documentation

10.61.2.1 DisplayAddComponentEntry()

```
template<typename T>
void Vesper::SceneHierarchyPanel::DisplayAddComponentEntry (
    const std::string & entryName) [private]
00545     if (!m_SelectionContext.GetComponent<T>())
00546     {
00547         if (ImGui::MenuItem(entryName.c_str()))
00548         {
00549             m_SelectionContext.AddComponent<T>();
00550             ImGui::CloseCurrentPopup();
00551         }
00552     }
00553 }
```

10.61.2.2 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 =
00422             projectionTypeStrings[(int)camera.GetProjectionType()];
00423         if (ImGui::BeginCombo("Projection", currentProjectionTypeString))
00424         {
00425             for (int i = 0; i < 2; i++)
00426             {
00427                 if (ImGui::Selectable(projectionTypeStrings[i]))
00428                     component.ProjectionType = i;
00429             }
00430         }
00431     });
00432 }
```

```

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
00482     // texture
00483     if (component.Texture)
00484     {
00485         ImGui::TextUnformatted(component.Texture->GetName().c_str());
00486         ImGui::SameLine();
00487         if (ImGui::Button("Change Texture"))
00488         {
00489             std::string filePath = FileDialogs::OpenFile("Image Files
00490             (*.png;*.jpg;*.jpeg;*.bmp;*.tga)\0*.png;*.jpg;*.jpeg;*.bmp;*.tga\0All Files (*.*)\0*\.\0");
00491             if (!filePath.empty())
00492             {
00493                 auto tex = Texture2D::Create(filePath);
00494                 if (tex)
00495                 {
00496                     component.Texture = tex;
00497                     component.TextureEnabled = true;
00498                 }
00499                 else
00500                 {
00501                     VZ_WARN("Could not load texture {0}", filePath);
00502                 }
00503             }
00504             ImGui::SameLine();
00505             if (ImGui::Button("Clear Texture"))
00506             {
00507                 component.Texture = nullptr;
00508                 component.TextureEnabled = false;
00509             }
00510         }
00511     }
00512 });

```

```

00511             {
00512                 ImGui::SameLine();
00513                 if (ImGui::Button("Set Texture"))
00514                 {
00515                     std::string filePath = FileDialogs::OpenFile("Image Files
00516 (*.png;*.jpg;*.jpeg;*.bmp;*.tga)\0*.png;*.jpg;*.jpeg;*.bmp;*.tga\0All Files (*.*)\0*\0");
00517                     if (!filePath.empty())
00518                     {
00519                         auto tex = Texture2D::Create(filePath);
00520                         if (tex)
00521                             {
00522                                 component.Texture = tex;
00523                                 component.TextureEnabled = true;
00524                             }
00525                         else
00526                             {
00527                                 VZ_WARN("Could not load texture {0}", filePath);
00528                             }
00529                     }
00530                 }
00531             }
00532             ImGui::DragFloat("Tiling Factor", &component.TilingFactor, 0.1f, 0.0f, 100.0f);
00533         });
00535     DrawComponent<SubTextureComponent>("SubTexture", entity, [](auto& component)
00536     {
00537         SubTextureEdit(component.SubTexture->GetTexture()->GetName(), component);
00538     });
00540 }
00541 }
```

10.61.2.3 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)
00073         | ImGuiTreeNodeFlags_OpenOnArrow;
00074         flags |= ImGuiTreeNodeFlags_SpanAvailWidth;
00075         bool opened = ImGui::TreeNodeEx(nodeID, flags, name.c_str());
00076         if (ImGui::IsItemClicked())
00077         {
00078             m_SelectionContext = entity;
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                 {
00092                     auto& src = entity.GetComponent<NameComponent>();
00093                     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             }
00109         }
00110     }
00111 }
```

```

00107
00108      }
00109      if (entity.HasComponent<SpriteRendererComponent>())
00110      {
00111          auto& src = entity.GetComponent<SpriteRendererComponent>();
00112          newEntity.AddComponent<SpriteRendererComponent>(src);
00113      }
00114      if (entity.HasComponent<CameraComponent>())
00115      {
00116          auto& cc = entity.GetComponent<CameraComponent>();
00117          newEntity.AddComponent<CameraComponent>(cc);
00118      }
00119      if (entity.HasComponent<NativeScriptComponent>())
00120      {
00121          auto& nsc = entity.GetComponent<NativeScriptComponent>();
00122          newEntity.AddComponent<NativeScriptComponent>(nsc);
00123      }
00124      if (entity.HasComponent<TextureAnimationComponent>())
00125      {
00126          auto& tac = entity.GetComponent<TextureAnimationComponent>();
00127          newEntity.AddComponent<TextureAnimationComponent>(tac);
00128      }
00129      // Add more components as needed
00130      m_SelectionContext = newEntity;
00131
00132      ImGui::EndPopup();
00133  }
00134
00135  if (opened)
00136  {
00137      ImGuiTreeNodeFlags flags = ImGuiTreeNodeFlags_OpenOnArrow | 
00138      ImGuiTreeNodeFlags_SpanAvailWidth;
00139      bool opened = ImGui::TreeNodeEx((void*)9817239, flags, name.c_str());
00140      if (opened)
00141      {
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 }

```

10.61.2.4 GetSelectedEntity()

```

Entity Vesper::SceneHierarchyPanel::GetSelectedEntity () const [inline]
00021 { return m_SelectionContext; }

```

10.61.2.5 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::BeginPopupContextMenuWindow(0, ImGuiPopupFlags_NoOpenOverItems |
00044              ImGuiPopupFlags_MouseButtonRight))
00045          {
00046              if (ImGui::MenuItem("Create Empty Entity"))
00047                  m_SelectionContext = m_Context->CreateEntity("Empty Entity");
00048

```

```

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 }
```

10.61.2.6 SetContext()

```

void Vesper::SceneHierarchyPanel::SetContext (
    const Ref< Scene > & context)
{
    m_Context = context;
    m_SelectionContext = {};
}
```

10.61.2.7 SetSelectedEntity()

```

void Vesper::SceneHierarchyPanel::SetSelectedEntity (
    Entity entity)
{
    m_SelectionContext = entity;
}
```

10.61.3 Member Data Documentation

10.61.3.1 m_Context

`Ref<Scene> Vesper::SceneHierarchyPanel::m_Context [private]`

10.61.3.2 m_Framebuffer

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

10.61.3.3 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](#)

10.62 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](#)

10.62.1 Constructor & Destructor Documentation

10.62.1.1 SceneSerializer()

```
Vesper::SceneSerializer::SceneSerializer (
    const Ref< Scene > & scene)
0017     : m_Scene(scene)
0018     {
0019 }
```

References [SceneSerializer\(\)](#).

Referenced by [SceneSerializer\(\)](#).

10.62.2 Member Function Documentation

10.62.2.1 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 serializedEntity = m_Scene->CreateEntity(name, uuid);
00231             YAML::Node transformNode = entityNode["TransformComponent"];
00232             if (transformNode)
00233             {
00234                 auto& tc = serializedEntity.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 = serializedEntity.AddComponent<CameraComponent>();
00243             auto& camera = cameraComp.Camera;
00244             YAML::Node camProps = cameraNode["Camera"];
00245             camera.SetOrthographic(camProps["OrthographicSize"].as<float>(),
00246             camProps["OrthographicNear"].as<float>(), camProps["OrthographicFar"].as<float>());
00247             camera.SetPerspective(camProps["PerspectiveFOV"].as<float>(),
00248             camProps["PerspectiveNear"].as<float>(), camProps["PerspectiveFar"].as<float>());
00249             cameraComp.Primary = cameraNode["Primary"].as<bool>();
00250
00251             camera.SetProjectionType((SceneCamera::ProjectionType)cameraNode["ProjectionType"].as<int>());
00252             cameraComp.FixedAspectRatio = cameraNode["FixedAspectRatio"].as<bool>();
00253         }
00254         YAML::Node spriteNode = entityNode["SpriteRendererComponent"];
00255         if (spriteNode)
00256         {
00257             auto& src = serializedEntity.AddComponent<SpriteRendererComponent>();
00258             src.Color = spriteNode["Color"].as<glm::vec4>();
00259         }
00260     }
00261     return true;
00262 }
```

Referenced by [Vesper::EditorLayer::OnAttach\(\)](#), and [Vesper::EditorLayer::OpenScene\(\)](#).

10.62.2.2 DeserializeRuntime()

```

bool Vesper::SceneSerializer::DeserializeRuntime (
    const std::string & filepath)
00264 {
00265     VZ_CORE_ASSERT(false, "Not implemented");
00266     return false;
00267 }
```

10.62.2.3 Serialize()

```

void Vesper::SceneSerializer::Serialize (
    const std::string & filepath)
00179     YAML::Emitter out;
00180     out << YAML::BeginMap; // Scene
00181     out << YAML::Key << "Scene" << YAML::Value << m_Scene->GetName();
00182     out << YAML::Key << "Entities" << YAML::Value << YAML::BeginSeq; // Entities
00183     m_Scene->m_Registry.view<entt::entity>().each([&](auto entityID) {
00184
00185         Entity entity = { entityID, m_Scene.get() };
00186         if (!entity)
00187             return;
00188
00189         SerializeEntity(out, entity);
00190     });
00191
00192     out << YAML::EndSeq; // Entities
00193     out << YAML::EndMap; // Scene
00194
00195     std::ofstream fout(filepath);
00196     fout << out.c_str();
00197 }
```

Referenced by [Vesper::EditorLayer::SaveSceneAs\(\)](#).

10.62.2.4 SerializeRuntime()

```
void Vesper::SceneSerializer::SerializeRuntime (
    const std::string & filepath)
00202 {
00203     VZ_CORE_ASSERT(false, "Not implemented");
00204 }
```

10.62.3 Member Data Documentation

10.62.3.1 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

10.63 Vesper::ScriptableEntity Class Reference

Base class for scriptable entities within a scene.

```
#include <ScriptableEntity.h>
```

Public Member Functions

- virtual `~ScriptableEntity ()`
- template<typename T>
`T & GetComponent ()`

Retrieves a reference to a component of type T attached to the entity.

Protected Member Functions

- virtual void `OnCreate ()`
Sets the entity associated with this scriptable entity.
- virtual void `OnDestroy ()`
Called when the entity is destroyed.
- virtual void `OnUpdate (Timestep ts)`
Called every frame to update the entity.

Private Attributes

- Entity `m_Entity`

Friends

- class [Scene](#)

10.63.1 Detailed Description

Base class for scriptable entities within a scene.

A [ScriptableEntity](#) allows for custom behavior to be defined for entities in a scene, not already provided as standardized components.

10.63.2 Constructor & Destructor Documentation

10.63.2.1 ~ScriptableEntity()

```
virtual Vesper::ScriptableEntity::~ScriptableEntity () [inline], [virtual]  
00013 {};
```

10.63.3 Member Function Documentation

10.63.3.1 GetComponent()

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

Retrieves a reference to a component of type T attached to the entity.

Template Parameters

| | |
|----------|--|
| <i>T</i> | The type of the component to retrieve. |
|----------|--|

Returns

Reference to the component of type T. Will assert if the component does not exist.

```
00022 {  
00023     return m_Entity.GetComponent<T>();  
00024 }
```

10.63.3.2 OnCreate()

```
virtual void Vesper::ScriptableEntity::OnCreate () [inline], [protected], [virtual]
```

Sets the entity associated with this scriptable entity.

Meant for internal use by the [Scene](#) class. Should be overridden in derived classes if additional setup is required when the entity is set.

```
00031 {}
```

10.63.3.3 OnDestroy()

```
virtual void Vesper::ScriptableEntity::OnDestroy () [inline], [protected], [virtual]
```

Called when the entity is destroyed.

Meant for internal use by the [Scene](#) class. Should be overridden in derived classes to handle cleanup.

```
00037 {}
```

10.63.3.4 OnUpdate()

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

Called every frame to update the entity.

Parameters

| | |
|-----------|--|
| <i>ts</i> | The timestep representing the time elapsed since the last update. Custom behavior should be implemented here in derived classes. |
|-----------|--|

```
00043 {}
```

10.63.4 Friends And Related Symbol Documentation

10.63.4.1 Scene

```
friend class Scene [friend]
```

10.63.5 Member Data Documentation

10.63.5.1 m_Entity

```
Entity Vesper::ScriptableEntity::m_Entity [private]
```

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

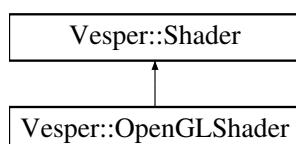
- Vesper/src/Vesper/Scene/[ScriptableEntity.h](#)

10.64 Vesper::Shader Class Reference

An abstraction for a shader program.

```
#include <Shader.h>
```

Inheritance diagram for Vesper::Shader:



Public Member Functions

- virtual [~Shader](#) ()=default
- virtual void [Bind](#) () const =0
 - connects the shader program for use.*
- virtual void [Unbind](#) () const =0
 - disconnects the shader program.*
- virtual void [SetMat4](#) (const std::string &name, const glm::mat4 &value)=0
 - Sets a 4x4 matrix uniform in the shader.*
- virtual void [SetFloat4](#) (const std::string &name, const glm::vec4 &value)=0
 - Sets a 4-component float vector uniform in the shader.*
- virtual void [SetFloat3](#) (const std::string &name, const glm::vec3 &value)=0
 - Sets a 3-component float vector uniform in the shader.*
- virtual void [SetFloat](#) (const std::string &name, float value)=0
 - Sets a single float uniform in the shader.*
- virtual void [SetInt](#) (const std::string &name, int value)=0
 - Sets a single integer uniform in the shader.*
- virtual void [SetIntArray](#) (const std::string &name, int *values, uint32_t count)=0
 - Sets an array of integers uniform in the shader.*
- 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)

10.64.1 Detailed Description

An abstraction for a shader program.

10.64.2 Constructor & Destructor Documentation

10.64.2.1 [~Shader\(\)](#)

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

10.64.3 Member Function Documentation

10.64.3.1 [Bind\(\)](#)

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

connects the shader program for use.

Implemented in [Vesper::OpenGLShader](#).

10.64.3.2 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
00014             supported!"); return nullptr;
00015         case RendererAPI::API::OpenGL: return CreateRef<OpenGLShader>(filepath);
00016     }
00017     VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00018     return nullptr;
}

```

References [Vesper::Renderer::GetAPI\(\)](#), [Vesper::RendererAPI::None](#), and [Vesper::RendererAPI::OpenGL](#).

10.64.3.3 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
00025             supported!"); return nullptr;
00026         case RendererAPI::API::OpenGL: return CreateRef<OpenGLShader>(name, vertexSrc,
00027             fragmentSrc);
00028     }
00029     VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00030     return nullptr;
}

```

References [Vesper::Renderer::GetAPI\(\)](#), [Vesper::RendererAPI::None](#), and [Vesper::RendererAPI::OpenGL](#).

10.64.3.4 GetName()

```
virtual const std::string & Vesper::Shader::GetName () const [pure virtual]
```

Implemented in [Vesper::OpenGLShader](#).

10.64.3.5 SetFloat()

```

virtual void Vesper::Shader::SetFloat (
    const std::string & name,
    float value) [pure virtual]

```

Sets a single float uniform in the shader.

Implemented in [Vesper::OpenGLShader](#).

10.64.3.6 SetFloat3()

```

virtual void Vesper::Shader::SetFloat3 (
    const std::string & name,
    const glm::vec3 & value) [pure virtual]

```

Sets a 3-component float vector uniform in the shader.

Implemented in [Vesper::OpenGLShader](#).

10.64.3.7 SetFloat4()

```
virtual void Vesper::Shader::SetFloat4 (
    const std::string & name,
    const glm::vec4 & value) [pure virtual]
```

Sets a 4-component float vector uniform in the shader.

Implemented in [Vesper::OpenGLShader](#).

10.64.3.8 SetInt()

```
virtual void Vesper::Shader::SetInt (
    const std::string & name,
    int value) [pure virtual]
```

Sets a single integer uniform in the shader.

Implemented in [Vesper::OpenGLShader](#).

10.64.3.9 SetIntArray()

```
virtual void Vesper::Shader::SetIntArray (
    const std::string & name,
    int * values,
    uint32_t count) [pure virtual]
```

Sets an array of integers uniform in the shader.

Implemented in [Vesper::OpenGLShader](#).

10.64.3.10 SetMat4()

```
virtual void Vesper::Shader::SetMat4 (
    const std::string & name,
    const glm::mat4 & value) [pure virtual]
```

Sets a 4x4 matrix uniform in the shader.

Implemented in [Vesper::OpenGLShader](#).

10.64.3.11 Unbind()

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

disconnects the shader program.

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](#)

10.65 Vesper::ShaderLibrary Class Reference

A library for managing and storing shaders.

```
#include <Shader.h>
```

Public Member Functions

- void [Add](#) (const std::string &name, const [Ref< Shader >](#) &shader)
Adds a shader to the library with the specified name.
- void [Add](#) (const [Ref< Shader >](#) &shader)
Adds a shader to the library using its own name.
- [Ref< Shader >](#) [Load](#) (const std::string &filepath)
Loads a shader from the specified filepath and adds it to the library.
- [Ref< Shader >](#) [Load](#) (const std::string &name, const std::string &filepath)
Loads a shader from the specified filepath and adds it to the library with the given name.
- [Ref< Shader >](#) [Get](#) (const std::string &name)
Retrieves a shader from the library by name.
- bool [Exists](#) (const std::string &name) const
Checks if a shader with the specified name exists in the library.

Private Attributes

- std::unordered_map< std::string, [Ref< Shader >](#) > [m_Shaders](#)

10.65.1 Detailed Description

A library for managing and storing shaders.

10.65.2 Member Function Documentation

10.65.2.1 Add() [1/2]

```
void Vesper::ShaderLibrary::Add (
    const Ref< Shader > & shader)
```

Adds a shader to the library using its own name.

```
00039  {
00040      VZ\_PROFILE\_FUNCTION\(\);
00041      auto& name = shader->GetName\(\);
00042      Add\(name, shader\);
00043 }
```

10.65.2.2 Add() [2/2]

```
void Vesper::ShaderLibrary::Add (
    const std::string & name,
    const Ref< Shader > & shader)
```

Adds a shader to the library with the specified name.

```
00032  {
00033      VZ\_PROFILE\_FUNCTION\(\);
00034      VZ\_CORE\_ASSERT\(!Exists\(name\), "Shader already exists!"\);
00035      m\_Shaders\[name\] = shader;
00036 }
```

10.65.2.3 Exists()

```
bool Vesper::ShaderLibrary::Exists (
    const std::string & name) const
```

Checks if a shader with the specified name exists in the library.

```
00068 {
00069     VZ_PROFILE_FUNCTION();
00070     return m_Shaders.find(name) != m_Shaders.end();
00071 }
```

10.65.2.4 Get()

```
Vesper::Ref< Vesper::Shader > Vesper::ShaderLibrary::Get (
    const std::string & name)
```

Retrieves a shader from the library by name.

```
00062 {
00063     VZ_CORE_ASSERT(Exists(name), "Shader not found!");
00064     return m_Shaders[name];
00065 }
```

10.65.2.5 Load() [1/2]

```
Vesper::Ref< Vesper::Shader > Vesper::ShaderLibrary::Load (
    const std::string & filepath)
```

Loads a shader from the specified filepath and adds it to the library.

```
00046 {
00047     VZ_PROFILE_FUNCTION();
00048     auto shader = Shader::Create(filepath);
00049     Add(Ref<Shader>(shader));
00050     return shader;
00051 }
```

10.65.2.6 Load() [2/2]

```
Vesper::Ref< Vesper::Shader > Vesper::ShaderLibrary::Load (
    const std::string & name,
    const std::string & filepath)
```

Loads a shader from the specified filepath and adds it to the library with the given name.

```
00054 {
00055     VZ_PROFILE_FUNCTION();
00056     auto shader = Shader::Create(filepath);
00057     Add(Ref<Vesper::Shader>(shader));
00058     return shader;
00059 }
```

10.65.3 Member Data Documentation

10.65.3.1 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

10.66 Vesper::SpriteRendererComponent Struct Reference

Component that holds sprite rendering data.

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

Public Member Functions

- `SpriteRendererComponent ()=default`
- `SpriteRendererComponent (const SpriteRendererComponent &)=default`
- `SpriteRendererComponent (const glm::vec4 &color)`
Constructor that initializes the color.
- `operator glm::vec4 & ()`
- `operator const glm::vec4 & () const`
- `glm::vec4 & GetColor ()`
Returns the color of the sprite.

Public Attributes

- `glm::vec4 Color { 1.0f, 1.0f, 1.0f, 1.0f }`
Color of the sprite.
- `Ref< Texture2D > Texture = nullptr`
Texture of the sprite.
- `float TilingFactor = 1.0f`
Tiling factor for the texture.
- `bool TextureEnabled = false`
whether the texturing is enabled
- `bool Billboard = false`
whether the sprite should always face the camera

10.66.1 Detailed Description

Component that holds sprite rendering data.

10.66.2 Constructor & Destructor Documentation

10.66.2.1 `SpriteRendererComponent()` [1/3]

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

10.66.2.2 `SpriteRendererComponent()` [2/3]

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

10.66.2.3 SpriteRendererComponent() [3/3]

```
Vesper::SpriteRendererComponent::SpriteRendererComponent (
    const glm::vec4 & color) [inline]
```

Constructor that initializes the color.

```
00102         : Color(color) {
00103 }
```

10.66.3 Member Function Documentation

10.66.3.1 GetColor()

```
glm::vec4 & Vesper::SpriteRendererComponent::GetColor () [inline]
```

Returns the color of the sprite.

```
00109 { return Color; }
```

10.66.3.2 operator const glm::vec4 &()

```
Vesper::SpriteRendererComponent::operator const glm::vec4 & () const [inline]
00106 { return Color; }
```

10.66.3.3 operator glm::vec4 &()

```
Vesper::SpriteRendererComponent::operator glm::vec4 & () [inline]
00105 { return Color; }
```

10.66.4 Member Data Documentation

10.66.4.1 Billboard

```
bool Vesper::SpriteRendererComponent::Billboard = false
```

whether the sprite should always face the camera

WIP

10.66.4.2 Color

```
glm::vec4 Vesper::SpriteRendererComponent::Color { 1.0f, 1.0f, 1.0f, 1.0f }
```

Color of the sprite.

```
00092 { 1.0f, 1.0f, 1.0f, 1.0f };
```

10.66.4.3 Texture

```
Ref<Texture2D> Vesper::SpriteRendererComponent::Texture = nullptr
```

[Texture](#) of the sprite.

10.66.4.4 TextureEnabled

```
bool Vesper::SpriteRendererComponent::TextureEnabled = false
```

whether the texturing is enabled

10.66.4.5 TilingFactor

```
float Vesper::SpriteRendererComponent::TilingFactor = 1.0f
```

Tiling factor for the texture.

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

- Vesper/src/Vesper/Scene/[Components.h](#)

10.67 Vesper::Renderer2D::Statistics Struct Reference

2D Renderer Statistics

```
#include <Renderer2D.h>
```

Public Member Functions

- uint32_t [GetTotalVertexCount](#) ()
Calculates the total number of vertices drawn.
- uint32_t [GetTotalIndexCount](#) ()
Calculates the total number of indices drawn.

Public Attributes

- uint32_t [DrawCalls](#) = 0
The number of draw calls being made.
- uint32_t [QuadCount](#) = 0
The number of quads being drawn.

10.67.1 Detailed Description

2D Renderer Statistics

10.67.2 Member Function Documentation

10.67.2.1 GetTotalIndexCount()

```
uint32_t Vesper::Renderer2D::Statistics::GetTotalIndexCount () [inline]
```

Calculates the total number of indices drawn.

```
00213 { return QuadCount * 6; }
```

References [QuadCount](#).

10.67.2.2 GetTotalVertexCount()

```
uint32_t Vesper::Renderer2D::Statistics::GetTotalVertexCount () [inline]
```

Calculates the total number of vertices drawn.

```
00211 { return QuadCount * 4; }
```

References [QuadCount](#).

10.67.3 Member Data Documentation

10.67.3.1 DrawCalls

```
uint32_t Vesper::Renderer2D::Statistics::DrawCalls = 0
```

The number of draw calls being made.

Referenced by [Vesper::Renderer2D::Flush\(\)](#).

10.67.3.2 QuadCount

```
uint32_t Vesper::Renderer2D::Statistics::QuadCount = 0
```

The number of quads being drawn.

Referenced by [Vesper::Renderer2D::DrawQuad\(\)](#), [Vesper::Renderer2D::DrawQuadRotated\(\)](#), [Vesper::Renderer2D::DrawQuadRotatedWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuadWithTexture\(\)](#), [Vesper::Renderer2D::DrawQuads\(\)](#), [GetTotalIndexCount\(\)](#), and [GetTotalVertexCount\(\)](#).

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

- [Vesper/src/Vesper/Renderer/Renderer2D.h](#)

10.68 Vesper::SubTexture2D Class Reference

Represents a sub-region of a 2D texture, useful for sprite sheets.

```
#include <SubTexture2D.h>
```

Public Member Functions

- `SubTexture2D (const Ref< Texture2D > &texture, const glm::vec2 &min, const glm::vec2 &max)`
Constructs a SubTexture2D from the given texture and texture coordinates.
- `const Ref< Texture2D > GetTexture ()`
Returns the underlying texture of the sub-texture.
- `glm::vec2 * GetTexCoords ()`
Returns the texture coordinates of the sub-texture.

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})`
Creates a SubTexture2D from a grid of cells within the given texture.

Private Attributes

- `Ref< Texture2D > m_Texture`
The underlying texture of the sub-texture.
- `glm::vec2 m_TexCoords [4]`
The texture coordinates of the sub-texture.

10.68.1 Detailed Description

Represents a sub-region of a 2D texture, useful for sprite sheets.

10.68.2 Constructor & Destructor Documentation

10.68.2.1 SubTexture2D()

```
Vesper::SubTexture2D::SubTexture2D (
    const Ref< Texture2D > & texture,
    const glm::vec2 & min,
    const glm::vec2 & max)
```

Constructs a SubTexture2D from the given texture and texture coordinates.

Parameters

| | |
|----------------------|--|
| <code>texture</code> | The texture from which the sub-texture is derived. |
| <code>min</code> | The minimum texture coordinates (bottom-left). |
| <code>max</code> | The maximum texture coordinates (top-right). |

```
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\(\)](#).

10.68.3 Member Function Documentation

10.68.3.1 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]
```

Creates a [SubTexture2D](#) from a grid of cells within the given texture.

Parameters

| | |
|-------------------|--|
| <i>texture</i> | The texture from which the sub-texture is derived. |
| <i>coords</i> | The coordinates of the cell in the grid. |
| <i>cellSize</i> | The size of each cell in the grid. |
| <i>spriteSize</i> | The size of the sprite in cells (default is 1x1). |

```
00018     {
00019         glm::vec2 min = { (coords.x * cellSize.x) / texture->GetWidth(), (coords.y * cellSize.y) /
00020                         texture->GetHeight() };
00020         glm::vec2 max = { ((coords.x + spriteSize.x) * cellSize.x) / texture->GetWidth(), ((coords.y +
00021                         spriteSize.y) * cellSize.y) / texture->GetHeight() };
00022         return CreateRef<SubTexture2D>(texture, min, max);
00023     }
```

10.68.3.2 GetTexCoords()

```
glm::vec2 * Vesper::SubTexture2D::GetTexCoords () [inline]
```

Returns the texture coordinates of the sub-texture.

```
00025 { return m_TexCoords; }
```

10.68.3.3 GetTexture()

```
const Ref< Texture2D > Vesper::SubTexture2D::GetTexture () [inline]
```

Returns the underlying texture of the sub-texture.

```
00023 { return m_Texture; }
```

10.68.4 Member Data Documentation

10.68.4.1 m_TexCoords

```
glm::vec2 Vesper::SubTexture2D::m_TexCoords[4] [private]
```

The texture coordinates of the sub-texture.

10.68.4.2 m_Texture

```
Ref<Texture2D> Vesper::SubTexture2D::m_Texture [private]
```

The underlying texture of the sub-texture.

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

- Vesper/src/Vesper/Renderer/SubTexture2D.h
- Vesper/src/Vesper/Renderer/SubTexture2D.cpp

10.69 Vesper::SubTextureComponent Struct Reference

Component that holds sub-texture data for sprites.

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

Public Member Functions

- **SubTextureComponent ()=default**
- **SubTextureComponent (const Ref< Texture2D > &texture)**
Constructor that initializes the sub-texture from a full texture.
- **SubTextureComponent (const Ref< SubTexture2D > &subTexture)**
Constructor that initializes the sub-texture directly.
- **void SetTexture (const Ref< Texture2D > &texture)**
Copy constructor.
- **void SetTilingFactor (const glm::vec2 &tiling)**
Sets the tiling factor for the sub-texture.
- **void SetOffset (const glm::vec2 &offset)**
Sets the offset for the sub-texture.
- **operator Ref< SubTexture2D > & ()**
- **operator const Ref< SubTexture2D > & () const**
- **Ref< SubTexture2D > & GetSubTexture ()**

Public Attributes

- **Ref< SubTexture2D > SubTexture**
The sub-texture reference.
- **glm::vec2 TilingFactor = { 1.0f, 1.0f }**
Tiling factor for the sub-texture.
- **glm::vec2 Offset = { 0.0f, 0.0f }**
Offset for the sub-texture.

10.69.1 Detailed Description

Component that holds sub-texture data for sprites.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 SubTextureComponent() [1/3]

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

10.69.2.2 SubTextureComponent() [2/3]

```
Vesper::SubTextureComponent::SubTextureComponent (
    const Ref< Texture2D > & texture) [inline]
```

Constructor that initializes the sub-texture from a full texture.

```
00135         : SubTexture(SubTexture2D::CreateFromCoords(texture, { 0, 0 }, { texture->GetWidth(),
    texture->GetHeight() })) {
00136 }
```

10.69.2.3 SubTextureComponent() [3/3]

```
Vesper::SubTextureComponent::SubTextureComponent (
    const Ref< SubTexture2D > & subTexture) [inline]
```

Constructor that initializes the sub-texture directly.

```
00139         : SubTexture(subTexture) {
00140 }
```

10.69.3 Member Function Documentation

10.69.3.1 GetSubTexture()

```
Ref< SubTexture2D > & Vesper::SubTextureComponent::GetSubTexture () [inline]
00158 { return SubTexture; }
```

10.69.3.2 operator const Ref< SubTexture2D > &()

```
Vesper::SubTextureComponent::operator const Ref< SubTexture2D > & () const [inline]
00157 { return SubTexture; }
```

10.69.3.3 operator Ref< SubTexture2D > &()

```
Vesper::SubTextureComponent::operator Ref< SubTexture2D > & () [inline]
00156 { return SubTexture; }
```

10.69.3.4 SetOffset()

```
void Vesper::SubTextureComponent::SetOffset (
    const glm::vec2 & offset) [inline]
```

Sets the offset for the sub-texture.

```
00152             {
00153     Offset = offset;
00154 }
```

10.69.3.5 SetTexture()

```
void Vesper::SubTextureComponent::SetTexture (
    const Ref< Texture2D > & texture) [inline]
```

Copy constructor.

```
00142             SubTexture = SubTexture2D::CreateFromCoords { texture, { 0, 0 }, { texture->GetWidth(),
00143                                         texture->GetHeight() } };
00144 }
```

10.69.3.6 SetTilingFactor()

```
void Vesper::SubTextureComponent::SetTilingFactor (
    const glm::vec2 & tiling) [inline]
```

Sets the tiling factor for the sub-texture.

```
00147             TilingFactor = tiling;
00148 }
00149 }
```

10.69.4 Member Data Documentation

10.69.4.1 Offset

```
glm::vec2 Vesper::SubTextureComponent::Offset = { 0.0f, 0.0f }
```

Offset for the sub-texture.

```
00128 { 0.0f, 0.0f },
```

10.69.4.2 SubTexture

```
Ref<SubTexture2D> Vesper::SubTextureComponent::SubTexture
```

The sub-texture reference.

10.69.4.3 TilingFactor

```
glm::vec2 Vesper::SubTextureComponent::TilingFactor = { 1.0f, 1.0f }
```

Tiling factor for the sub-texture.

```
00126 { 1.0f, 1.0f },
```

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

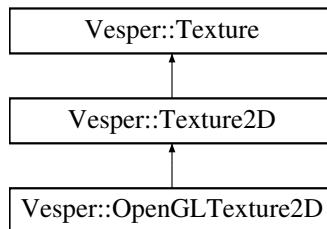
- Vesper/src/Vesper/Scene/Components.h

10.70 Vesper::Texture Class Reference

An abstraction for a texture.

```
#include <Texture.h>
```

Inheritance diagram for Vesper::Texture:



Public Member Functions

- virtual [~Texture \(\)=default](#)
- virtual uint32_t [GetWidth \(\) const =0](#)
Returns the width of the texture.
- virtual uint32_t [GetHeight \(\) const =0](#)
Returns the height of the texture.
- virtual uint32_t [GetRendererID \(\) const =0](#)
Returns the renderer ID of the texture.
- virtual void [Bind \(uint32_t slot=0\) const =0](#)
Binds the texture to the specified slot for use.
- virtual void [SetData \(void *data, uint32_t size\)=0](#)
Sets the data of the texture.
- virtual bool [operator== \(const Texture2D &other\) const =0](#)
- virtual std::string [GetName \(\) const =0](#)

10.70.1 Detailed Description

An abstraction for a texture.

10.70.2 Constructor & Destructor Documentation

10.70.2.1 ~Texture()

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

10.70.3 Member Function Documentation

10.70.3.1 Bind()

```
virtual void Vesper::Texture::Bind (
    uint32_t slot = 0) const [pure virtual]
```

Binds the texture to the specified slot for use.

Implemented in [Vesper::OpenGLTexture2D](#).

10.70.3.2 GetHeight()

```
virtual uint32_t Vesper::Texture::GetHeight () const [pure virtual]
```

Returns the height of the texture.

Implemented in [Vesper::OpenGLTexture2D](#).

10.70.3.3 GetName()

```
virtual std::string Vesper::Texture::GetName () const [pure virtual]
```

Implemented in [Vesper::OpenGLTexture2D](#).

10.70.3.4 GetRendererID()

```
virtual uint32_t Vesper::Texture::GetRendererID () const [pure virtual]
```

Returns the renderer ID of the texture.

Implemented in [Vesper::OpenGLTexture2D](#).

10.70.3.5 GetWidth()

```
virtual uint32_t Vesper::Texture::GetWidth () const [pure virtual]
```

Returns the width of the texture.

Implemented in [Vesper::OpenGLTexture2D](#).

10.70.3.6 operator==()

```
virtual bool Vesper::Texture::operator== (
    const Texture2D & other) const [pure virtual]
```

Implemented in [Vesper::OpenGLTexture2D](#).

10.70.3.7 SetData()

```
virtual void Vesper::Texture::SetData (
    void * data,
    uint32_t size) [pure virtual]
```

Sets the data of the texture.

Parameters

| | |
|-------------|--------------------------------|
| <i>data</i> | Pointer to the data to be set. |
| <i>size</i> | Size of the data in bytes. |

Implemented in [Vesper::OpenGLTexture2D](#).

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

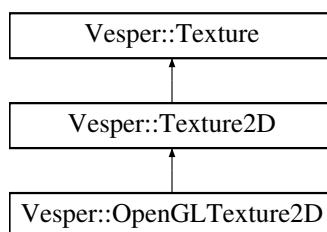
- Vesper/src/Vesper/Renderer/Texture.h

10.71 Vesper::Texture2D Class Reference

An abstraction for a 2D texture.

```
#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
Returns the width of the texture.
- virtual uint32_t [GetHeight](#) () const =0
Returns the height of the texture.
- virtual uint32_t [GetRendererID](#) () const =0
Returns the renderer ID of the texture.
- virtual void [Bind](#) (uint32_t slot=0) const =0
Binds the texture to the specified slot for use.
- virtual void [SetData](#) (void *data, uint32_t size)=0
Sets the data of the texture.
- virtual bool [operator==](#) (const [Texture2D](#) &other) const =0
- virtual std::string [GetName](#) () const =0

10.71.1 Detailed Description

An abstraction for a 2D texture.

10.71.2 Member Function Documentation

10.71.2.1 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
00026                                             not supported!"); return nullptr;
00027         case RendererAPI::API::OpenGL:        return CreateRef<OpenGLTexture2D>(path);
00028     }
00029     VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00030     return nullptr;
00031 }
```

References [Vesper::Renderer::GetAPI\(\)](#), [Vesper::RendererAPI::None](#), and [Vesper::RendererAPI::OpenGL](#).

10.71.2.2 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
00015                                             not supported!"); return nullptr;
00016         case RendererAPI::API::OpenGL:        return CreateRef<OpenGLTexture2D>(width, height);
00017     }
00018     VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
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](#)

10.72 Vesper::TextureAnimationComponent Struct Reference

Animates through a series of sub textures.

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

Public Member Functions

- `TextureAnimationComponent ()=default`
- `TextureAnimationComponent (const TextureAnimationComponent &)=default`
- `TextureAnimationComponent (const std::vector< Ref< SubTexture2D > > &subTextures, float frameTime)`
Constructor that initializes the sub-textures and frame time.
- `operator std::vector< Ref< SubTexture2D > > & ()`
- `operator const std::vector< Ref< SubTexture2D > > & () const`
- `std::vector< Ref< SubTexture2D > > & GetSubTextures ()`
- `uint32_t GetCurrentFrame () const`
Returns the index of the current frame.
- `void Update (float deltaTime)`
Updates the animation based on the elapsed time.

Public Attributes

- `std::vector< Ref< SubTexture2D > > SubTextures`
The list of sub-textures for the animation.
- `uint32_t CurrentFrame = 0`
The current frame index.
- `float FrameTime = 0.6f`
Time per frame in seconds.
- `float TimeAccumulator = 0.0f`
Accumulated time for frame switching.

10.72.1 Detailed Description

Animates through a series of sub textures.

(can be used with full textures)

10.72.2 Constructor & Destructor Documentation

10.72.2.1 `TextureAnimationComponent()` [1/3]

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

10.72.2.2 `TextureAnimationComponent()` [2/3]

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

10.72.2.3 TextureAnimationComponent() [3/3]

```
Vesper::TextureAnimationComponent::TextureAnimationComponent (
    const std::vector< Ref< SubTexture2D > > & subTextures,
    float frameTime) [inline]
```

Constructor that initializes the sub-textures and frame time.

Parameters

| | |
|--------------------|---|
| <i>subTextures</i> | Vector of sub-textures for the animation. |
| <i>frameTime</i> | Time per frame in seconds. |

```
00182     : SubTextures(subTextures), FrameTime(frameTime) {
00183 }
```

10.72.3 Member Function Documentation

10.72.3.1 GetCurrentFrame()

```
uint32_t Vesper::TextureAnimationComponent::GetCurrentFrame () const [inline]
```

Returns the index of the current frame.

```
00190 { return CurrentFrame; }
```

10.72.3.2 GetSubTextures()

```
std::vector< Ref< SubTexture2D > > & Vesper::TextureAnimationComponent::GetSubTextures () [inline]
00187 { return SubTextures; }
```

10.72.3.3 operator const std::vector< Ref< SubTexture2D > > &()

```
Vesper::TextureAnimationComponent::operator const std::vector< Ref< SubTexture2D > > & ()
const [inline]
00185 { return SubTextures; }
```

10.72.3.4 operator std::vector< Ref< SubTexture2D > > &()

```
Vesper::TextureAnimationComponent::operator std::vector< Ref< SubTexture2D > > & () [inline]
00184 { return SubTextures; }
```

10.72.3.5 Update()

```
void Vesper::TextureAnimationComponent::Update (
    float deltaTime) [inline]
```

Updates the animation based on the elapsed time.

Parameters

| | |
|------------------|---|
| <i>deltaTime</i> | The elapsed time since the last update. |
|------------------|---|

```
00195         {
00196     if (SubTextures.empty() || FrameTime <= 0.0f)
00197         return;
00198
00199     TimeAccumulator += deltaTime;
00200     while (TimeAccumulator >= FrameTime) {
00201         CurrentFrame = (CurrentFrame + 1) % static_cast<uint32_t>(SubTextures.size());
00202         TimeAccumulator -= FrameTime;
00203     }
00204 }
```

10.72.4 Member Data Documentation

10.72.4.1 CurrentFrame

```
uint32_t Vesper::TextureAnimationComponent::CurrentFrame = 0
```

The current frame index.

10.72.4.2 FrameTime

```
float Vesper::TextureAnimationComponent::FrameTime = 0.6f
```

Time per frame in seconds.

10.72.4.3 SubTextures

```
std::vector<Ref<SubTexture2D>> Vesper::TextureAnimationComponent::SubTextures
```

The list of sub-textures for the animation.

10.72.4.4 TimeAccumulator

```
float Vesper::TextureAnimationComponent::TimeAccumulator = 0.0f
```

Accumulated time for frame switching.

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

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

10.73 Vesper::TextureLibrary Class Reference

A library for managing and storing textures.

```
#include <Texture.h>
```

Public Member Functions

- void **Add** (const std::string &name, const Ref< Texture2D > &texture)
Adds a texture to the library with the specified name.
- void **Add** (const Ref< Texture2D > &texture)
Adds a texture to the library using its own name.
- Ref< Texture2D > **Load** (const std::string &filepath)
Loads a texture from the specified filepath and adds it to the library.
- Ref< Texture2D > **Load** (const std::string &name, const std::string &filepath)
Loads a texture from the specified filepath and adds it to the library with the given name.
- Ref< Texture2D > **Get** (const std::string &name) const
Retrieves a texture from the library by name.
- bool **Exists** (const std::string &name) const
Checks if a texture with the specified name exists in the library.

Private Attributes

- std::unordered_map< std::string, Ref< Texture2D > > **m_Textures**

10.73.1 Detailed Description

A library for managing and storing textures.

No current serialization implemented.

10.73.2 Member Function Documentation

10.73.2.1 Add() [1/2]

```
void Vesper::TextureLibrary::Add (
    const Ref< Texture2D > & texture)
```

Adds a texture to the library using its own name.

```
00040 {
00041     VZ_PROFILE_FUNCTION();
00042     auto& name = texture->GetName();
00043     Add(name, texture);
00044 }
```

10.73.2.2 Add() [2/2]

```
void Vesper::TextureLibrary::Add (
    const std::string & name,
    const Ref< Texture2D > & texture)
```

Adds a texture to the library with the specified name.

```
00033     {
00034         VZ_PROFILE_FUNCTION();
00035         VZ_CORE_ASSERT(!Exists(name), "Texture already exists!");
00036         m_Textures[name] = texture;
00037     }
```

10.73.2.3 Exists()

```
bool Vesper::TextureLibrary::Exists (
    const std::string & name) const
```

Checks if a texture with the specified name exists in the library.

```
00070     {
00071         VZ_PROFILE_FUNCTION();
00072         return m_Textures.find(name) != m_Textures.end();
00073     }
```

10.73.2.4 Get()

```
Ref< Texture2D > Vesper::TextureLibrary::Get (
    const std::string & name) const
```

Retrieves a texture from the library by name.

```
00063     {
00064         VZ_PROFILE_FUNCTION();
00065         VZ_CORE_ASSERT(Exists(name), "Texture not found!");
00066         return m_Textures.at(name);
00067     }
```

10.73.2.5 Load() [1/2]

```
Ref< Texture2D > Vesper::TextureLibrary::Load (
    const std::string & filepath)
```

Loads a texture from the specified filepath and adds it to the library.

```
00047     {
00048         VZ_PROFILE_FUNCTION();
00049         auto texture = Texture2D::Create(filepath);
00050         Add(texture);
00051         return texture;
00052     }
```

10.73.2.6 Load() [2/2]

```
Ref< Texture2D > Vesper::TextureLibrary::Load (
    const std::string & name,
    const std::string & filepath)
```

Loads a texture from the specified filepath and adds it to the library with the given name.

```
00055     {
00056         VZ_PROFILE_FUNCTION();
00057         auto texture = Texture2D::Create(filepath);
00058         Add(texture);
00059         return texture;
00060     }
```

10.73.3 Member Data Documentation

10.73.3.1 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](#)

10.74 Vesper::Timestep Class Reference

Represents a time step in seconds.

```
#include <Timestep.h>
```

Public Member Functions

- [Timestep \(float time=0.0f\)](#)
Constructs a [Timestep](#) with the given time in seconds.
- [operator float \(\) const](#)
- [float GetSeconds \(\) const](#)
Returns the time in seconds.
- [float GetMilliseconds \(\) const](#)
Returns the time in milliseconds.

Private Attributes

- [float m_Time](#)

10.74.1 Detailed Description

Represents a time step in seconds.

10.74.2 Constructor & Destructor Documentation

10.74.2.1 Timestep()

```
Vesper::Timestep::Timestep (
    float time = 0.0f) [inline]
```

Constructs a [Timestep](#) with the given time in seconds.

Parameters

| | |
|-------------|----------------------|
| <i>time</i> | The time in seconds. |
|-------------|----------------------|

```
00019         : m_Time(time)
00020     {
00021 }
```

References [m_Time](#).

10.74.3 Member Function Documentation

10.74.3.1 GetMilliseconds()

```
float Vesper::Timestep::GetMilliseconds () const [inline]
```

Returns the time in milliseconds.

```
00028 { return m_Time * 1000.0f; }
```

References [m_Time](#).

10.74.3.2 GetSeconds()

```
float Vesper::Timestep::GetSeconds () const [inline]
```

Returns the time in seconds.

```
00026 { return m_Time; }
```

References [m_Time](#).

10.74.3.3 operator float()

```
Vesper::Timestep::operator float () const [inline]
00023 { return m_Time; }
```

References [m_Time](#).

10.74.4 Member Data Documentation

10.74.4.1 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](#)

10.75 Vesper::TransformComponent Struct Reference

Component that holds the transform of an entity.

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

Public Member Functions

- [TransformComponent \(\)=default](#)
- [TransformComponent \(const TransformComponent &\)=default](#)
- [TransformComponent \(const glm::vec3 &translation\)](#)
Constructor that initializes the translation with a 3D vector.
- [glm::mat4 GetTransform \(\) const](#)
Calculates and returns the transformation matrix.

Public Attributes

- [glm::vec3 Translation = { 0.0f, 0.0f, 0.0f }](#)
Translation (position) vector.
- [glm::vec3 Rotation = { 0.0f, 0.0f, 0.0f }](#)
Rotation vector (in radians).
- [glm::vec3 Scale = { 1.0f, 1.0f, 1.0f }](#)
Scale vector.

10.75.1 Detailed Description

Component that holds the transform of an entity.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 TransformComponent() [1/3]

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

10.75.2.2 TransformComponent() [2/3]

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

10.75.2.3 TransformComponent() [3/3]

```
Vesper::TransformComponent::TransformComponent (
    const glm::vec3 & translation) [inline]
```

Constructor that initializes the translation with a 3D vector.

```
00074     : Translation(translation) {
00075 }
```

10.75.3 Member Function Documentation

10.75.3.1 GetTransform()

```
glm::mat4 Vesper::TransformComponent::GetTransform () const [inline]
```

Calculates and returns the transformation matrix.

```
00079     {
00080         glm::mat4 rotation = glm::toMat4(glm::quat(Rotation));
00081
00082         return glm::translate(glm::mat4(1.0f), Translation)
00083             * rotation
00084             * glm::scale(glm::mat4(1.0f), Scale);
00085     }
```

10.75.4 Member Data Documentation

10.75.4.1 Rotation

```
glm::vec3 Vesper::TransformComponent::Rotation = { 0.0f, 0.0f, 0.0f }
```

Rotation vector (in radians).

```
00066 { 0.0f, 0.0f, 0.0f };
```

10.75.4.2 Scale

```
glm::vec3 Vesper::TransformComponent::Scale = { 1.0f, 1.0f, 1.0f }
```

Scale vector.

```
00068 { 1.0f, 1.0f, 1.0f };
```

10.75.4.3 Translation

```
glm::vec3 Vesper::TransformComponent::Translation = { 0.0f, 0.0f, 0.0f }
```

Translation (position) vector.

```
00064 { 0.0f, 0.0f, 0.0f };
```

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

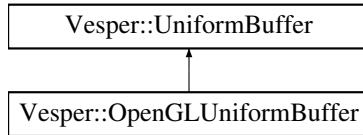
- Vesper/src/Vesper/Scene/Components.h

10.76 Vesper::UniformBuffer Class Reference

An abstraction for a uniform buffer object (UBO).

```
#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)

10.76.1 Detailed Description

An abstraction for a uniform buffer object (UBO).

10.76.2 Constructor & Destructor Documentation

10.76.2.1 ~UniformBuffer()

```
virtual Vesper::UniformBuffer::~UniformBuffer () [inline], [virtual]
00012 { }
```

10.76.3 Member Function Documentation

10.76.3.1 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
00014                                         supported!"); return nullptr;
00015         case RendererAPI::API::OpenGL: return CreateRef<OpenGLUniformBuffer>(size, binding);
00016     }
00017     VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00018     return nullptr;
00019 }
```

References [Vesper::Renderer::GetAPI\(\)](#), [Vesper::RendererAPI::None](#), and [Vesper::RendererAPI::OpenGL](#).

10.76.3.2 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](#)

10.77 Vesper::UUID Struct Reference

Universally Unique Identifier.

```
#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](#)
The string representation of the [UUID](#).

10.77.1 Detailed Description

Universally Unique Identifier.

A simple wrapper around a string representing a [UUID](#).

10.77.2 Constructor & Destructor Documentation

10.77.2.1 UUID() [1/2]

```
Vesper::UUID::UUID () [inline]
00021 { ID = Random::UUID(); }
```

10.77.2.2 UUID() [2/2]

```
Vesper::UUID::UUID (
    const std::string & id) [inline]
00023         : ID{ id } {
00024 }
```

10.77.3 Member Function Documentation

10.77.3.1 operator const std::string &()

```
Vesper::UUID::operator const std::string & () const [inline]
00026 { return ID; }
```

10.77.3.2 operator std::string &()

```
Vesper::UUID::operator std::string & () [inline]
00025 { return ID; }
```

10.77.4 Member Data Documentation

10.77.4.1 ID

std::string Vesper::UUID::ID

The string representation of the [UUID](#).

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

- Vesper/src/Vesper/Scene/[Components.h](#)

10.78 Vesper::UUIDComponent Struct Reference

Component that holds a [UUID](#).

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

Public Member Functions

- [UUIDComponent \(\)](#)
- [UUIDComponent \(const UUIDComponent &\) =default](#)
- [UUIDComponent \(const std::string &id\)](#)

Public Attributes

- [UUID ID](#)

The [UUID](#) of the owning entity.

10.78.1 Detailed Description

Component that holds a [UUID](#).

10.78.2 Constructor & Destructor Documentation

10.78.2.1 [UUIDComponent\(\)](#) [1/3]

```
Vesper::UUIDComponent::UUIDComponent () [inline]
00035     : ID() {
00036 }
```

10.78.2.2 [UUIDComponent\(\)](#) [2/3]

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

10.78.2.3 [UUIDComponent\(\)](#) [3/3]

```
Vesper::UUIDComponent::UUIDComponent (
    const std::string & id) [inline]
00039     : ID{ id } {
00040 }
```

10.78.3 Member Data Documentation

10.78.3.1 [ID](#)

[UUID](#) [Vesper::UUIDComponent::ID](#)

The [UUID](#) of the owning entity.

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

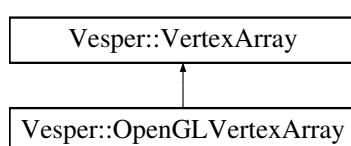
- [Vesper/src/Vesper/Scene/Components.h](#)

10.79 Vesper::VertexArray Class Reference

An abstraction for a vertex array object (VAO).

```
#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`
Adds a vertex buffer to the vertex array.
- virtual void `SetIndexBuffer (const Ref< IndexBuffer > &indexBuffer)=0`
Sets the index buffer for the vertex array.
- virtual const std::vector< `Ref< VertexBuffer >` > & `GetVertexBuffers ()=0`
- virtual const `Ref< IndexBuffer >` & `GetIndexBuffer () const =0`

Static Public Member Functions

- static `Ref< VertexArray > Create ()`

10.79.1 Detailed Description

An abstraction for a vertex array object (VAO).

10.79.2 Constructor & Destructor Documentation

10.79.2.1 `~VertexArray()`

```
virtual Vesper::VertexArray::~VertexArray () [inline], [virtual]
00013 {}
```

10.79.3 Member Function Documentation

10.79.3.1 `AddVertexBuffer()`

```
virtual void Vesper::VertexArray::AddVertexBuffer (
    const Ref< VertexBuffer > & vertexBuffer) [pure virtual]
```

Adds a vertex buffer to the vertex array.

Implemented in [Vesper::OpenGLVertexArray](#).

10.79.3.2 `Bind()`

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

Implemented in [Vesper::OpenGLVertexArray](#).

10.79.3.3 Create()

```

Ref< VertexArray > Vesper::VertexArray::Create ()  [static]
0008     {
0009         switch (Renderer::GetAPI())
0010         {
0011             case RendererAPI::API::None:           VZ_CORE_ASSERT(false, "RendererAPI::None is currently not
0012                                         supported!"); return nullptr;
0013             case RendererAPI::API::OpenGL:        return CreateRef<OpenGLVertexArray>();
0014             }
0015             VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
0016         return nullptr;
0016     }

```

References [Vesper::Renderer::GetAPI\(\)](#), [Vesper::RendererAPI::None](#), and [Vesper::RendererAPI::OpenGL](#).

10.79.3.4 GetIndexBuffer()

```
virtual const Ref< IndexBuffer > & Vesper::VertexArray::GetIndexBuffer () const  [pure virtual]
```

Implemented in [Vesper::OpenGLVertexArray](#).

10.79.3.5 GetVertexBuffers()

```
virtual const std::vector< Ref< VertexBuffer > > & Vesper::VertexArray::GetVertexBuffers ()  [pure virtual]
```

Implemented in [Vesper::OpenGLVertexArray](#).

10.79.3.6 SetIndexBuffer()

```
virtual void Vesper::VertexArray::SetIndexBuffer (
    const Ref< IndexBuffer > & indexBuffer)  [pure virtual]
```

Sets the index buffer for the vertex array.

Implemented in [Vesper::OpenGLVertexArray](#).

10.79.3.7 Unbind()

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

Implemented in [Vesper::OpenGLVertexArray](#).

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

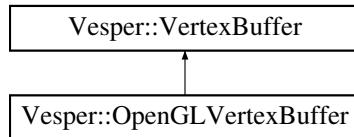
- [Vesper/src/Vesper/Renderer/VertexArray.h](#)
- [Vesper/src/Vesper/Renderer/VertexArray.cpp](#)

10.80 Vesper::VertexBuffer Class Reference

Abstract base class for a vertex buffer.

```
#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)`

10.80.1 Detailed Description

Abstract base class for a vertex buffer.

10.80.2 Constructor & Destructor Documentation

10.80.2.1 `~VertexBuffer()`

```
virtual Vesper::VertexBuffer::~VertexBuffer () [inline], [virtual]
00134 {}
```

10.80.3 Member Function Documentation

10.80.3.1 `Bind()`

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

Implemented in [Vesper::OpenGLVertexBuffer](#).

10.80.3.2 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
00027             not supported!"); return nullptr;
00028             case RendererAPI::API::OpenGL:        return CreateRef<OpenGLVertexBuffer>(vertices, size);
00029         }
00030         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00031     }
00031 }
```

References [Vesper::Renderer::GetAPI\(\)](#), [Vesper::RendererAPI::None](#), and [Vesper::RendererAPI::OpenGL](#).

10.80.3.3 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
00016             supported!"); return nullptr;
00016             case RendererAPI::API::OpenGL:        return CreateRef<OpenGLVertexBuffer>(size);
00017         }
00018         VZ_CORE_ASSERT(false, "Unknown RendererAPI!");
00019     }
00020 }
```

References [Vesper::Renderer::GetAPI\(\)](#), [Vesper::RendererAPI::None](#), and [Vesper::RendererAPI::OpenGL](#).

10.80.3.4 GetLayout()

```
virtual const BufferLayout & Vesper::VertexBuffer::GetLayout () const  [pure virtual]
```

Implemented in [Vesper::OpenGLVertexBuffer](#).

10.80.3.5 SetData()

```
virtual void Vesper::VertexBuffer::SetData (
    const void * data,
    uint32_t size)  [pure virtual]
```

Implemented in [Vesper::OpenGLVertexBuffer](#).

10.80.3.6 SetLayout()

```
virtual void Vesper::VertexBuffer::setLayout (
    const BufferLayout & layout)  [pure virtual]
```

Implemented in [Vesper::OpenGLVertexBuffer](#).

10.80.3.7 Unbind()

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

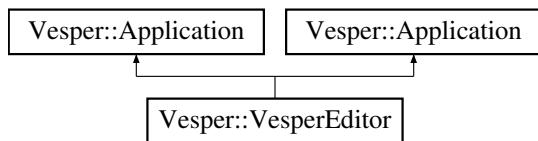
Implemented in [Vesper::OpenGLVertexBuffer](#).

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

- [Vesper/src/Vesper/Renderer/Buffer.h](#)
- [Vesper/src/Vesper/Renderer/Buffer.cpp](#)

10.81 Vesper::VesperEditor Class Reference

Inheritance diagram for Vesper::VesperEditor:



Public Member Functions

- [VesperEditor \(\)](#)
- [~VesperEditor \(\)](#)
- [VesperEditor \(\)](#)
- [~VesperEditor \(\)](#)

Public Member Functions inherited from [Vesper::Application](#)

- [Application \(const std::string &name=""\)](#)
Constructs the `Application` with the given name.
- [virtual ~Application \(\)](#)
- [void Run \(\)](#)
Starts the main application loop.
- [void OnEvent \(Event &e\)](#)
Handles incoming events and dispatches them to the appropriate handlers.
- [void PushLayer \(Layer *layer\)](#)
Adds a layer to the application layer stack.
- [void PushOverlay \(Layer *overlay\)](#)
Adds an overlay layer to the application layer stack.
- [void Close \(\)](#)
Closes the application.
- [ImGuiLayer * GetImGuiLayer \(\)](#)
Retrieves the `ImGui` layer.
- [Window & GetWindow \(\)](#)
Retrieves the application window.

Additional Inherited Members

Static Public Member Functions inherited from [Vesper::Application](#)

- static [Application & Get \(\)](#)
Retrieves the singleton instance of the Application.

10.81.1 Constructor & Destructor Documentation

10.81.1.1 [VesperEditor\(\)](#) [1/2]

```
Vesper::VesperEditor::VesperEditor () [inline]
00012         : Application("Vesper Editor")
00013         {
00014             PushLayer(new EditorLayer());
00015         }
```

References [Vesper::EditorLayer::EditorLayer\(\)](#), and [Vesper::Application::PushLayer\(\)](#).

Referenced by [Vesper::CreateApplication\(\)](#).

10.81.1.2 [~VesperEditor\(\)](#) [1/2]

```
Vesper::VesperEditor::~VesperEditor () [inline]
00018         {
00019
00020     }
```

10.81.1.3 [VesperEditor\(\)](#) [2/2]

```
Vesper::VesperEditor::VesperEditor () [inline]
00012         : Application("Vesper Editor")
00013         {
00014             PushLayer(new EditorLayer());
00015         }
00016     }
```

10.81.1.4 [~VesperEditor\(\)](#) [2/2]

```
Vesper::VesperEditor::~VesperEditor () [inline]
00018         {
00019
00020     }
```

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

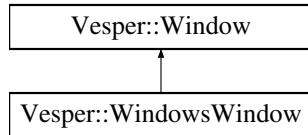
- [Vesper-Editor/src/VesperEditorApp.cpp](#)

10.82 Vesper::Window Class Reference

Abstract interface representing an application window.

```
#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`
Called every frame to update the window.
- virtual `uint32_t GetWidth() const =0`
Retrieves the width of the window.
- virtual `uint32_t GetHeight() const =0`
Retrieves the height of the window.
- virtual void `SetEventCallback(const EventCallbackFn &callback)=0`
Sets the callback function for window events.
- virtual void `SetVSync(bool enabled)=0`
Sets whether vertical synchronization (VSync) is enabled.
- virtual `bool IsVSync() const =0`
Checks if vertical synchronization (VSync) is enabled.
- virtual `* GetNativeWindow() const =0`
Retrieves the native window handle.

Static Public Member Functions

- static `Scope< Window > Create(const WindowProps &props=WindowProps())`
Creates a window instance with the specified properties.

10.82.1 Detailed Description

Abstract interface representing an application window.

Todo Add `Window` mode functionality

10.82.2 Member Typedef Documentation

10.82.2.1 EventCallbackFn

```
using Vesper::Window::EventCallbackFn = std::function<void(Event&)>
```

10.82.3 Constructor & Destructor Documentation

10.82.3.1 ~Window()

```
virtual Vesper::Window::~Window () [inline], [virtual]  
00047 {}
```

10.82.4 Member Function Documentation

10.82.4.1 Create()

```
Scope< Window > Vesper::Window::Create (  
    const WindowProps & props = WindowProps()) [static]
```

Creates a window instance with the specified properties.

Parameters

| | |
|--------------|---|
| <i>props</i> | The properties to initialize the window with. |
|--------------|---|

Returns

A scoped pointer to the created window instance.

```
00021 {  
00022     return CreateScope<WindowsWindow>(props);  
00023 }
```

10.82.4.2 GetHeight()

```
virtual uint32_t Vesper::Window::GetHeight () const [pure virtual]
```

Retrieves the height of the window.

Implemented in [Vesper::WindowsWindow](#).

Referenced by [Vesper::ImGuiLayer::End\(\)](#).

10.82.4.3 GetNativeWindow()

```
virtual void * Vesper::Window::GetNativeWindow () const [pure virtual]
```

Retrieves the native window handle.

Implemented in [Vesper::WindowsWindow](#).

Referenced by [Vesper::Input::GetPosition\(\)](#), [Vesper::Input::IsKeyPressed\(\)](#), and [Vesper::Input::IsMouseButtonPressed\(\)](#).

10.82.4.4 GetWidth()

```
virtual uint32_t Vesper::Window::GetWidth () const [pure virtual]
```

Retrieves the width of the window.

Implemented in [Vesper::WindowsWindow](#).

Referenced by [Vesper::ImGuiLayer::End\(\)](#).

10.82.4.5 IsVSync()

```
virtual bool Vesper::Window::IsVSync () const [pure virtual]
```

Checks if vertical synchronization (VSync) is enabled.

Implemented in [Vesper::WindowsWindow](#).

10.82.4.6 OnUpdate()

```
virtual void Vesper::Window::OnUpdate () [pure virtual]
```

Called every frame to update the window.

Implemented in [Vesper::WindowsWindow](#).

10.82.4.7 SetEventCallback()

```
virtual void Vesper::Window::SetEventCallback (
    const EventCallbackFn & callback) [pure virtual]
```

Sets the callback function for window events.

Implemented in [Vesper::WindowsWindow](#).

10.82.4.8 SetVSync()

```
virtual void Vesper::Window::SetVSync (
    bool enabled) [pure virtual]
```

Sets whether vertical synchronization (VSync) is enabled.

Implemented in [Vesper::WindowsWindow](#).

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

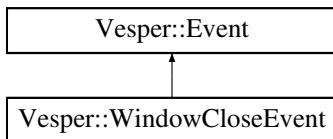
- [Vesper/src/Vesper/App/Window.h](#)
- [Vesper/src/Platform/Windows/WindowsWindow.cpp](#)

10.83 Vesper::WindowCloseEvent Class Reference

[Event](#) for registering window close.

```
#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](#)

Get the type of the event.
- virtual const char * [GetName \(\) const =0](#)

Get the name of the event.
- virtual int [GetCategoryFlags \(\) const =0](#)

Get the category flags of the event.
- virtual std::string [ToString \(\) const](#)

Convert the event to a string representation.
- bool [IsInCategory \(EventCategory category\)](#)

Check if the event is in a specific category.

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled](#) = false
Indicates whether the event has been handled.

10.83.1 Detailed Description

[Event](#) for registering window close.

10.83.2 Constructor & Destructor Documentation

10.83.2.1 [WindowCloseEvent\(\)](#)

```
Vesper::WindowCloseEvent::WindowCloseEvent () [inline]  
00051 {}
```

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

- Vesper/src/Vesper/Events/[ApplicationEvent.h](#)

10.84 [Vesper::WindowProps](#) Struct Reference

Holds the data for window configuration.

```
#include <Window.h>
```

Public Member Functions

- [WindowProps](#) (const std::string &title="Vesper Engine", uint32_t width=1600, uint32_t height=900)
Constructor to initialize [WindowProps](#) with given or default values.

Public Attributes

- std::string [Title](#)
- uint32_t [Width](#)
- uint32_t [Height](#)

10.84.1 Detailed Description

Holds the data for window configuration.

10.84.2 Constructor & Destructor Documentation

10.84.2.1 WindowProps()

```
Vesper::WindowProps::WindowProps (
    const std::string & title = "Vesper Engine",
    uint32_t width = 1600,
    uint32_t height = 900) [inline]
```

Constructor to initialize [WindowProps](#) with given or default values.

Parameters

| | |
|---------------|---------------------------|
| <i>title</i> | The title of the window. |
| <i>width</i> | The width of the window. |
| <i>height</i> | The height of the window. |

```
00036 : Title(title), Width(width), Height(height) {}
```

References [Height](#), and [Width](#).

10.84.3 Member Data Documentation

10.84.3.1 Height

```
uint32_t Vesper::WindowProps::Height
```

Referenced by [Vesper::WindowsWindow::Init\(\)](#), and [WindowProps\(\)](#).

10.84.3.2 Title

```
std::string Vesper::WindowProps::Title
```

10.84.3.3 Width

```
uint32_t Vesper::WindowProps::Width
```

Referenced by [Vesper::WindowsWindow::Init\(\)](#), and [WindowProps\(\)](#).

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

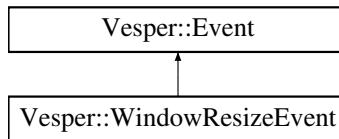
- Vesper/src/Vesper/App/[Window.h](#)

10.85 Vesper::WindowResizeEvent Class Reference

[Event](#) for registering window resize.

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

Inheritance diagram for Vesper::WindowResizeEvent:



Public Member Functions

- [WindowResizeEvent](#) (unsigned int width, unsigned int height)
Construct a [WindowResizeEvent](#) with the specified width and height.
- unsigned int [GetWidth](#) () const
Get the new width of the window.
- unsigned int [GetHeight](#) () const
Get the new height of the window.
- std::string [ToString](#) () const override
Convert the event to a string representation.

Public Member Functions inherited from [Vesper::Event](#)

- virtual [~Event](#) ()=default
- virtual [EventType GetEventType](#) () const =0
Get the type of the event.
- virtual const char * [GetName](#) () const =0
Get the name of the event.
- virtual int [GetCategoryFlags](#) () const =0
Get the category flags of the event.
- bool [IsInCategory](#) ([EventCategory](#) category)
Check if the event is in a specific category.

Private Attributes

- unsigned int [m_Width](#)
- unsigned int [m_Height](#)

Additional Inherited Members

Public Attributes inherited from [Vesper::Event](#)

- bool [Handled](#) = false
Indicates whether the event has been handled.

10.85.1 Detailed Description

[Event](#) for registering window resize.

10.85.2 Constructor & Destructor Documentation

10.85.2.1 WindowResizeEvent()

```
Vesper::WindowResizeEvent::WindowResizeEvent (
    unsigned int width,
    unsigned int height) [inline]
```

Construct a [WindowResizeEvent](#) with the specified width and height.

Parameters

| | |
|---------------|-------------------------------|
| <i>width</i> | The new width of the window. |
| <i>height</i> | The new height of the window. |

```
00024 : m_Width(width), m_Height(height) {}
```

References [m_Height](#), and [m_Width](#).

10.85.3 Member Function Documentation

10.85.3.1 GetHeight()

```
unsigned int Vesper::WindowResizeEvent::GetHeight () const [inline]
```

Get the new height of the window.

```
00029 { return m_Height; }
```

References [m_Height](#).

Referenced by [Vesper::Application::OnWindowResize\(\)](#), and [Vesper::OrthographicCameraController::OnWindowResized\(\)](#).

10.85.3.2 GetWidth()

```
unsigned int Vesper::WindowResizeEvent::GetWidth () const [inline]
```

Get the new width of the window.

```
00027 { return m_Width; }
```

References [m_Width](#).

Referenced by [Vesper::Application::OnWindowResize\(\)](#), and [Vesper::OrthographicCameraController::OnWindowResized\(\)](#).

10.85.3.3 ToString()

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

Convert the event to a string representation.

Reimplemented from [Vesper::Event](#).

```
00033     {
00034         std::stringstream ss;
00035         ss << "WindowResizeEvent: " << m_Width << ", " << m_Height;
00036         return ss.str();
00037     }
```

References [m_Height](#), and [m_Width](#).

10.85.4 Member Data Documentation

10.85.4.1 m_Height

```
unsigned int Vesper::WindowResizeEvent::m_Height [private]
```

Referenced by [GetHeight\(\)](#), [ToString\(\)](#), and [WindowResizeEvent\(\)](#).

10.85.4.2 m_Width

```
unsigned int Vesper::WindowResizeEvent::m_Width [private]
```

Referenced by [GetWidth\(\)](#), [ToString\(\)](#), and [WindowResizeEvent\(\)](#).

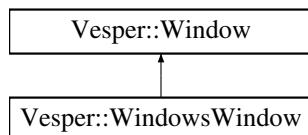
The documentation for this class was generated from the following file:

- [Vesper/src/Vesper/Events/ApplicationEvent.h](#)

10.86 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`

Called every frame to update the window.
- unsigned int `GetWidth () const override`

Retrieves the width of the window.
- unsigned int `GetHeight () const override`

Retrieves the height of the window.
- void `SetEventCallback (const EventCallbackFn &callback) override`

Sets the callback function for window events.
- void `SetVSync (bool enabled) override`

Sets whether vertical synchronization (VSync) is enabled.
- bool `IsVSync () const override`

Checks if vertical synchronization (VSync) is enabled.
- virtual void * `GetNativeWindow () const override`

Retrieves the native window handle.

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())`

Creates a window instance with the specified properties.

10.86.1 Class Documentation

10.86.1.1 struct Vesper::WindowsWindow::WindowData

Class Members

| | | |
|---------------------------------|---------------|--|
| EventCallbackFn | EventCallback | |
| unsigned int | Height | |
| string | Title | |
| bool | VSync | |
| unsigned int | Width | |

10.86.2 Constructor & Destructor Documentation

10.86.2.1 WindowsWindow()

```
Vesper::WindowsWindow::WindowsWindow (
    const WindowProps & props)
00026     {
00027         VZ\_PROFILE\_FUNCTION\(\);
00028         Init(props);
00029     }
```

References [Init\(\)](#).

10.86.2.2 ~WindowsWindow()

```
Vesper::WindowsWindow::~WindowsWindow () [virtual]
00032     {
00033         Shutdown();
00034     }
```

References [Shutdown\(\)](#).

10.86.3 Member Function Documentation

10.86.3.1 GetHeight()

```
unsigned int Vesper::WindowsWindow::GetHeight () const [inline], [override], [virtual]
```

Retrieves the height of the window.

Implements [Vesper::Window](#).

```
00019 { return m\_Data.Height; }
```

10.86.3.2 GetNativeWindow()

```
virtual void * Vesper::WindowsWindow::GetNativeWindow () const [inline], [override], [virtual]
```

Retrieves the native window handle.

Implements [Vesper::Window](#).

```
00025 { return m_Window; }
```

10.86.3.3 GetWidth()

```
unsigned int Vesper::WindowsWindow::GetWidth () const [inline], [override], [virtual]
```

Retrieves the width of the window.

Implements [Vesper::Window](#).

```
00018 { return m_Data.Width; }
```

10.86.3.4 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(),
00055     nullptr, nullptr);
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
00083     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\(\)](#).

10.86.3.5 IsVSync()

```
bool Vesper::WindowsWindow::IsVSync () const [override], [virtual]
```

Checks if vertical synchronization (VSync) is enabled.

Implements [Vesper::Window](#).

```

00175 {
00176     return m_Data.VSync;
00177 }
```

10.86.3.6 OnUpdate()

```
void Vesper::WindowsWindow::OnUpdate () [override], [virtual]
```

Called every frame to update the window.

Implements [Vesper::Window](#).

```
00157     {
00158         VZ_PROFILE_FUNCTION();
00159         glfwPollEvents();
00160         m_Context->SwapBuffers();
00161     }
```

References [m_Context](#), and [Vesper::GraphicsContext::SwapBuffers\(\)](#).

10.86.3.7 SetEventCallback()

```
void Vesper::WindowsWindow::SetEventCallback (
    const EventCallbackFn & callback) [inline], [override], [virtual]
```

Sets the callback function for window events.

Implements [Vesper::Window](#).

```
00022 { m_Data.EventCallback = callback; }
```

10.86.3.8 SetVSync()

```
void Vesper::WindowsWindow::SetVSync (
    bool enabled) [override], [virtual]
```

Sets whether vertical synchronization (VSync) is enabled.

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\(\)](#).

10.86.3.9 Shutdown()

```
void Vesper::WindowsWindow::Shutdown () [private], [virtual]
00151     {
00152         VZ_PROFILE_FUNCTION();
00153         glfwDestroyWindow(m_Window);
00154     }
```

Referenced by [~WindowsWindow\(\)](#).

10.86.4 Member Data Documentation

10.86.4.1 m_Context

`GraphicsContext* Vesper::WindowsWindow::m_Context [private]`

Referenced by [Init\(\)](#), and [OnUpdate\(\)](#).

10.86.4.2 m_Data

`WindowData Vesper::WindowsWindow::m_Data [private]`

10.86.4.3 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 11

File Documentation

11.1 docs/README.md File Reference

11.2 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](#)

11.2.1 Variable Documentation

11.2.1.1 [s_MapHeight](#)

```
const uint32_t s_MapHeight = 10 [static]
```

Referenced by [Vesper::EditorLayer::OnUpdate\(\)](#).

11.2.1.2 s_MapTiles

```
const char * s_MapTiles [static]
```

Initial value:

```
=  
"GGGGGGGGGGGGGGGGGGGGGG"  
"GGGCCCCCCCCCCCCCCCGGG"  
"GGGCGGGGGGGGGGGCGGG"  
"GGGCGRGGGGGRGGCGGG"  
"GGGCGGGGGGGGGGGCGGG"  
"GGGCGGGGGGGGGGGGGCGGG"  
"GGGCGGPGGGGGPGGCGGG"  
"GGGCGGGGGGGGGGGGCGGG"  
"GGGCCCCCCCCCCCCCCCGGG"  
"GGGGGGGGGGGGGGGGGGGG"
```

Referenced by [Vesper::EditorLayer::OnUpdate\(\)](#).

11.2.1.3 s_MapWidth

```
const uint32_t s_MapWidth = 20 [static]
```

Referenced by [Vesper::EditorLayer::OnUpdate\(\)](#).

11.3 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.

11.4 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"
0010
0011 namespace Vesper {
0012
0013     class EditorLayer : public Layer
0014     {
0015     public:
0016         EditorLayer();
0017         virtual ~EditorLayer() = default;
0018
0019         virtual void OnAttach() override;
0020         virtual void OnDetach() override;
0021         virtual void OnUpdate(Timestep ts) override;
0022         virtual void OnImGuiRender() override;
0023         virtual void OnEvent(Event& e) override;
0024
0025     private:
0026         bool OnKeyPressed(KeyPressedEvent& e);
0027
0028         void NewScene();
0029         void OpenScene();
0030         void SaveSceneAs();
0031         void ResetScene();
0032
0033     private:
0034         SceneHierarchyPanel m_SceneHierarchyPanel;
0035
0036         Ref<Scene> m_ActiveScene;
0037         Ref<Scene> m_EditorScene;
0038
0039         enum class SceneState
0040         {
0041             Edit = 0, Play = 1, Simulate = 2
0042         };
0043         SceneState m_SceneState = SceneState::Edit;
0044
0045         bool m_ViewFocused = false, m_ViewHovered = false;
0046         glm::vec2 m_ViewSize = {0,0};
0047         glm::vec2 m_ViewBounds[2] = { {0,0}, {0,0} };
0048         bool m_PrimaryCamera = true;
0049         Entity m_CameraEntity;
0050         //Entity m_SecondaryCameraEntity;
0051         int m_GizmoType = -1;
0052         float m_TranslationSnap = 0.5f, m_RotationSnap = 45.0f, m_ScaleSnap = 0.5f;
0053
0054         OrthographicCameraController m_CameraController;
0055
0056         float lastFrameTime = 0.0f;
0057         Entity m_FireEntity, m_SmokeEntity;
0058
0059         // Temp
0060         Ref<VertexArray> m_SquareVA;
0061         Ref<Shader> m_FlatColorShader;
0062         Ref<Texture2D> m_CheckerboardTexture,
0063             m_SpriteSheetFire,
0064             m_SpriteSheetSmoke,
0065             m_SpriteSheetTown,
0066             m_SpriteSheetCrystals,
0067             m_SpriteSheetRocks,
0068             m_SpriteSheetCursedLands;
0069
0070         Ref<SubTexture2D> m_SubTextureFire,
0071         Ref<SubTexture2D> m_SubTextureSmoke,
0072         Ref<SubTexture2D> m_SubTextureTown,
0073         //Ref<SubTexture2D> m_SubTextureCrystal;
0074         //Ref<SubTexture2D> m_SubTextureRock;
0075         //Ref<SubTexture2D> m_SubTexturePlant;
0076
0077         Ref<Framebuffer> m_Framebuffer;
0078
0079         EditorCamera m_EditorCamera;
0080
0081         float m_textureScale = 1.0f;
0082         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     bool scene1 = false, scene2 = false, scene3 = false, scene4 = false;
00090     bool useEntityScene = true;
00091
00092     glm::vec4 m_SquareColor = { 0.2f, 0.3f, 0.8f, 1.0f };
00093     glm::vec4 m_TextureTintColor1 = { 1.0f, 1.0f, 1.0f, 1.0f };
00094     glm::vec4 m_TextureTintColor2 = { 1.0f, 1.0f, 1.0f, 1.0f };
00095     glm::vec4 m_BackgroundColor = { 0.1f, 0.1f, 0.1f, 1.0f };
00096     glm::vec4 m_ClearColor = { 0.1f, 0.3f, 0.3f, 1.0f };
00097     glm::vec4 m_SpecialQuadColor = { 0.9f, 0.2f, 0.8f, 1.0f };
00098     bool m_UseSpecialQuadColor = false;
00099
00100     std::unordered_map<char, Ref<SubTexture2D>> s_TextureMap;
00101
00102 };
00103
00104
00105
00106 }
```

11.5 Vesper-Editor/src/Panels/SceneHierarchyPanel.cpp File Reference

```
#include <Vesper/Utils/PlatformUtils.h>
#include "SceneHierarchyPanel.h"
#include "Vesper/Scene/Components.h"
#include <ImGuizmo/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)

11.6 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.

11.7 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 }
```

11.8 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 \(\)](#)

11.9 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.

11.10 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](#)

11.10.1 Macro Definition Documentation

11.10.1.1 GLFW_EXPOSE_NATIVE_WIN32

```
#define GLFW_EXPOSE_NATIVE_WIN32
```

11.11 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

11.12 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.

11.12.1 Class Documentation

11.12.1.1 struct Vesper::WindowsWindow::WindowData

Class Members

| | | |
|------------------------------|----------------------------|--|
| <code>EventCallbackFn</code> | <code>EventCallback</code> | |
| <code>unsigned int</code> | <code>Height</code> | |
| <code>string</code> | <code>Title</code> | |
| <code>bool</code> | <code>VSync</code> | |
| <code>unsigned int</code> | <code>Width</code> | |

11.13 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 =
00023             callback; }
00024         void SetVSync(bool enabled) override;
00025         bool IsVSync() const override;
00026         inline virtual void* GetNativeWindow() const override { return m_Window; }
00027
00028     private:
00029         GLFWwindow* m_Window;
00030         GraphicsContext* m_Context;
00031         struct WindowData
00032         {
00033             std::string Title;
00034             unsigned int Width, Height;
00035             bool VSync;
00036
00037             EventCallbackFn EventCallback;
00038         };
00039         WindowData m_Data;
00040     };
00041
00042
00043 }
```

11.14 Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLBuffer.h"
#include <glad/glad.h>
```

Namespaces

- namespace **Vesper**
TEMPORARY.

11.15 Vesper/src/RenderAPI/OpenGL/OpenGLBuffer.h File Reference

```
#include "Vesper/Renderer/Buffer.h"
```

Classes

- class **Vesper::OpenGLVertexBuffer**
- class **Vesper::OpenGLIndexBuffer**

Namespaces

- namespace **Vesper**
TEMPORARY.

11.16 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 }
```

11.17 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.

11.18 Vesper/src/RenderAPI/OpenGL/OpenGLContext.h File Reference

```
#include "Vesper/Renderer/GraphicsContext.h"
```

Classes

- class [Vesper::OpenGLContext](#)

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.19 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 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 }
```

11.20 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!

11.21 Vesper/src/RenderAPI/OpenGL/OpenGLFramebuffer.h File Reference

```
#include "Vesper/Renderer/Framebuffer.h"
```

Classes

- class [Vesper::OpenGLFramebuffer](#)

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.22 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             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     private:
00023     uint32_t m_RendererID;
00024     uint32_t m_ColorAttachment, m_DepthAttachment;
00025     FramebufferSpecification m_Specification;
00026
00027 };
00028
00029 }
```

11.23 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 "ImgUI.h"
```

Namespaces

- namespace **Vesper**

TEMPORARY.

11.24 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.

11.25 OpenGLGuiLayer.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 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 }
```

11.26 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.

11.27 Vesper/src/RenderAPI/OpenGL/OpenGLRendererAPI.h File Reference

```
#include "Vesper/Renderer/RendererAPI.h"
#include <glm/glm.hpp>
#include <memory>
```

Classes

- class [Vesper::OpenGLRendererAPI](#)
An implementation of the [RendererAPI](#) for OpenGL.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.28 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
00010     /// @brief An implementation of the RendererAPI for OpenGL.
00011     /// @note Should only be called by the RenderCommand class.
00012     class OpenGLRendererAPI : public RendererAPI
00013     {
00014     public:
00015         /// @brief Initializes the OpenGL rendering API.
00016         virtual void Init() override;
00017         /// @brief Sets the viewport dimensions for OpenGL.
00018         virtual void SetViewport(uint32_t x, uint32_t y, uint32_t width, uint32_t height) override;
00019         /// @brief Sets the clear color for OpenGL.
00020         virtual void SetClearColor(const glm::vec4& color) override;
00021         /// @brief Clears the OpenGL rendering buffers.
00022         virtual void Clear() override;
00023         /// @brief Draws indexed geometry using the provided vertex array in OpenGL.
00024         virtual void DrawIndexed(const Ref<VertexArray>& vertexArray, uint32_t indexCount = 0)
00025             override;
00026     };

```

11.29 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)

11.30 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](#)

11.30.1 Typedef Documentation

11.30.1.1 GLenum

```
typedef unsigned int GLenum
```

11.31 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 }
```

11.32 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.

11.33 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.

11.34 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 }
```

11.35 Vesper/src/RenderAPI/OpenGL/OpenGLUniformBuffer.cpp File Reference

```
#include "vzpch.h"
#include "OpenGLUniformBuffer.h"
#include <glad/glad.h>
```

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.36 Vesper/src/RenderAPI/OpenGL/OpenGLUniformBuffer.h File Reference

```
#include "Vesper/Renderer/UniformBuffer.h"
```

Classes

- class [Vesper::OpenGLUniformBuffer](#)

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.37 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 }
```

11.38 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 \(ShaderDataType type\)](#)

11.39 Vesper/src/RenderAPI/OpenGL/OpenGLVertexArray.h File Reference

```
#include "Vesper/Renderer/VertexArray.h"
```

Classes

- class [Vesper::OpenGLVertexArray](#)

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.40 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();
0010
0011         void Bind() const override;
0012         void Unbind() const override;
0013
0014         void AddVertexBuffer(const Ref<VertexBuffer>& vertexBuffer) override;
0015         void SetIndexBuffer(const Ref<IndexBuffer>& indexBuffer) override;
0016
0017         const std::vector<Ref<VertexBuffer>>& GetVertexBuffers() override { return m_VertexBuffers; }
0018         const Ref<IndexBuffer>& GetIndexBuffer() const override { return m_IndexBuffer; }
0019
0020     private:
0021         uint32_t m_RendererID;
0022         uint32_t m_VertexBufferIndex = 0;
0023         std::vector<Ref<VertexBuffer>> m_VertexBuffers;
0024         Ref<IndexBuffer> m_IndexBuffer;
0025
0026     };
0027
0028 }
```

11.41 Vesper/src/Vesper.h File Reference

Main header file for the [Vesper](#) engine. For use by clients in [Vesper](#) applications.

```
#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"
```

11.41.1 Detailed Description

Main header file for the [Vesper](#) engine. For use by clients in [Vesper](#) applications.

Author

Damon S. Green II

11.42 Vesper.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 /// @file Vesper.h
00004 /// @author Damon S. Green II
00005 /// @brief Main header file for the Vesper engine.
00006 /// For use by clients in Vesper applications
00007
00008 #include "Vesper/Core/Base.h"
00009
00010 #include "Vesper/App/Application.h"
00011 #include "Vesper/App/Layer.h"
00012
00013 #include "Vesper/Debug/Instrumentor.h"
00014
00015 #include "Vesper/Core/Log.h"
00016 #include "Vesper/Core/Random.h"
00017 #include "Vesper/Core/Color.h"
00018 #include "Vesper/Core/Timestep.h"
00019 #include "Vesper/Core/Timer.h"
00020 #include "Vesper/Core/Math.h"
00021
00022 #include "Vesper/Input/Input.h"
00023 ##include "Vesper/Input/InputContext.h"           /// @todo Input Context class
00024 ##include "Vesper/Input/InputAction.h"           /// @todo Input Action class
00025 #include "Vesper/Input/KeyCodes.h"
00026 #include "Vesper/Input/MouseButtonCodes.h"
00027
00028 /// GUI
00029 #include "Vesper/ImGui/ImGuiLayer.h"               /// @todo Abstract this to
00030     OpenGL/DirectX/Vulkan etc ImGui layers
00031 // -- Particle System (Temporary) -----
00032     renderer                                         /// Simple particle system for stress testing
00033 #include "Vesper/ParticleSystem/ParticleSystem.h"    /// Temporary starter particle system
00034 // -- Scene - Entity - Component - System -----
00035 #include "Vesper/Scene/Entity.h"
```

```

00036 #include "Vesper/Scene/ScriptableEntity.h"
00037 #include "Vesper/Scene/Components.h"
00038
00039 /// @todo Give scene a System[] variable
00040 #include "Vesper/Scene/Scene.h"
00041
00042 /// @todo Systems class
00043 // #include "Vesper/Scene/Systems.h"
00044
00045 /// @todo Static SystemsManager class
00046 // #include "Vesper/Scene/SystemsManager.h"
00047
00048 // -- Renderer-----
00049 #include "Vesper/Renderer/Renderer.h"
00050 #include "Vesper/Renderer/Renderer2D.h"
00051 #include "Vesper/Renderer/RenderCommand.h"
00052
00053 #include "Vesper/Renderer/Buffer.h"
00054 #include "Vesper/Renderer/Framebuffer.h"
00055 #include "Vesper/Renderer/Shader.h"
00056 #include "Vesper/Renderer/Texture.h"
00057 #include "Vesper/Renderer/SubTexture2D.h"
00058 #include "Vesper/Renderer/VertexArray.h"
00059
00060 #include "Vesper/Renderer/Camera.h"
00061 #include "Vesper/Renderer/EditorCamera.h"
00062 #include "Vesper/Renderer/OrthographicCamera.h"
00063 #include "Vesper/Renderer/OrthographicCameraController.h"
00064
00065
00066 // -- Renderer API-----

```

11.43 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.

11.44 Vesper/src/Vesper/App/Application.h File Reference

Controls the run of the [Vesper](#) Engine.

```

#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](#)

The core application class that manages the main loop, window, layers, and event handling.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

Functions

- [Application * Vesper::CreateApplication \(\)](#)

11.44.1 Detailed Description

Controls the run of the [Vesper](#) Engine.

Author

Damon S. Green II

Todo Convert all files/classes in App directory to Vesper::App namespace (create it)

11.44.2 Class Documentation

11.44.2.1 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 | |

11.45 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 #include "Vesper/Renderer/RendererAPI.h"
0010
0011 /// @file Application.h
0012 /// @author Damon S. Green II
0013 /// @brief Controls the run of the Vesper Engine
0014 /// @todo Convert all files/classes in App directory to Vesper::App namespace (create it)
0015
0016 namespace Vesper {
0017
0018
0019     /// WIP
0020     struct ApplicationSettings {
0021         std::string ApplicationName = "Vesper Application";
0022         std::string WorkingDirectory;
0023         RendererAPI::API RendererAPI = RendererAPI::API::OpenGL;
0024         uint32_t Width = 1280;
0025         uint32_t Height = 720;
0026         WindowMode Mode = WindowMode::Windowed;
0027         bool EnableImGui = true;
0028         bool EnableVSync = false;
0029     };
0030
0031     /// @class Application
0032     /// @brief The core application class that manages the main loop, window, layers, and event
0033     /// handling.
0034     /// @todo Update class to accept ApplicationSettings in constructor and verify its utility/use
0035     class Application
0036     {
0037     public:
0038         /// @brief Constructs the Application with the given name.
0039         ///
0040         /// @param name The name of the application. Defaults to an empty string.
0041         Application(const std::string& name = "");
0042
0043         virtual ~Application();
0044
0045         /// @brief Starts the main application loop.
0046         ///
0047         /// @todo Add Layer rendering into the main loop
0048         /// @todo Add separate threads for rendering and updating
0049         void Run();
0050
0051         /// @brief Handles incoming events and dispatches them to the appropriate handlers.
0052         void OnEvent(Event& e);
0053
0054         /// @brief Adds a layer to the application layer stack.
0055         void PushLayer(Layer* layer);
0056
0057         /// @brief Adds an overlay layer to the application layer stack.
0058         void PushOverlay(Layer* overlay);
0059
0060         /// @brief Closes the application.
0061         void Close();
0062
0063         /// @brief Retrieves the ImGui layer.
0064         ImGuiLayer* GetImGuiLayer() { return m_ImGuiLayer; }
0065
0066         /// @brief Retrieves the singleton instance of the Application.
0067         inline static Application& Get() { return *s_Instance; }
0068
0069         /// @brief Retrieves the application window.
0070         inline Window& GetWindow() { return *m_Window; }
0071     private:
0072
0073         /// @brief Event handler for window close events.
0074         bool OnWindowClose(WindowCloseEvent& e);
0075         /// @brief Event handler for window resize events.
0076         bool OnWindowResize(WindowResizeEvent& e);
0077
0078         /// @brief Scoped pointer to the applications underlying window.
0079         Scope<Window> m_Window;
0080         /// @brief ImGui layer for rendering GUI elements
0081         ImGuiLayer* m_ImGuiLayer;

```

```

00082     /// @brief Flag indicating whether the application is running.
00083     bool m_Running = true;
00084     /// @brief Flag indicating whether the application is minimized.
00085     bool m_Minimized = false;
00086     /// @brief Stack of layers managed by the application.
00087     LayerStack m_LayerStack;
00088     /// @brief Time of the last frame, used for calculating timestep.
00089     float m_LastFrameTime = 0.0f;
00090
00091     private:
00092         static Application* s_Instance;
00093     };
00094
00095     // To be defined in CLIENT
00096     Application* CreateApplication();
00097 }
```

11.46 Vesper/src/Vesper/App/EntryPoint.h File Reference

11.47 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
```

11.48 Vesper/src/Vesper/App/Layer.cpp File Reference

```
#include "vzpch.h"
#include "Layer.h"
```

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.49 Vesper/src/Vesper/App/Layer.h File Reference

Defines the Layer class for creating reusable application layers.

```
#include "Vesper/Core/Timestep.h"
#include "Vesper/Events/Event.h"
```

Classes

- class [Vesper::Layer](#)

Represents a reusable application layer that receives lifecycle callbacks (attach, detach, update, events, render, and ImGui render). Intended as a base class for concrete layers.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.49.1 Detailed Description

Defines the Layer class for creating reusable application layers.

Author

Damon S. Green II

Todo Add layer priority system (maybe?)

11.50 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 /// @file Layer.h
00007 /// @author Damon S. Green II
00008 /// @brief Defines the Layer class for creating reusable application layers.
00009 /// @todo Add layer priority system (maybe?)
00010
00011 namespace Vesper {
00012
00013     /// @class Layer
00014     /// @brief Represents a reusable application layer that receives lifecycle callbacks (attach,
00015             // detach, update, events, render, and ImGui render).
00016     /// Intended as a base class for concrete layers.
00017     class Layer
00018     {
00019         public:
00020             /// @brief Constructs a Layer with an optional name for debugging purposes.
00021             ///
00022             /// @param name The name of the layer, used for debugging, defaulted to "Layer".
00023             Layer(const std::string& name = "Layer");
00024
00025         virtual ~Layer();
```

```

00026     /// @brief Called when the layer is attached to the application.
00027     virtual void OnAttach() {};
00028
00029     /// @brief Called when the layer is detached from the application.
00030     virtual void OnDetach() {};
00031
00032     /// @brief Called every frame to update the layer with the given timestep.
00033     ///
00034     /// @param ts The timestep representing the time elapsed since the last update.
00035     virtual void OnUpdate(Timestep ts) {};
00036
00037     /// @brief Called when an event is dispatched to the layer.
00038     ///
00039     /// @param event The event being dispatched.
00040     virtual void OnEvent(Event& event){}
00041
00042     /// @brief Called when the layer should render its contents.
00043     ///
00044     /// @note Layers are responsible for their own rendering needs.
00045     /// @todo add layer rendering into the application's render loop
00046     virtual void OnRender() {};
00047
00048     /// @brief Called when the layer should render its ImGui components.
00049     virtual void OnImGuiRender() {};
00050
00051     /// @brief Retrieves the name of the layer for debugging purposes.
00052     inline const std::string& GetName() const { return m_DebugName; }
00053
00054     /// @brief The name of the layer assigned at creation, used for debugging.
00055     std::string m_DebugName;
00056 };
00057
00058 };
00059
00060 }
```

11.51 Vesper/src/Vesper/App/LayerStack.cpp File Reference

```
#include "vzpch.h"
#include "LayerStack.h"
```

Namespaces

- namespace **Vesper**

TEMPORARY.

11.52 Vesper/src/Vesper/App/LayerStack.h File Reference

Defines the LayerStack class for managing a stack of application layers.

```
#include "Vesper/Core/Base.h"
#include "Layer.h"
```

Classes

- class **Vesper::LayerStack**

Manages an ordered stack of [Layer](#) pointers. Layers can be pushed or popped and the stack can be iterated in forward or reverse order.

Namespaces

- namespace **Vesper**
TEMPORARY.

11.52.1 Detailed Description

Defines the LayerStack class for managing a stack of application layers.

Author

Damon S. Green II

11.53 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 /// @file LayerStack.h
00007 /// @author Damon S. Green II
00008 /// @brief Defines the LayerStack class for managing a stack of application layers.
00009
00010 namespace Vesper {
00011
00012     /// @class LayerStack
00013     /// @brief Manages an ordered stack of Layer pointers. Layers can be pushed or popped and the
00014     /// stack can be iterated in forward or reverse order.
00015     /// @todo Add layer priority system (maybe?)
00016     class LayerStack
00017     {
00018         public:
00019             LayerStack();
00020             ~LayerStack();
00021
00022             /// @brief Adds the layer to the stack at the position before the first overlay.
00023             /// @param layer Pointer to the layer to be added.
00024             void PushLayer(Layer* layer);
00025
00026             /// @brief Pushes the overlay layer on top of all other layers.
00027             /// @param overlay Pointer to the overlay layer to be added.
00028             void PushOverlay(Layer* overlay);
00029
00030             /// @brief Removes the specified layer from the stack.
00031             /// @param layer Pointer to the layer to be removed.
00032             void PopLayer(Layer* layer);
00033
00034             /// @brief Removes the specified overlay layer from the stack.
00035             /// @param overlay Pointer to the overlay layer to be removed.
00036             void PopOverlay(Layer* overlay);
00037
00038             /// @brief Returns an iterator to the beginning of the layer stack.
00039             std::vector<Layer*>::iterator begin() { return m_Layers.begin(); }
00040             /// @brief Returns an iterator to the end of the layer stack.
00041             std::vector<Layer*>::iterator end() { return m_Layers.end(); }
00042             /// @brief Returns a reverse iterator to the beginning of the layer stack.
00043             std::vector<Layer*>::reverse_iterator rbegin() { return m_Layers.rbegin(); }
00044             /// @brief Returns a reverse iterator to the end of the layer stack.
00045             std::vector<Layer*>::reverse_iterator rend() { return m_Layers.rend(); }
00046
00047
00048
00049
00050
00051     private:
00052         /// @brief Vector holding pointers to the layers in the stack.
00053         std::vector<Layer*> m_Layers;
00054         /// @brief Index indicating where to insert new layers (before overlays).
00055         unsigned int m_LayerInsertIndex = 0;
00056     };
00057
00058
00059 }
```

11.54 Vesper/src/Vesper/App/Window.h File Reference

Defines the abstract Window class and WindowProps struct for window management.

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
```

Classes

- struct [Vesper::WindowProps](#)
Holds the data for window configuration.
- class [Vesper::Window](#)
Abstract interface representing an application window.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

Enumerations

- enum class [Vesper::WindowMode](#) { [Vesper::Windowed](#) = 0 , [Vesper::Fullscreen](#) = 1 , [Vesper::Borderless](#) = 2 }
- WIP.*

11.54.1 Detailed Description

Defines the abstract Window class and WindowProps struct for window management.

Author

Damon S. Green II

Todo Add support for different window modes (windowed, fullscreen, borderless).

11.55 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 /// @file Window.h
00009 /// @author Damon S. Green II
00010 /// @brief Defines the abstract Window class and WindowProps struct for window management.
00011 /// @todo Add support for different window modes (windowed, fullscreen, borderless).
00012 namespace Vesper {
00013
00014     /// WIP
00015     enum class WindowMode
00016     {
00017         Windowed = 0,
00018         Fullscreen = 1,
00019         Borderless = 2
00020     };
00021
00022     /// @brief Holds the data for window configuration
00023     struct WindowProps {
00024         std::string Title;
00025         uint32_t Width;
00026         uint32_t Height;
00027
00028         /// @brief Constructor to initialize WindowProps with given or default values.
00029         ///
00030         /// @param title The title of the window.
00031         /// @param width The width of the window.
00032         /// @param height The height of the window.
00033         WindowProps(const std::string& title = "Vesper Engine",
00034             uint32_t width = 1600,
00035             uint32_t height = 900)
00036             : Title(title), Width(width), Height(height) {}
00037     };
00038
00039     /// @class Window
00040     /// @brief Abstract interface representing an application window.
00041     /// @todo Add Window mode functionality
00042     class Window
00043     {
00044     public:
00045         using EventCallbackFn = std::function<void(Event&)>;
00046
00047         virtual ~Window() {}
00048
00049         /// @brief Called every frame to update the window.
00050         virtual void OnUpdate() = 0;
00051
00052         /// @brief Retrieves the width of the window.
00053         virtual uint32_t GetWidth() const = 0;
00054         /// @brief Retrieves the height of the window.
00055         virtual uint32_t GetHeight() const = 0;
00056
00057         /// @brief Sets the callback function for window events.
00058         virtual void SetEventCallback(const EventCallbackFn& callback) = 0;
00059         /// @brief Sets whether vertical synchronization (VSync) is enabled.
00060         virtual void SetVSync(bool enabled) = 0;
00061         /// @brief Checks if vertical synchronization (VSync) is enabled.
00062         virtual bool IsVSync() const = 0;
00063
00064         /// @brief Retrieves the native window handle.
00065         virtual void* GetNativeWindow() const = 0;
00066
00067         /// @brief Creates a window instance with the specified properties.
00068         ///
00069         /// @param props The properties to initialize the window with.
00070         /// @return A scoped pointer to the created window instance.
00071         static Scope<Window> Create(const WindowProps& props = WindowProps());
00072
00073     };
00074 }

```

11.56 Vesper/src/Vesper/Core/Asserts.h File Reference

Provides assertion macros for debugging and error handling.

Macros

- `#define VZ_ASSERT(x, ...)`
ASSERT MACROS.
- `#define VZ_CORE_ASSERT(x, ...)`

11.56.1 Detailed Description

Provides assertion macros for debugging and error handling.

Author

Damon S. Green II

11.56.2 Macro Definition Documentation

11.56.2.1 VZ_ASSERT

```
#define VZ_ASSERT(
    x,
    ...)
```

ASSERT MACROS.

11.56.2.2 VZ_CORE_ASSERT

```
#define VZ_CORE_ASSERT(
    x,
    ...)
```

11.57 Asserts.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 /// @file Asserts.h
00004 /// @author Damon S. Green II
00005 /// @brief Provides assertion macros for debugging and error handling.
00006
00007
00008 /// ASSERT MACROS
00009 #ifdef VZ_DEBUG
00010 #define VZ_ENABLE_ASSERTS
00011 #endif
00012
00013 #ifndef VZ_ENABLE_ASSERTS
00014 /// @brief Asserts that the given condition is true. If not, logs an error message and triggers a
breakpoint.
00015 #define VZ_ASSERT(x, ...) { if(!(x)) { VZ_CORE_ERROR("Assertion Failed: {0}", __VA_ARGS__);
__debugbreak(); } }
00016 /// @brief Asserts that the given condition is true in core code. If not, logs an error message and
triggers a breakpoint.
00017 #define VZ_CORE_ASSERT(x, ...) { if(!(x)) { VZ_CORE_ERROR("Assertion Failed: {0}", __VA_ARGS__);
__debugbreak(); } }
00018 #else
00019 #define VZ_ASSERT(x, ...)
00020 #define VZ_CORE_ASSERT(x, ...)
00021 #endif
```

11.58 Vesper/src/Vesper/Core/Base.h File Reference

The base header file that includes core definitions and configurations for the [Vesper](#) engine.

```
#include "PlatformDetection.h"
#include "Config.h"
#include "Asserts.h"
#include "Defines_Macros.h"
```

11.58.1 Detailed Description

The base header file that includes core definitions and configurations for the [Vesper](#) engine.

Author

Damon S. Green II

Todo Move Core classes and helpers in the Core directory to a Vesper::Core namespace.

11.59 Base.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 /// @file Base.h
00004 /// @author Damon S. Green II
00005 /// @brief The base header file that includes core definitions and configurations for the Vesper
00006 /// @todo Move Core classes and helpers in the Core directory to a Vesper::Core namespace.
00007
00008 #include "PlatformDetection.h"
00009 #include "Config.h"
00010 #include "Asserts.h"
00011 #include "Defines_Macros.h"
```

11.60 Vesper/src/Vesper/Core/Color.h File Reference

Provides commonly used colors and color manipulation functions.

```
#include <glm/glm.hpp>
```

Namespaces

- namespace [Vesper](#)
TEMPORARY.
- namespace [Vesper::Color](#)
Provides commonly used colors and color manipulation functions.

Functions

- static `glm::vec4 Vesper::Color::White ()`
Returns a vec4 representing the color white.
- static `glm::vec4 Vesper::Color::Black ()`
Returns a vec4 representing the color black.
- static `glm::vec4 Vesper::Color::Gray ()`
Returns a vec4 representing the color gray.
- static `glm::vec4 Vesper::Color::Red ()`
Returns a vec4 representing the color red.
- static `glm::vec4 Vesper::Color::Orange ()`
Returns a vec4 representing the color orange.
- static `glm::vec4 Vesper::Color::Yellow ()`
Returns a vec4 representing the color yellow.
- static `glm::vec4 Vesper::Color::Green ()`
Returns a vec4 representing the color green.
- static `glm::vec4 Vesper::Color::Blue ()`
Returns a vec4 representing the color blue.
- static `glm::vec4 Vesper::Color::Indigo ()`
Returns a vec4 representing the color indigo.
- static `glm::vec4 Vesper::Color::Purple ()`
Returns a vec4 representing the color purple.
- static `glm::vec4 Vesper::Color::Cyan ()`
Returns a vec4 representing the color cyan.
- static `glm::vec4 Vesper::Color::Magenta ()`
Returns a vec4 representing the color magenta.
- static `glm::vec4 Vesper::Color::Pink ()`
Returns a vec4 representing the color pink.
- static `glm::vec4 Vesper::Color::Brown ()`
Returns a vec4 representing the color brown.
- static `glm::vec4 Vesper::Color::Transparent ()`
Returns a vec4 representing a fully transparent color.
- static `glm::vec4 Vesper::Color::StripAlpha (const glm::vec4 &color)`
Strips the alpha component from the given color, setting it to 1.0 (fully opaque).
- static `glm::vec4 Vesper::Color::SetAlpha (const glm::vec4 &color, float alpha=0.0f)`
Sets the alpha component of the given color to the specified value.

11.60.1 Detailed Description

Provides commonly used colors and color manipulation functions.

Author

Damon S. Green II

11.61 Color.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <glm/glm.hpp>
00004
00005 /// @file Color.h
00006 /// @author Damon S. Green II
00007 /// @brief Provides commonly used colors and color manipulation functions.
00008
00009 namespace Vesper {
00010
00011     /// @namespace Vesper::Color
00012     /// @brief Provides commonly used colors and color manipulation functions.
00013     namespace Color {
00014
00015         /// @brief Returns a vec4 representing the color white.
00016         static glm::vec4 White() { return glm::vec4(1.0f, 1.0f, 1.0f, 1.0f); }
00017         /// @brief Returns a vec4 representing the color black.
00018         static glm::vec4 Black() { return glm::vec4(0.0f, 0.0f, 0.0f, 1.0f); }
00019         /// @brief Returns a vec4 representing the color gray.
00020         static glm::vec4 Gray() { return glm::vec4(0.5f, 0.5f, 0.5f, 1.0f); }
00021
00022         /// @brief Returns a vec4 representing the color red.
00023         static glm::vec4 Red() { return glm::vec4(1.0f, 0.0f, 0.0f, 1.0f); }
00024         /// @brief Returns a vec4 representing the color orange.
00025         static glm::vec4 Orange() { return glm::vec4(1.0f, 0.5f, 0.0f, 1.0f); }
00026         /// @brief Returns a vec4 representing the color yellow.
00027         static glm::vec4 Yellow() { return glm::vec4(1.0f, 1.0f, 0.0f, 1.0f); }
00028         /// @brief Returns a vec4 representing the color green.
00029         static glm::vec4 Green() { return glm::vec4(0.0f, 1.0f, 0.0f, 1.0f); }
00030         /// @brief Returns a vec4 representing the color blue.
00031         static glm::vec4 Blue() { return glm::vec4(0.0f, 0.0f, 1.0f, 1.0f); }
00032         /// @brief Returns a vec4 representing the color indigo.
00033         static glm::vec4 Indigo() { return glm::vec4(0.29f, 0.0f, 0.51f, 1.0f); }
00034         /// @brief Returns a vec4 representing the color purple.
00035         static glm::vec4 Purple() { return glm::vec4(0.5f, 0.0f, 0.5f, 1.0f); }
00036
00037         /// @brief Returns a vec4 representing the color cyan.
00038         static glm::vec4 Cyan() { return glm::vec4(0.0f, 1.0f, 1.0f, 1.0f); }
00039         /// @brief Returns a vec4 representing the color magenta.
00040         static glm::vec4 Magenta() { return glm::vec4(1.0f, 0.0f, 1.0f, 1.0f); }
00041         /// @brief Returns a vec4 representing the color pink.
00042         static glm::vec4 Pink() { return glm::vec4(1.0f, 0.75f, 0.8f, 1.0f); }
00043         /// @brief Returns a vec4 representing the color brown.
00044         static glm::vec4 Brown() { return glm::vec4(0.6f, 0.4f, 0.2f, 1.0f); }
00045         /// @brief Returns a vec4 representing a fully transparent color.
00046         static glm::vec4 Transparent() { return glm::vec4(0.0f, 0.0f, 0.0f, 0.0f); }
00047
00048         /// @brief Strips the alpha component from the given color, setting it to 1.0 (fully opaque).
00049         static glm::vec4 StripAlpha(const glm::vec4& color) { return glm::vec4(color.x, color.y,
00050             color.z, 1.0f); }
00051         /// @brief Sets the alpha component of the given color to the specified value.
00052         static glm::vec4 SetAlpha(const glm::vec4& color, float alpha = 0.0f) { return
00053             glm::vec4(color.x, color.y, color.z, alpha); }
00054     }

```

11.62 Vesper/src/Vesper/Core/Config.h File Reference

Configuration macros for the [Vesper](#) engine and editor.

Macros

- `#define VZ_DEFAULT_TEXTURE Texture2D::Create("Resources/Textures/Checkerboard.png")`
- Editor Configurations.*

11.62.1 Detailed Description

Configuration macros for the [Vesper](#) engine and editor.

Author

Damon S. Green II

Todo : Implement Default Graphics API

11.62.2 Macro Definition Documentation

11.62.2.1 VZ_DEFAULT_TEXTURE

```
#define VZ_DEFAULT_TEXTURE Texture2D::Create("Resources/Textures/Checkerboard.png")
```

Editor Configurations.

11.63 Config.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 /// @file Config.h
00004 /// @author Damon S. Green II
00005 /// @brief Configuration macros for the Vesper engine and editor.
00006 /// @todo: Implement Default Graphics API
00007
00008
00009 /// Editor Configurations
0010 #ifdef VZ_EDITOR_USE_DEFAULT_SCENE
0011 #define VZ_EDITOR_DEFAULT_SCENE "Resources/Scenes/TriColored3DCubeAndSpriteAnims.vesper"
0012 #endif
0013
0014 #define VZ_DEFAULT_TEXTURE Texture2D::Create("Resources/Textures/Checkerboard.png")
```

11.64 Vesper/src/Vesper/Core/Defines_Macros.h File Reference

Provides commonly used macros along with project wide typedefs.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

Macros

- `#define BIT(x)`
Macro to shift 1 by x positions to create a bitmask.
- `#define VZ_BIND_EVENT_FN(fn)`
Binds a member function as an event handler.
- `#define BIND_EVENT_FN(x)`
Binds the given member function of the Application class as an event handler.
- `#define EVENT_CLASS_TYPE(type)`
Macro to define event type in event subclasses.
- `#define EVENT_CLASS_CATEGORY(category)`
Macro to define event category in event subclasses.

Typedefs

- `template<typename T>`
`using Vesper::Scope = std::unique_ptr<T>`
A smart pointer type representing exclusive ownership of an object.
- `template<typename T>`
`using Vesper::Ref = std::shared_ptr<T>`
A smart pointer type representing shared ownership of an object.

Functions

- `template<typename T, typename... Args>`
`constexpr Scope< T > Vesper::CreateScope (Args &&... args)`
Creates a Scope (unique_ptr) for the given type and constructor arguments.
- `template<typename T, typename... Args>`
`constexpr Ref< T > Vesper::CreateRef (Args &&... args)`
Creates a Ref (shared_ptr) for the given type and constructor arguments.

11.64.1 Detailed Description

Provides commonly used macros along with project wide typedefs.

Author

Damon S. Green II

11.64.2 Macro Definition Documentation

11.64.2.1 BIND_EVENT_FN

```
#define BIND_EVENT_FN(
    x)
```

Value:

```
std::bind(&Application::x, this, std::placeholders::_1)
```

Binds the given member function of the Application class as an event handler.

Parameters

| | |
|---|------------------------------|
| x | The member function to bind. |
|---|------------------------------|

11.64.2.2 BIT

```
#define BIT(
    x)
```

Value:

```
(1 << x)
```

Macro to shift 1 by x positions to create a bitmask.

Parameters

| | |
|---|-----------------------------------|
| x | The number of positions to shift. |
|---|-----------------------------------|

Returns

The resulting bitmask.

Note

This macro is typically used for defining event categories.

Warning

Be cautious when using this macro with values greater than or equal to the number of bits in an integer, as it may lead to undefined behavior.

Attention

This macro is not type-safe; consider using constexpr functions or enum classes for better type safety.

11.64.2.3 EVENT_CLASS_CATEGORY

```
#define EVENT_CLASS_CATEGORY(
    category)
```

Value:

```
virtual int GetCategoryFlags() const override { return category; }
```

Macro to define event category in event subclasses.

Parameters

| | |
|-----------------|-------------------------------|
| <i>category</i> | The event category to define. |
|-----------------|-------------------------------|

11.64.2.4 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; }
```

Macro to define event type in event subclasses.

Parameters

| | |
|-------------|---------------------------|
| <i>type</i> | The event type to define. |
|-------------|---------------------------|

```
00025 #define EVENT_CLASS_TYPE(type) static EventType GetStaticType() { return EventType::##type; } \
00026     virtual EventType GetEventType() const override { return \
00027         GetStaticType(); } \
00028     virtual const char* GetName() const override { return #type; }
```

11.64.2.5 VZ_BIND_EVENT_FN

```
#define VZ_BIND_EVENT_FN(
    fn)
```

Value:

```
[this](auto&&... args) -> decltype(auto) { return this->fn(std::forward<decltype(args)>(args)...); }
```

Binds a member function as an event handler.

Parameters

| | |
|-----------|------------------------------|
| <i>fn</i> | The member function to bind. |
|-----------|------------------------------|

11.65 Defines_Macros.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 /// @file Defines_Macros.h
00003 /// @author Damon S. Green II
00004 /// @brief Provides commonly used macros along with project wide typedefs.
00005
00006
00007 /// @brief Macro to shift 1 by x positions to create a bitmask.
00008 /// @param x The number of positions to shift.
00009 /// @return The resulting bitmask.
00010 /// @note This macro is typically used for defining event categories.
00011 /// @warning Be cautious when using this macro with values greater than or equal to the number of bits
00012 /// in an integer, as it may lead to undefined behavior.
00012 /// @attention This macro is not type-safe; consider using constexpr functions or enum classes for
00013 better type safety.
00013 #define BIT(x) (1 << x)
00014
00015 /// @brief Binds a member function as an event handler.
00016 /// @param fn The member function to bind.
00017 #define VZ_BIND_EVENT_FN(fn) [this](auto&&... args) -> decltype(auto) { return
00018     this->fn(std::forward<decltype(args)>(args)...); }
00019
00020 /// @brief Binds the given member function of the Application class as an event handler.
00021 /// @param x The member function to bind.
00021 #define BIND_EVENT_FN(x) std::bind(&Application::x, this, std::placeholders::_1)
00022
00023 /// @brief Macro to define event type in event subclasses.
00024 /// @param type The event type to define.
00025 #define EVENT_CLASS_TYPE(type) static EventType GetStaticType() { return EventType::##type; }
00026             virtual EventType GetEventType() const override { return
00027                 GetStaticType(); }
00028             virtual const char* GetName() const override { return #type; }
00029
00030 /// @brief Macro to define event category in event subclasses.
00031 /// @param category The event category to define.
00031 #define EVENT_CLASS_CATEGORY(category) virtual int GetCategoryFlags() const override { return
00032     category; }
00033
00034 /// @namespace Vesper
00035 /// @brief The main namespace for the Vesper engine.
00036 namespace Vesper
00037 {
00038     /// @brief A smart pointer type representing exclusive ownership of an object.
00039     ///
00040     /// @tparam T The type of object to manage.
00041     /// @note This is an alias for std::unique_ptr.
00042     template <typename T>
00043     using Scope = std::unique_ptr<T>;
00044
00045     /// @brief Creates a Scope (unique_ptr) for the given type and constructor arguments.
00046     ///
00047     /// @tparam T The type of object to create.
00048     /// @tparam Args The types of constructor arguments.
00049     /// @param args The constructor arguments.
00050     /// @return A Scope (unique_ptr) managing the created object.
00051     template <typename T, typename... Args>
00052     constexpr Scope<T> CreateScope(Args&&... args)
00053     {
00054         return std::make_unique<T>(std::forward<Args>(args)...);
00055     }
00056
00057
00058     /// @brief A smart pointer type representing shared ownership of an object.
00059     ///
00060     /// @tparam T The type of object to manage.
00061     /// @note This is an alias for std::shared_ptr.
00062     template <typename T>
00063     using Ref = std::shared_ptr<T>;
00064
00065     /// @brief Creates a Ref (shared_ptr) for the given type and constructor arguments.
00066     ///
00067     /// @tparam T The type of object to create.
00068     /// @tparam Args The types of constructor arguments.
00069     /// @param args The constructor arguments.
00070     /// @return A Ref (shared_ptr) managing the created object.
00071     template <typename T, typename... Args>
00072     constexpr Ref<T> CreateRef(Args&&... args)
00073     {
00074         return std::make_shared<T>(std::forward<Args>(args)...);
00075     }
00076
00077 }
```

11.66 Vesper/src/Vesper/Core/Log.cpp File Reference

Implements the logging system for the [Vesper](#) engine.

```
#include "vzpch.h"
#include "Log.h"
#include "spdlog/sinks/stdout_color_sinks.h"
```

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.66.1 Detailed Description

Implements the logging system for the [Vesper](#) engine.

Author

Damon S. Green II

See also

[Log.h](#)

11.67 Vesper/src/Vesper/Core/Log.h File Reference

Declares the logging system for the [Vesper](#) engine.

```
#include "Base.h"
#include "spdlog/spdlog.h"
#include "spdlog/fmt/ostr.h"
```

Classes

- class [Vesper::Log](#)
A logging utility class for the [Vesper](#) engine.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

Macros

- `#define VZ_CORE_TRACE(...)`
trace: lowest level, for detailed debug information
- `#define VZ_CORE_INFO(...)`
info: general information about application flow
- `#define VZ_CORE_WARN(...)`
warn: indicates a potential issue or important event
- `#define VZ_CORE_ERROR(...)`
error: indicates an error that has occurred
- `#define VZ_CORE_FATAL(...)`
fatal: indicates a critical error that may cause application termination
- `#define VZ_TRACE(...)`
trace: lowest level, for detailed debug information
- `#define VZ_INFO(...)`
info: general information about application flow
- `#define VZ_WARN(...)`
warn: indicates a potential issue or important event
- `#define VZ_ERROR(...)`
error: indicates an error that has occurred
- `#define VZ_FATAL(...)`
fatal: indicates a critical error that may cause application termination

11.67.1 Detailed Description

Declares the logging system for the [Vesper](#) engine.

Author

Damon S. Green II

11.67.2 Macro Definition Documentation

11.67.2.1 VZ_CORE_ERROR

```
#define VZ_CORE_ERROR(
    ...)
```

Value:

```
::Vesper::Log::GetCoreLogger() -> error(__VA_ARGS__)
```

error: indicates an error that has occurred

11.67.2.2 VZ_CORE_FATAL

```
#define VZ_CORE_FATAL(
    ...)
```

Value:

```
::Vesper::Log::GetCoreLogger() -> critical(__VA_ARGS__)
```

fatal: indicates a critical error that may cause application termination

11.67.2.3 VZ_CORE_INFO

```
#define VZ_CORE_INFO(  
    ...)
```

Value:

```
::Vesper::Log::GetCoreLogger() -> info(__VA_ARGS__)
```

info: general information about application flow

11.67.2.4 VZ_CORE_TRACE

```
#define VZ_CORE_TRACE(  
    ...)
```

Value:

```
::Vesper::Log::GetCoreLogger() -> trace(__VA_ARGS__)
```

trace: lowest level, for detailed debug information

11.67.2.5 VZ_CORE_WARN

```
#define VZ_CORE_WARN(  
    ...)
```

Value:

```
::Vesper::Log::GetCoreLogger() -> warn(__VA_ARGS__)
```

warn: indicates a potential issue or important event

11.67.2.6 VZ_ERROR

```
#define VZ_ERROR(  
    ...)
```

Value:

```
::Vesper::Log::GetClientLogger() -> error(__VA_ARGS__)
```

error: indicates an error that has occurred

11.67.2.7 VZ_FATAL

```
#define VZ_FATAL(  
    ...)
```

Value:

```
::Vesper::Log::GetClientLogger() -> critical(__VA_ARGS__)
```

fatal: indicates a critical error that may cause application termination

11.67.2.8 VZ_INFO

```
#define VZ_INFO(
    ...)
```

Value:

```
::Vesper::Log::GetClientLogger() ->info(__VA_ARGS__)
```

info: general information about application flow

11.67.2.9 VZ_TRACE

```
#define VZ_TRACE(
    ...)
```

Value:

```
::Vesper::Log::GetClientLogger() ->trace(__VA_ARGS__)
```

trace: lowest level, for detailed debug information

11.67.2.10 VZ_WARN

```
#define VZ_WARN(
    ...)
```

Value:

```
::Vesper::Log::GetClientLogger() ->warn(__VA_ARGS__)
```

warn: indicates a potential issue or important event

11.68 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 /// @file Log.h
00007 /// @author Damon S. Green II
00008 /// @brief Declares the logging system for the Vesper engine.
00009
00010 namespace Vesper {
00011
00012     /// @class Log
00013     /// @brief A logging utility class for the Vesper engine.
00014     /// @todo Rethink logging flow with Macros and possibly implement different loggers for different
00015     /// modules.
00016     class Log
00017     {
00018         public:
00019             /// @brief Initializes the logging system.
00020             static void Init();
00021
00022             /// @brief Returns the core logger instance.
00023             inline static std::shared_ptr<spdlog::logger>& GetCoreLogger() { return s_CoreLogger; }
00024             /// @brief Returns the client logger instance.
00025             inline static std::shared_ptr<spdlog::logger>& GetClientLogger() { return s_ClientLogger; }
00026         private:
00027             static std::shared_ptr<spdlog::logger> s_CoreLogger;
```

```

00028     static std::shared_ptr<spdlog::logger> s_ClientLogger;
00029 };
00030
00031
00032 }
00033
00034 /// @brief trace: lowest level, for detailed debug information
00035 #define VZ_CORE_TRACE(...)      ::Vesper::Log::GetCoreLogger()->trace(__VA_ARGS__)
00036 /// @brief info: general information about application flow
00037 #define VZ_CORE_INFO(...)       ::Vesper::Log::GetCoreLogger()->info(__VA_ARGS__)
00038 /// @brief warn: indicates a potential issue or important event
00039 #define VZ_CORE_WARN(...)      ::Vesper::Log::GetCoreLogger()->warn(__VA_ARGS__)
00040 /// @brief error: indicates an error that has occurred
00041 #define VZ_CORE_ERROR(...)     ::Vesper::Log::GetCoreLogger()->error(__VA_ARGS__)
00042 /// @brief fatal: indicates a critical error that may cause application termination
00043 #define VZ_CORE_FATAL(...)      ::Vesper::Log::GetCoreLogger()->critical(__VA_ARGS__)
00044
00045 /// @brief trace: lowest level, for detailed debug information
00046 #define VZ_TRACE(...)          ::Vesper::Log::GetClientLogger()->trace(__VA_ARGS__)
00047 /// @brief info: general information about application flow
00048 #define VZ_INFO(...)           ::Vesper::Log::GetClientLogger()->info(__VA_ARGS__)
00049 /// @brief warn: indicates a potential issue or important event
00050 #define VZ_WARN(...)           ::Vesper::Log::GetClientLogger()->warn(__VA_ARGS__)
00051 /// @brief error: indicates an error that has occurred
00052 #define VZ_ERROR(...)          ::Vesper::Log::GetClientLogger()->error(__VA_ARGS__)
00053 /// @brief fatal: indicates a critical error that may cause application termination
00054 #define VZ_FATAL(...)          ::Vesper::Log::GetClientLogger()->critical(__VA_ARGS__)

```

11.69 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)`

Decomposes a transformation matrix into its translation, rotation, and scale components.

11.69.1 Macro Definition Documentation

11.69.1.1 GLM_ENABLE_EXPERIMENTAL

```
#define GLM_ENABLE_EXPERIMENTAL
```

11.70 Vesper/src/Vesper/Core/Math.h File Reference

Provides mathematical utility functions.

```
#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)

Decomposes a transformation matrix into its translation, rotation, and scale components.

11.70.1 Detailed Description

Provides mathematical utility functions.

Author

Damon S. Green II

11.71 Math.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <glm/glm.hpp>
00004
00005 /// @file Math.h
00006 /// @author Damon S. Green II
00007 /// @brief Provides mathematical utility functions.
00008
00009 namespace Vesper::Math {
0010
0011     /// @brief Decomposes a transformation matrix into its translation, rotation, and scale
0012     /// components.
0013     bool DecomposeTransform(const glm::mat4& transform, glm::vec3& translation, glm::vec3& rotation,
0014                             glm::vec3& scale);
0015 }
```

11.72 Vesper/src/Vesper/Core/PlatformDetection.h File Reference

Detects the current platform and defines corresponding macros.

```
#include <memory>
```

11.72.1 Detailed Description

Detects the current platform and defines corresponding macros.

Author

Damon S. Green II

11.73 PlatformDetection.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 /// @file PlatformDetection.h
00004 /// @author Damon S. Green II
00005 /// @brief Detects the current platform and defines corresponding macros.
00006
00007 #include <memory>
00008
00009 // Platform detection
00010 #ifdef _WIN32
00011     /*Windows x64/x86*/
00012 #ifdef _WIN64
00013     /*Windows x64 */
00014 #ifndef VZ_PLATFORM_WINDOWS
00015 #define VZ_PLATFORM_WINDOWS
00016 #endif
00017 #else
00018     /*Windows x86*/
00019 #error "x86 Builds are not supported!"
00020 #endif
00021 #elif defined(__APPLE__) || defined(__MACH__)
00022 #include <TargetConditionals.h>
00023 /*Apple platforms */
00024 #if TARGET_IPHONE_SIMULATOR == 1
00025 #error "IOS Simulator is not supported!"
00026 #elif TARGET_OS_IPHONE == 1
00027 #define VZ_PLATFORM_IOS
00028 #error "IOS is not supported!"
00029 #elif TARGET_OS_MAC == 1
00030 #define VZ_PLATFORM_MACOS
00031 #error "MacOS is not supported!"
00032 #else
00033 #error "Unknown Apple platform!"
00034 #endif
00035 #elif defined(__ANDROID__)
00036 #define VZ_PLATFORM_ANDROID
00037 #error "Android is not supported!"
00038 #elif defined(__linux__)
00039 #define VZ_PLATFORM_LINUX
00040 #error "Linux is not supported!"
00041 #else
00042     /*Unknown compiler/platform*/
00043 #error "Unknown platform!"
00044 #endif // End of platform detection
```

11.74 Vesper/src/Vesper/Core/Random.h File Reference

Provides random number generation utilities.

```
#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)

11.74.1 Detailed Description

Provides random number generation utilities.

Author

Damon S. Green II

11.75 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 /// @file Random.h
00009 /// @author Damon S. Green II
00010 /// @brief Provides random number generation utilities.
00011
00012 namespace Vesper {
00013 }
```

```

00014     namespace Random {
00015
00016         inline std::mt19937& GetRNG() {
00017             static thread_local std::mt19937 rng{ std::random_device{}() };
00018             return rng;
00019         }
00020
00021         inline void Seed(uint32_t seed) {
00022             GetRNG().seed(seed);
00023         }
00024
00025         inline uint32_t UInt1(uint32_t max) {
00026             VZ_PROFILE_FUNCTION();
00027             std::uniform_int_distribution<uint32_t> dist(0, max - 1);
00028             return dist(GetRNG());
00029         }
00030
00031         inline bool Bool1(float trueChance) {
00032             VZ_PROFILE_FUNCTION();
00033             std::uniform_real_distribution<float> dist(0.0f, 1.0f);
00034             return dist(GetRNG()) < trueChance;
00035         }
00036
00037         inline unsigned char Char() {
00038             VZ_PROFILE_FUNCTION();
00039             std::uniform_int_distribution<int> dist(0, 255);
00040             return static_cast<unsigned char>(dist(GetRNG()));
00041         }
00042
00043         inline std::string String(size_t length) {
00044             VZ_PROFILE_FUNCTION();
00045             const char charset[] =
00046                 "0123456789";
00047                 "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
00048                 "abcdefghijklmnopqrstuvwxyz";
00049             const size_t max_index = (sizeof(charset) - 1);
00050             std::string str(length, 0);
00051             for (size_t i = 0; i < length; ++i) {
00052                 str[i] = charset[UInt1(static_cast<uint32_t>(max_index))];
00053             }
00054             return str;
00055         }
00056
00057         inline std::string HexString(size_t length) {
00058             VZ_PROFILE_FUNCTION();
00059             const char charset[] =
00060                 "0123456789";
00061                 "ABCDEF";
00062             const size_t max_index = (sizeof(charset) - 1);
00063             std::string str(length, 0);
00064             for (size_t i = 0; i < length; ++i) {
00065                 str[i] = charset[UInt1(static_cast<uint32_t>(max_index))];
00066             }
00067             return str;
00068         }
00069
00070         inline std::string UUID() {
00071             VZ_PROFILE_FUNCTION();
00072             return HexString(8) + "-" + HexString(4) + "-" + HexString(4) + "-" + HexString(4) + "-" + HexString(12);
00073         }
00074
00075         // Returns a float in the range [0.0, 1.0] = [inclusive, exclusive]
00076         inline float Float1() {
00077             VZ_PROFILE_FUNCTION();
00078             static thread_local std::uniform_real_distribution<float> dist(0.0f, 1.0f);
00079             return dist(GetRNG());
00080         }
00081
00082         // Returns a float in the range [min, max]
00083         // If you need max to be possible, use the "_Inclusive" variant which advances the upper bound
00084         inline float RangeF1(float min, float max) {
00085             VZ_PROFILE_FUNCTION();
00086             if (min > max) std::swap(min, max);
00087             std::uniform_real_distribution<float> dist(min, max);
00088             return dist(GetRNG());
00089         }
00090
00091         // Inclusive-upper variant: returns value in [min, max] (max possible)
00092         inline float RangeF1_Inclusive(float min, float max) {
00093             if (min > max) std::swap(min, max);
00094             float upper = std::nextafter(max, std::numeric_limits<float>::infinity());
00095             std::uniform_real_distribution<float> dist(min, upper);
00096             return dist(GetRNG());
00097         }
00098
00099         // Returns a 2D float vector with each component in the range [0.0, 1.0] = [inclusive,

```

```

    exclusive]
00100     inline glm::vec2 Float2() {
00101         return glm::vec2{ Float1(), Float1() };
00102     }
00103
00104     // Returns a 2D float vector with each component in the range [min, max]
00105     inline glm::vec2 RangeF2(float min, float max) {
00106         return glm::vec2{ RangeF1(min, max), RangeF1(min, max) };
00107     }
00108
00109     // Returns a 2D float vector with each component in the range defined by the components of the
00110     given vector
00111     inline glm::vec2 RangeF2(float min1, float max1, float min2, float max2) {
00112         return glm::vec2{ RangeF1(min1, max1), RangeF1(min2, max2) };
00113     }
00114
00115     // Returns a 2D float vector with each component in the specified ranges
00116     inline glm::vec2 RangeF2(const glm::vec2& minRange, const glm::vec2& maxRange) {
00117         return glm::vec2{ RangeF1(minRange.x, maxRange.x), RangeF1(minRange.y, maxRange.y) };
00118     }
00119
00120     // Returns a 3D float vector with each component in the range [0.0, 1.0] = [inclusive,
00121     exclusive]
00122     inline glm::vec3 Float3() {
00123         return glm::vec3{ Float1(), Float1(), Float1() };
00124     }
00125
00126     // Returns a 3D float vector with each component in the range [min, max]
00127     inline glm::vec3 RangeF3(float min, float max) {
00128         return glm::vec3{ RangeF1(min, max), RangeF1(min, max), RangeF1(min, max) };
00129     }
00130
00131     // Returns a 3D float vector with each component in the specified ranges
00132     inline glm::vec3 RangeF3(float min1, float max1, float min2, float max2, float
00133     min3, float max3) {
00134         return glm::vec3{ RangeF1(min1, max1), RangeF1(min2, max2), RangeF1(min3, max3) };
00135
00136     // Returns a 3D float vector with each component in the specified ranges
00137     inline glm::vec3 RangeF3(const glm::vec2& range1, const glm::vec2& range2, const glm::vec2&
00138     range3) {
00139         return glm::vec3{ RangeF1(range1.x, range1.y), RangeF1(range2.x, range2.y),
00140             RangeF1(range3.x, range3.y) };
00141
00142     // Returns a 4D float vector with each component in the range [0.0, 1.0] = [inclusive,
00143     exclusive]
00144     inline glm::vec4 Float4() {
00145         return glm::vec4{ Float1(), Float1(), Float1(), Float1() };
00146     }
00147
00148     // Returns a 4D float vector with each component in the range [min, max]
00149     inline glm::vec4 RangeF4(float min, float max) {
00150         return glm::vec4{ RangeF1(min, max), RangeF1(min, max), RangeF1(min,
00151             max), RangeF1(min, max) };
00152     }
00153 }
00154 }
```

11.76 Vesper/src/Vesper/Core/Timer.h File Reference

Declares a Timer class for measuring elapsed time.

```
#include <chrono>
```

Namespaces

- namespace **Vesper**
TEMPORARY.

11.76.1 Detailed Description

Declares a Timer class for measuring elapsed time.

Author

Damon S. Green II

11.77 Timer.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <chrono>
00003
00004 /// @file Timer.h
00005 /// @author Damon S. Green II
00006 /// @brief Declares a Timer class for measuring elapsed time.
00007
00008 namespace Vesper
00009 {
00010     /// @todo Finish timer class
00011
00012
00013
00014 }
```

11.78 Vesper/src/Vesper/Core/Timestep.h File Reference

Declares the Timestep class representing a time step in seconds.

Classes

- class [Vesper::Timestep](#)
Represents a time step in seconds.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.78.1 Detailed Description

Declares the Timestep class representing a time step in seconds.

Author

Damon S. Green II

11.79 Timestep.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 /// @file Timestep.h
00004 /// @author Damon S. Green II
00005 /// @brief Declares the Timestep class representing a time step in seconds.
00006
00007 namespace Vesper {
00008
00009     /// @class Timestep
0010     /// @brief Represents a time step in seconds.
0011     class Timestep
0012     {
0013         public:
0014             /// @brief Constructs a Timestep with the given time in seconds.
0015             ///
0016             /// @param time The time in seconds.
0017
0018             Timestep(float time = 0.0f)
0019                 : m_Time(time)
0020             {
0021             }
0022
0023             operator float() const { return m_Time; }
0024
0025             /// @brief Returns the time in seconds.
0026             float GetSeconds() const { return m_Time; }
0027             /// @brief Returns the time in milliseconds.
0028             float GetMilliseconds() const { return m_Time * 1000.0f; }
0029         private:
0030             float m_Time;
0031     };
0032 }
```

11.80 Vesper/src/Vesper/Debug/Instrumentor.h File Reference

Provides instrumentation and profiling utilities.

```
#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])`

11.80.1 Detailed Description

Provides instrumentation and profiling utilities.

Author

TheCherno

Damon S. Green II

11.80.2 Class Documentation

11.80.2.1 struct Vesper::ProfileResult

Class Members

| | | |
|-----------|----------|--|
| long long | End | |
| string | Name | |
| long long | Start | |
| uint32_t | ThreadID | |

11.80.2.2 struct Vesper::InstrumentationSession

Class Members

| | | |
|--------|------|--|
| string | Name | |
|--------|------|--|

11.80.2.3 struct InstrumentorUtils::ChangeResult

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

Class Members

| | | |
|------|--------|--|
| char | Data ↪ | |
| | [N] | |

11.80.3 Macro Definition Documentation

11.80.3.1 VZ_FUNC_SIG

```
#define VZ_FUNC_SIG "VZ_FUNC_SIG unknown!"
```

11.80.3.2 VZ_PROFILE

```
#define VZ_PROFILE 1
```

11.80.3.3 VZ_PROFILE_BEGIN_SESSION

```
#define VZ_PROFILE_BEGIN_SESSION(
    name,
    filepath)
```

Value:

```
::Vesper::Instrumentor::Get().BeginSession(name, filepath)
```

11.80.3.4 VZ_PROFILE_END_SESSION

```
#define VZ_PROFILE_END_SESSION()
```

Value:

```
::Vesper::Instrumentor::Get().EndSession()
```

11.80.3.5 VZ_PROFILE_FUNCTION

```
#define VZ_PROFILE_FUNCTION()
```

Value:

```
VZ_PROFILE_SCOPE(VZ_FUNC_SIG)
```

11.80.3.6 VZ_PROFILE_SCOPE

```
#define VZ_PROFILE_SCOPE(
    name)
```

Value:

```
VZ_PROFILE_SCOPE_LINE(name, __LINE__)
```

11.80.3.7 VZ_PROFILE_SCOPE_LINE

```
#define VZ_PROFILE_SCOPE_LINE(
    name,
    line)
```

Value:

```
VZ_PROFILE_SCOPE_LINE2(name, line)
```

11.80.3.8 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)
00236 #define VZ_PROFILE_SCOPE_LINE2(name, line) constexpr auto fixedName##line = \
InstrumentorUtils::CleanupOutputString(name, "__cdecl "); \
00237 ::Vesper::InstrumentationTimer \
timer##line(fixedName##line.Data)
```

11.81 Instrumentor.h

[Go to the documentation of this file.](#)

```
00001
00002 /// @file Instrumentor.h
00003 /// @author TheCherno
00004 /// @author Damon S. Green II
00005 /// @brief Provides instrumentation and profiling utilities.
00006
00007 //Copyright TheCherno
00008 //Modifications Copyright 2025 Damon S.Green II(nomad_ii_damon)
00009 //
00010 //Licensed under the Apache License, Version 2.0 (the "License");
00011 //you may not use this file except in compliance with the License.
00012 //You may obtain a copy of the License at
00013 //
00014 // [Apache] (http://www.apache.org/licenses/LICENSE-2.0)
00015
```

```

00016 #pragma once
00017
00018 #include "Vesper/Core/Log.h"
00019
00020 #include <algorithm>
00021 #include <chrono>
00022 #include <fstream>
00023 #include <iomanip>
00024 #include <string>
00025 #include <thread>
00026 #include <mutex>
00027 #include <sstream>
00028
00029 namespace Vesper {
00030
00031     using FloatingPointMicroseconds = std::chrono::duration<double, std::micro>;
00032
00033
00034     struct ProfileResult
00035     {
00036         std::string Name;
00037         long long Start, End;
00038         uint32_t ThreadID;
00039     };
00040
00041     struct InstrumentationSession
00042     {
00043         std::string Name;
00044     };
00045
00046     class Instrumentor
00047     {
00048     private:
00049         InstrumentationSession* m_CurrentSession;
00050         std::ofstream m_OutputStream;
00051         std::mutex m_Mutex;
00052     public:
00053         Instrumentor()
00054             : m_CurrentSession(nullptr)
00055         {
00056         }
00057
00058         void BeginSession(const std::string& name, const std::string& filepath = "results.json")
00059         {
00060             std::lock_guard lock(m_Mutex);
00061             if (m_CurrentSession) {
00062                 // If there is already a current session, end it and start new one
00063                 if (Log::GetCoreLogger())
00064                 {
00065                     VZ_CORE_ERROR("Instrumentor::BeginSession('{0}') when session '{1}' already
open.", name, m_CurrentSession->Name);
00066                     }
00067                     InternalEndSession();
00068                 }
00069                 m_OutputStream.open(filepath);
00070                 if (!m_OutputStream.is_open()) {
00071                     m_CurrentSession = new InstrumentationSession{ name };
00072                     WriteHeader();
00073                 }
00074                 else {
00075                     if (Log::GetCoreLogger())
00076                     {
00077                         VZ_CORE_ERROR("Instrumentor could not open results file '{0}'.", filepath);
00078                     }
00079                 }
00080                 WriteHeader();
00081                 m_CurrentSession = new InstrumentationSession{ name };
00082             }
00083
00084         void EndSession()
00085         {
00086             std::lock_guard lock(m_Mutex);
00087             InternalEndSession();
00088         }
00089
00090         void WriteProfile(const ProfileResult& result)
00091         {
00092             m_OutputStream << ",";
00093
00094             std::string name = result.Name;
00095             std::replace(name.begin(), name.end(), '\"', '\\\"');
00096
00097             m_OutputStream << "{";
00098             m_OutputStream << "\"cat\": \"function\",";
00099             m_OutputStream << "\"dur\": " << (result.End - result.Start) << ",";
00100             m_OutputStream << "\"name\": \"";
00101             m_OutputStream << name << "\"";
00102             m_OutputStream << "\"ph\": \"X\",";
00103

```

```

00102         m_OutputStream << "\"pid\":0,";
00103         m_OutputStream << "\"tid\":\"" << result.ThreadID << ",";
00104         m_OutputStream << "\"ts\":\"" << result.Start;
00105         m_OutputStream << "}";
00106
00107         std::lock_guard lock(m_Mutex);
00108         if (m_CurrentSession)
00109         {
00110             //m_OutputStream << json.str();
00111             m_OutputStream.flush();
00112         }
00113     }
00114
00115     void WriteHeader()
00116     {
00117         m_OutputStream << "{\"otherData\": {},\"traceEvents\":[";
00118         m_OutputStream.flush();
00119     }
00120
00121     void WriteFooter()
00122     {
00123         m_OutputStream << "]}";
00124         m_OutputStream.flush();
00125     }
00126
00127     void InternalEndSession()
00128     {
00129         if (m_CurrentSession)
00130         {
00131             WriteFooter();
00132             m_OutputStream.close();
00133             delete m_CurrentSession;
00134             m_CurrentSession = nullptr;
00135         }
00136     }
00137
00138     static Instrumentor& Get()
00139     {
00140         static Instrumentor instance;
00141         return instance;
00142     }
00143 };
00144
00145 class InstrumentationTimer
00146 {
00147 public:
00148     InstrumentationTimer(const char* name)
00149         : m_Name(name), m_Stopped(false)
00150     {
00151         m_StartTimepoint = std::chrono::high_resolution_clock::now();
00152     }
00153
00154     ~InstrumentationTimer()
00155     {
00156         if (!m_Stopped)
00157             Stop();
00158     }
00159
00160     void Stop()
00161     {
00162         auto endTimepoint = std::chrono::high_resolution_clock::now();
00163
00164         long long start =
00165             std::chrono::time_point_cast<std::chrono::microseconds>(m_StartTimepoint).time_since_epoch().count();
00166         long long end =
00167             std::chrono::time_point_cast<std::chrono::microseconds>(endTimepoint).time_since_epoch().count();
00168
00169         uint32_t threadID = std::hash<std::thread::id>{}(std::this_thread::get_id());
00170         Instrumentor::Get().WriteProfile({ m_Name, start, end, threadID });
00171         m_Stopped = true;
00172     }
00173 private:
00174     const char* m_Name;
00175     std::chrono::time_point<std::chrono::high_resolution_clock> m_StartTimepoint;
00176     bool m_Stopped;
00177 };
00178
00179 }
00180
00181 namespace InstrumentorUtils {
00182
00183     template <size_t N>
00184     struct ChangeResult
00185     {
00186         char Data[N];

```

```

00187     };
00188
00189     template <size_t N, size_t K>
00190     constexpr auto CleanupOutputString(const char(&expr) [N], const char(&remove) [K])
00191     {
00192         ChangeResult<N> result = {};
00193
00194         size_t srcIndex = 0;
00195         size_t dstIndex = 0;
00196         while (srcIndex < N)
00197         {
00198             size_t matchIndex = 0;
00199             while (matchIndex < K - 1 && srcIndex + matchIndex < N - 1 && expr[srcIndex + matchIndex]
00200             == remove[matchIndex])
00201                 matchIndex++;
00202                 if (matchIndex == K - 1)
00203                     srcIndex += matchIndex;
00204                     result.Data[dstIndex++] = expr[srcIndex] == '"' ? '\"' : expr[srcIndex];
00205                     srcIndex++;
00206             }
00207         return result;
00208     }
00209
00210
00211 #define VZ_PROFILE 1
00212 #if VZ_PROFILE
00213     // Resolve which function signature macro will be used. Note that this only
00214     // is resolved when the (pre)compiler starts, so the syntax highlighting
00215     // could mark the wrong one in your editor!
00216     #if defined(__GNUC__) || (defined(__MWERKS__) && (__MWERKS__ >= 0x3000)) || (defined(__ICC) &&
00217     (__ICC >= 600)) || defined(__ghs__)
00218         #define VZ_FUNC_SIG __PRETTY_FUNCTION__
00219     #elif defined(__DMC__) && (__DMC__ >= 0x810)
00220         #define VZ_FUNC_SIG __PRETTY_FUNCTION__
00221     #elif (defined(__FUNCSIG__) || (__MSC_VER))
00222         #define VZ_FUNC_SIG __FUNCSIG__
00223     #elif (defined(__INTEL_COMPILER) && (__INTEL_COMPILER >= 600)) || (defined(__IBMCPP__) &&
00224     (__IBMCPP__ >= 500))
00225         #define VZ_FUNC_SIG __FUNCTION__
00226     #elif defined(__BORLANDC__) && (__BORLANDC__ >= 0x550)
00227         #define VZ_FUNC_SIG __FUNC__
00228     #elif defined(__STDC_VERSION__) && (__STDC_VERSION__ >= 199901)
00229         #define VZ_FUNC_SIG __func__
00230     #elif defined(__cplusplus) && (__cplusplus >= 201103)
00231         #define VZ_FUNC_SIG __func__
00232     #else
00233         #define VZ_FUNC_SIG "VZ_FUNC_SIG unknown!"
00234     #endif
00235
00236     #define VZ_PROFILE_BEGIN_SESSION(name, filepath) ::Vesper::Instrumentor::Get().BeginSession(name,
00237     filepath)
00238     #define VZ_PROFILE_END_SESSION() ::Vesper::Instrumentor::Get().EndSession()
00239     #define VZ_PROFILE_SCOPE_LINE2(name, line) constexpr auto fixedName##line =
00240         InstrumentorUtils::CleanupOutputString(name, "__cdecl ");
00241     ::Vesper::InstrumentationTimer
00242         timer##line(fixedName##line.Data)
00243     #define VZ_PROFILE_SCOPE_LINE(name, line) VZ_PROFILE_SCOPE_LINE2(name, line)
00244     #define VZ_PROFILE_SCOPE(name) VZ_PROFILE_SCOPE_LINE(name, __LINE__)
00245     #define VZ_PROFILE_FUNCTION() VZ_PROFILE_SCOPE(VZ_FUNC_SIG)
00246 #endif

```

11.82 Vesper/src/Vesper/Events/ApplicationEvent.h File Reference

Defines application event classes.

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
```

Classes

- class `Vesper::WindowResizeEvent`
Event for registering window resize.
- class `Vesper::WindowCloseEvent`
Event for registering window close.
- class `Vesper::AppTickEvent`
Event for registering application tick.
- class `Vesper::AppUpdateEvent`
Event for registering application update.
- class `Vesper::AppRenderEvent`
Event for registering application render.

Namespaces

- namespace `Vesper`
TEMPORARY.

11.82.1 Detailed Description

Defines application event classes.

Author

Damon S. Green II

11.83 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 /// @file ApplicationEvent.h
00008 /// @author Damon S. Green II
00009 /// @brief Defines application event classes.
00010
00011 namespace Vesper {
00012
00013     /// @class WindowResizeEvent
00014     /// @brief Event for registering window resize.
00015     class WindowResizeEvent : public Event
00016     {
00017         public:
00018
00019             /// @brief Construct a WindowResizeEvent with the specified width and height.
00020             ///
00021             /// @param width The new width of the window.
00022             /// @param height The new height of the window.
00023             WindowResizeEvent(unsigned int width, unsigned int height)
00024                 : m_Width(width), m_Height(height) {}
00025
00026             /// @brief Get the new width of the window.
00027             unsigned int GetWidth() const { return m_Width; }
00028             /// @brief Get the new height of the window.
00029             unsigned int GetHeight() const { return m_Height; }
00030
00031             /// @brief Convert the event to a string representation.
00032             std::string ToString() const override
00033         {
```

```

00034         std::stringstream ss;
00035         ss << "WindowResizeEvent: " << m_Width << ", " << m_Height;
00036         return ss.str();
00037     }
00038
00039     EVENT_CLASS_TYPE(WindowResize)
00040     EVENT_CLASS_CATEGORY(EventCategoryApplication)
00041
00042     private:
00043         unsigned int m_Width, m_Height;
00044     };
00045
00046     /// @class WindowCloseEvent
00047     /// @brief Event for registering window close.
00048     class WindowCloseEvent : public Event
00049     {
00050     public:
00051         WindowCloseEvent() {}
00052
00053         EVENT_CLASS_TYPE(WindowClose)
00054         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00055     };
00056
00057     /// @class AppTickEvent
00058     /// @brief Event for registering application tick.
00059     class AppTickEvent : public Event
00060     {
00061     public:
00062         AppTickEvent() {}
00063
00064         EVENT_CLASS_TYPE(AppTick)
00065         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00066     };
00067
00068     /// @class AppUpdateEvent
00069     /// @brief Event for registering application update.
00070     class AppUpdateEvent : public Event
00071     {
00072     public:
00073         AppUpdateEvent() {}
00074
00075         EVENT_CLASS_TYPE(AppUpdate)
00076         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00077     };
00078
00079     /// @class AppRenderEvent
00080     /// @brief Event for registering application render.
00081     class AppRenderEvent : public Event
00082     {
00083     public:
00084         AppRenderEvent() {}
00085
00086         EVENT_CLASS_TYPE(AppRender)
00087         EVENT_CLASS_CATEGORY(EventCategoryApplication)
00088     };
00089 }
```

11.84 Vesper/src/Vesper/Events/Event.h File Reference

Defines the base Event class and related enumerations and macros.

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
```

Classes

- class **Vesper::Event**
Abstract base class for all events.
- class **Vesper::EventDispatcher**
Stack-based templated event dispatcher.

Namespaces

- namespace **Vesper**

TEMPORARY.

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**

}

Enumeration of event types.
- 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) }

Enumeration of event categories.

Functions

- std::string **Vesper::format_as** (const **Event** &e)

Format an event as a string.

11.84.1 Detailed Description

Defines the base Event class and related enumerations and macros.

Author

Damon S. Green II

11.85 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 /// @file Event.h
00007 /// @author Damon S. Green II
00008 /// @brief Defines the base Event class and related enumerations and macros.
00009
00010 namespace Vesper
00011 {
00012     /// @brief Enumeration of event types.
00013     enum class EventType
00014     {
00015         None = 0,
00016         WindowClose, WindowResize, WindowFocus, WindowLostFocus, WindowMoved,
00017         AppTick, AppUpdate, AppRender,
00018         KeyPressed, KeyReleased, KeyTyped,
00019         MouseButtonPressed, MouseButtonReleased, MouseMoved, MouseScrolled
00020     };
00021 }
```

```

00022     /// @brief Enumeration of event categories.
00023     enum EventCategory
00024     {
00025         None = 0,
00026         EventCategoryApplication = BIT(0),
00027         EventCategoryInput = BIT(1),
00028         EventCategoryKeyboard = BIT(2),
00029         EventCategoryMouse = BIT(3),
00030         EventCategoryMouseButton = BIT(4)
00031     };
00032
00033
00034     /// @class Event
00035     /// @brief Abstract base class for all events.
00036     class Event
00037     {
00038         friend class EventDispatcher;
00039     public:
00040         virtual ~Event() = default;
00041
00042         /// @brief Get the type of the event.
00043         virtual EventType GetEventType() const = 0;
00044
00045         /// @brief Get the name of the event.
00046         virtual const char* GetName() const = 0;
00047
00048         /// @brief Get the category flags of the event.
00049         virtual int GetCategoryFlags() const = 0;
00050
00051         /// @brief Convert the event to a string representation.
00052         virtual std::string ToString() const { return GetName(); }
00053
00054         /// @brief Check if the event is in a specific category.
00055         inline bool IsInCategory(EventCategory category)
00056         {
00057             return GetCategoryFlags() & category;
00058         }
00059
00060     public:
00061         /// @brief Indicates whether the event has been handled.
00062         bool Handled = false;
00063     };
00064
00065     /// @class EventDispatcher
00066     /// @brief Stack-based templated event dispatcher.
00067     class EventDispatcher
00068     {
00069         /// @brief Type alias for event handling functions.
00070         template<typename T>
00071         using EventFn = std::function<bool(T&)>;
00072
00073     public:
00074
00075         /// @brief Construct an EventDispatcher for a specific event.
00076         ///
00077         /// @param event The event to dispatch.
00078         EventDispatcher(Event& event)
00079             : m_Event(event)
00080         {
00081         }
00082
00083         /// @brief Dispatch the event to the appropriate handler if the types match.
00084         ///
00085         /// @tparam T The type of the event to dispatch.
00086         /// @param func The function to handle the event.
00087         /// @return True if the event was handled, false otherwise.
00088         template<typename T>
00089         bool Dispatch(EventFn<T> func)
00090         {
00091             if (m_Event.GetEventType() == T::GetStaticType())
00092             {
00093                 m_Event.Handled = func(*((T*)&m_Event));
00094                 return true;
00095             }
00096             return false;
00097         }
00098     private:
00099         /// @brief The event to be dispatched.
00100         Event& m_Event;
00101     };
00102
00103     /// @brief Format an event as a string.
00104     inline std::string format_as(const Event& e)
00105     {
00106         return e.ToString();
00107     }
00108 }
```

11.86 Vesper/src/Vesper/Events/KeyEvent.h File Reference

Defines keyboard event classes.

```
#include "vzpch.h"
#include "Vesper/Core/Base.h"
#include "Vesper/Events/Event.h"
```

Classes

- class [Vesper::KeyEvent](#)
Base class for keyboard events.
- class [Vesper::KeyPressedEvent](#)
Event for registering key press.
- class [Vesper::KeyReleasedEvent](#)
Event for registering key release.
- class [Vesper::KeyTypedEvent](#)
Event for registering key typing.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.86.1 Detailed Description

Defines keyboard event classes.

Author

Damon S. Green II

11.87 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 /// @file KeyEvent.h
00008 /// @author Damon S. Green II
00009 /// @brief Defines keyboard event classes.
00010
00011 namespace Vesper {
00012
00013     /// @class KeyEvent
00014     /// @brief Base class for keyboard events.
00015     class KeyEvent : public Event
00016     {
00017         public:
00018             /// @brief Get the key code associated with the event.
```

```

00020     inline int GetKeyCode() const { return m_KeyCode; }
00021
00022     EVENT_CLASS_CATEGORY(EventCategoryKeyboard | EventCategoryInput)
00023 protected:
00024
00025     /// @brief Construct a KeyEvent with the specified key code.
00026     ///
00027     /// @param keycode The key code associated with the event.
00028     /// Restricted specific construction to derived classes.
00029     KeyEvent(int keycode)
00030         : m_KeyCode(keycode) {}
00031     int m_KeyCode;
00032 };
00033
00034 /// @class KeyPressedEvent
00035 /// @brief Event for registering key press.
00036 class KeyPressedEvent : public KeyEvent
00037 {
00038 public:
00039
00040     /// @brief Construct a KeyPressedEvent with the specified key code and repeat count.
00041     ///
00042     /// @param keycode The key code associated with the event.
00043     /// @param repeatCount The number of times the key press is repeated.
00044     KeyPressedEvent(int keycode, int repeatCount)
00045         : KeyEvent(keycode), m_RepeatCount(repeatCount) {}
00046
00047     /// @brief Get the repeat count of the key press event.
00048     inline int GetRepeatCount() const { return m_RepeatCount; }
00049
00050     /// @brief Convert the event to a string representation.
00051     std::string ToString() const override
00052     {
00053         std::stringstream ss;
00054         ss << "KeyPressedEvent: " << m_KeyCode << " (" << m_RepeatCount << " repeats)";
00055         return ss.str();
00056     }
00057
00058     EVENT_CLASS_TYPE(KeyPressed)
00059 private:
00060     int m_RepeatCount;
00061 };
00062
00063 /// @class KeyReleasedEvent
00064 /// @brief Event for registering key release.
00065 class KeyReleasedEvent : public KeyEvent
00066 {
00067 public:
00068     /// @brief Construct a KeyReleasedEvent with the specified key code.
00069     ///
00070     /// @param keycode The key code associated with the event.
00071     KeyReleasedEvent(int keycode)
00072         : KeyEvent(keycode) {}
00073
00074     /// @brief Convert the event to a string representation.
00075     std::string ToString() const override
00076     {
00077         std::stringstream ss;
00078         ss << "KeyReleasedEvent: " << m_KeyCode;
00079         return ss.str();
00080     }
00081
00082     EVENT_CLASS_TYPE(KeyReleased)
00083 };
00084
00085 /// @class KeyTypedEvent
00086 /// @brief Event for registering key typing.
00087 class KeyTypedEvent : public KeyEvent
00088 {
00089 public:
00090     /// @brief Construct a KeyTypedEvent with the specified key code.
00091     ///
00092     /// @param keycode The key code associated with the event.
00093     KeyTypedEvent(int keycode)
00094         : KeyEvent(keycode) {}
00095
00096
00097     /// @brief Convert the event to a string representation.
00098     std::string ToString() const override
00099     {
00100         std::stringstream ss;
00101         ss << "KeyTypedEvent: " << m_KeyCode;
00102         return ss.str();
00103     }
00104
00105     EVENT_CLASS_TYPE(KeyTyped)
00106 };

```

```
00107  
00108 }
```

11.88 Vesper/src/Vesper/Events/MouseEvent.h File Reference

Defines mouse event classes.

```
#include "vzpch.h"  
#include "Vesper/Core/Base.h"  
#include "Vesper/Events/Event.h"
```

Classes

- class [Vesper::MouseMovedEvent](#)
Event for registering mouse movement.
- class [Vesper::MouseScrolledEvent](#)
Event for registering mouse scroll wheel movement.
- class [Vesper::MouseButtonEvent](#)
Base class for mouse button events.
- class [Vesper::MouseButtonPressedEvent](#)
Event for registering mouse button presses.
- class [Vesper::MouseButtonReleasedEvent](#)
Event for registering mouse button releases.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.88.1 Detailed Description

Defines mouse event classes.

Author

Damon S. Green II

11.89 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 /// @file MouseEvent.h
00008 /// @author Damon S. Green II
00009 /// @brief Defines mouse event classes.
00010
00011 namespace Vesper {
00012
00013     /// @class MouseMovedEvent
00014     /// @brief Event for registering mouse movement.
00015     class MouseMovedEvent : public Event
00016     {
00017         public:
00018             /// @brief Construct a MouseMovedEvent with the specified x and y coordinates.
00019             ///
00020             /// @param x The x coordinate of the mouse.
00021             /// @param y The y coordinate of the mouse.
00022             MouseMovedEvent(float x, float y)
00023                 : m_MouseX(x), m_MouseY(y) {
00024
00025         }
00026
00027             /// @brief Get the x coordinate of the mouse.
00028             inline float GetX() const { return m_MouseX; }
00029             /// @brief Get the y coordinate of the mouse.
00030             inline float GetY() const { return m_MouseY; }
00031
00032             /// @brief Convert the event to a string representation.
00033             std::string ToString() const override
00034         {
00035             std::stringstream ss;
00036             ss << "MouseMovedEvent: " << m_MouseX << ", " << m_MouseY;
00037             return ss.str();
00038         }
00039
00040             EVENT_CLASS_TYPE(MouseMoved)
00041             EVENT_CLASS_CATEGORY(EventCategoryMouse | EventCategoryInput)
00042     private:
00043         float m_MouseX, m_MouseY;
00044     };
00045
00046     /// @class MouseScrolledEvent
00047     /// @brief Event for registering mouse scroll wheel movement.
00048     class MouseScrolledEvent : public Event
00049     {
00050         public:
00051             /// @brief Construct a MouseScrolledEvent with the specified x and y offsets.
00052             ///
00053             /// @param xOffset The offset of the mouse scroll in the x direction.
00054             /// @param yOffset The offset of the mouse scroll in the y direction.
00055             MouseScrolledEvent(float xOffset, float yOffset)
00056                 : m_XOffset(xOffset), m_YOffset(yOffset) {
00057
00058             /// @brief Get the x offset of the mouse scroll.
00059             inline float GetXOffset() const { return m_XOffset; }
00060             /// @brief Get the y offset of the mouse scroll.
00061             inline float GetYOffset() const { return m_YOffset; }
00062
00063             /// @brief Convert the event to a string representation.
00064             std::string ToString() const override
00065         {
00066             std::stringstream ss;
00067             ss << "MouseScrolledEvent: " << GetXOffset() << ", " << GetYOffset();
00068             return ss.str();
00069         }
00070
00071             EVENT_CLASS_TYPE(MouseScrolled)
00072             EVENT_CLASS_CATEGORY(EventCategoryMouse | EventCategoryInput)
00073     private:
00074         float m_XOffset, m_YOffset;
00075     };
00076
00077     /// @class MouseButtonEvent
00078     /// @brief Base class for mouse button events.
00079     class MouseButtonEvent : public Event
00080     {
00081         public:
00082

```

```

00083     /// @brief Get the mouse button associated with the event.
00084     inline int GetMouseButton() const { return m_Button; }
00085
00086     EVENT_CLASS_CATEGORY(EventCategoryMouse | EventCategoryInput)
00087 protected:
00088     /// @brief Construct a MouseButtonEvent with the specified button.
00089     ///
00090     /// @param button The mouse button associated with the event.
00091     /// Restricted specific construction to derived classes.
00092     MouseButtonEvent(int button)
00093         : m_Button(button) {
00094     }
00095     int m_Button;
00096 };
00097
00098 /// @class MouseButtonPressedEvent
00099 /// @brief Event for registering mouse button presses.
00100 class MouseButtonPressedEvent : public MouseButtonEvent
00101 {
00102 public:
00103     /// @brief Construct a MouseButtonPressedEvent with the specified button.
00104     ///
00105     /// @param button The mouse button associated with the event.
00106     MouseButtonPressedEvent(int button)
00107         : MouseButtonEvent(button) {
00108     }
00109
00110     /// @brief Convert the event to a string representation.
00111     std::string ToString() const override
00112     {
00113         std::stringstream ss;
00114         ss << "MouseButtonPressedEvent: " << GetMouseButton();
00115         return ss.str();
00116     }
00117
00118     EVENT_CLASS_TYPE(MouseButtonPressed)
00119 };
00120
00121 /// @class MouseButtonReleasedEvent
00122 /// @brief Event for registering mouse button releases.
00123 class MouseButtonReleasedEvent : public MouseButtonEvent
00124 {
00125 public:
00126
00127     /// @brief Construct a MouseButtonReleasedEvent with the specified button.
00128     ///
00129     /// @param button The mouse button associated with the event.
00130     MouseButtonReleasedEvent(int button)
00131         : MouseButtonEvent(button) {
00132     }
00133
00134     /// @brief Convert the event to a string representation.
00135     std::string ToString() const override
00136     {
00137         std::stringstream ss;
00138         ss << "MouseButtonReleasedEvent: " << GetMouseButton();
00139         return ss.str();
00140     }
00141
00142     EVENT_CLASS_TYPE(MouseButtonReleased)
00143 };
00144
00145 }

```

11.90 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**

11.90.1 Macro Definition Documentation

11.90.1.1 `IMGUI_IMPL_OPENGL_LOADER_GLAD`

```
#define IMGUI_IMPL_OPENGL_LOADER_GLAD
```

11.91 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.

11.92 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.

11.93 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 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 }
```

11.94 Vesper/src/Vesper/ImGui/VesperImGui.h File Reference

```
#include "imgui/imgui.h"
```

Namespaces

- namespace [Vesper](#)
TEMPORARY.

Functions

- static void [Vesper::DisplayVesperInfo_ImGui \(\)](#)

11.95 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 }
```

11.96 Vesper/src/Vesper/ImGui/ImGuiBuild.cpp File Reference

```
#include "vzpch.h"
```

11.97 Vesper/src/Vesper/Input/Input.h File Reference

```
#include "Vesper/Core/Base.h"
#include <glm/glm.hpp>
```

Classes

- class [Vesper::Input](#)

Base input class for querying input states.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.98 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 Input
0010     /// @brief Base input class for querying input states.
0011     class Input
0012     {
0013     protected:
0014         Input() = default;
0015     public:
0016         Input(const Input&) = delete;
0017         Input& operator=(const Input&) = delete;
0018
0019         /// @brief Checks if the specified key is currently pressed.
0020         ///
0021         /// @param keycode The keycode of the key to check.
0022         /// @return True if the key is pressed, false otherwise.
0023         static bool IsKeyPressed(int keycode);
0024
0025         /// @brief Checks if the specified mouse button is currently pressed.
0026         ///
0027         /// @param button The mouse button to check.
0028         /// @return True if the mouse button is pressed, false otherwise.
0029         static bool IsMouseButtonPressed(int button);
0030
0031         /// @brief Gets the current X position of the mouse cursor.
0032         static float GetMouseX();
0033         /// @brief Gets the current Y position of the mouse cursor.
0034         static float GetMouseY();
0035         /// @brief Gets the current position of the mouse cursor as a 2D vector.
0036         static glm::vec2 GetMousePosition();
0037     };
0038 }
```

11.99 Vesper/src/Vesper/Input/KeyCodes.h File Reference

Namespaces

- namespace [Vesper](#)

TEMPORARY.

- namespace [Vesper::Key](#)

Namespace for keyboard key codes.

Typedefs

- using `Vesper::KeyCode` = `uint16_t`
Alias for keyboard key code type.

Enumerations

- enum : `KeyCode` {

`Vesper::Key::Space` = 32 , `Vesper::Key::Apostrophe` = 39 , `Vesper::Key::Comma` = 44 , `Vesper::Key::Minus` = 45 ,
 `Vesper::Key::Period` = 46 , `Vesper::Key::Slash` = 47 , `Vesper::Key::D0` = 48 , `Vesper::Key::D1` = 49 ,
 `Vesper::Key::D2` = 50 , `Vesper::Key::D3` = 51 , `Vesper::Key::D4` = 52 , `Vesper::Key::D5` = 53 ,
 `Vesper::Key::D6` = 54 , `Vesper::Key::D7` = 55 , `Vesper::Key::D8` = 56 , `Vesper::Key::D9` = 57 ,
 `Vesper::Key::Semicolon` = 59 , `Vesper::Key::Equal` = 61 , `Vesper::Key::A` = 65 , `Vesper::Key::B` = 66 ,
 `Vesper::Key::C` = 67 , `Vesper::Key::D` = 68 , `Vesper::Key::E` = 69 , `Vesper::Key::F` = 70 ,
 `Vesper::Key::G` = 71 , `Vesper::Key::H` = 72 , `Vesper::Key::I` = 73 , `Vesper::Key::J` = 74 ,
 `Vesper::Key::K` = 75 , `Vesper::Key::L` = 76 , `Vesper::Key::M` = 77 , `Vesper::Key::N` = 78 ,
 `Vesper::Key::O` = 79 , `Vesper::Key::P` = 80 , `Vesper::Key::Q` = 81 , `Vesper::Key::R` = 82 ,
 `Vesper::Key::S` = 83 , `Vesper::Key::T` = 84 , `Vesper::Key::U` = 85 , `Vesper::Key::V` = 86 ,
 `Vesper::Key::W` = 87 , `Vesper::Key::X` = 88 , `Vesper::Key::Y` = 89 , `Vesper::Key::Z` = 90 ,
 `Vesper::Key::LeftBracket` = 91 , `Vesper::Key::Backslash` = 92 , `Vesper::Key::RightBracket` = 93 ,
 `Vesper::Key::GraveAccent` = 96 ,
 `Vesper::Key::World1` = 161 , `Vesper::Key::World2` = 162 , `Vesper::Key::Escape` = 256 , `Vesper::Key::Enter` = 257 ,
 `Vesper::Key::Tab` = 258 , `Vesper::Key::Backspace` = 259 , `Vesper::Key::Insert` = 260 , `Vesper::Key::Delete` = 261 ,
 `Vesper::Key::Right` = 262 , `Vesper::Key::Left` = 263 , `Vesper::Key::Down` = 264 , `Vesper::Key::Up` = 265 ,
 `Vesper::Key::PageUp` = 266 , `Vesper::Key::PageDown` = 267 , `Vesper::Key::Home` = 268 , `Vesper::Key::End` = 269 ,
 `Vesper::Key::CapsLock` = 280 , `Vesper::Key::ScrollLock` = 281 , `Vesper::Key::NumLock` = 282 ,
 `Vesper::Key::PrintScreen` = 283 ,
 `Vesper::Key::Pause` = 284 , `Vesper::Key::F1` = 290 , `Vesper::Key::F2` = 291 , `Vesper::Key::F3` = 292 ,
 `Vesper::Key::F4` = 293 , `Vesper::Key::F5` = 294 , `Vesper::Key::F6` = 295 , `Vesper::Key::F7` = 296 ,
 `Vesper::Key::F8` = 297 , `Vesper::Key::F9` = 298 , `Vesper::Key::F10` = 299 , `Vesper::Key::F11` = 300 ,
 `Vesper::Key::F12` = 301 , `Vesper::Key::F13` = 302 , `Vesper::Key::F14` = 303 , `Vesper::Key::F15` = 304 ,
 `Vesper::Key::F16` = 305 , `Vesper::Key::F17` = 306 , `Vesper::Key::F18` = 307 , `Vesper::Key::F19` = 308 ,
 `Vesper::Key::F20` = 309 , `Vesper::Key::F21` = 310 , `Vesper::Key::F22` = 311 , `Vesper::Key::F23` = 312 ,
 `Vesper::Key::F24` = 313 , `Vesper::Key::F25` = 314 , `Vesper::Key::KP_0` = 320 , `Vesper::Key::KP_1` = 321 ,
 `Vesper::Key::KP_2` = 322 , `Vesper::Key::KP_3` = 323 , `Vesper::Key::KP_4` = 324 , `Vesper::Key::KP_5` = 325 ,
 `Vesper::Key::KP_6` = 326 , `Vesper::Key::KP_7` = 327 , `Vesper::Key::KP_8` = 328 , `Vesper::Key::KP_9` = 329 ,
 `Vesper::Key::KP_DECIMAL` = 330 , `Vesper::Key::KP_DIVIDE` = 331 , `Vesper::Key::KP_MULTIPLY` = 332 ,
 `Vesper::Key::KP_SUBTRACT` = 333 ,
 `Vesper::Key::KP_ADD` = 334 , `Vesper::Key::KP_ENTER` = 335 , `Vesper::Key::KP_EQUAL` = 336 ,
 `Vesper::Key::LeftShift` = 340 ,
 `Vesper::Key::LeftControl` = 341 , `Vesper::Key::LeftAlt` = 342 , `Vesper::Key::LeftSuper` = 343 , `Vesper::Key::RightShift` = 344 ,
 `Vesper::Key::RightControl` = 345 , `Vesper::Key::RightAlt` = 346 , `Vesper::Key::RightSuper` = 347 ,
 `Vesper::Key::Menu` = 348 }

Enumeration of keyboard key codes.

11.100 KeyCodes.h

Go to the documentation of this file.

```
00001 #pragma once
```

```
00002
00003 namespace Vesper {
00004
00005     /// @brief Alias for keyboard key code type.
00006     using KeyCode = uint16_t;
00007
00008     /// @namespace Vesper::Key
00009     /// @brief Namespace for keyboard key codes.
00010     namespace Key {
00011
00012         /// @brief Enumeration of keyboard key codes.
00013         enum : KeyCode
00014         {
00015             // From glfw3.h
00016
00017             Space = 32,
00018             Apostrophe = 39, /* ' */
00019             Comma = 44, /* , */
00020             Minus = 45, /* - */
00021             Period = 46, /* . */
00022             Slash = 47, /* / */
00023
00024             D0 = 48,
00025             D1 = 49,
00026             D2 = 50,
00027             D3 = 51,
00028             D4 = 52,
00029             D5 = 53,
00030             D6 = 54,
00031             D7 = 55,
00032             D8 = 56,
00033             D9 = 57,
00034
00035             Semicolon = 59, /* ; */
00036             Equal = 61, /* = */
00037
00038             A = 65,
00039             B = 66,
00040             C = 67,
00041             D = 68,
00042             E = 69,
00043             F = 70,
00044             G = 71,
00045             H = 72,
00046             I = 73,
00047             J = 74,
00048             K = 75,
00049             L = 76,
00050             M = 77,
00051             N = 78,
00052             O = 79,
00053             P = 80,
00054             Q = 81,
00055             R = 82,
00056             S = 83,
00057             T = 84,
00058             U = 85,
00059             V = 86,
00060             W = 87,
00061             X = 88,
00062             Y = 89,
00063             Z = 90,
00064
00065             LeftBracket = 91, /* [ */
00066             Backslash = 92, /* \ */
00067             RightBracket = 93, /* ] */
00068             GraveAccent = 96, /* ` */
00069
00070             World1 = 161, /* non-US #1 */
00071             World2 = 162, /* non-US #2 */
00072
00073             /* Function keys */
00074             Escape = 256,
00075             Enter = 257,
00076             Tab = 258,
00077             Backspace = 259,
00078             Insert = 260,
00079             Delete = 261,
00080             Right = 262,
00081             Left = 263,
00082             Down = 264,
00083             Up = 265,
00084             PageUp = 266,
00085             PageDown = 267,
00086             Home = 268,
00087             End = 269,
00088             CapsLock = 280,
```

```

00089     ScrollLock = 281,
00090     NumLock = 282,
00091     PrintScreen = 283,
00092     Pause = 284,
00093     F1 = 290,
00094     F2 = 291,
00095     F3 = 292,
00096     F4 = 293,
00097     F5 = 294,
00098     F6 = 295,
00099     F7 = 296,
00100     F8 = 297,
00101     F9 = 298,
00102     F10 = 299,
00103     F11 = 300,
00104     F12 = 301,
00105     F13 = 302,
00106     F14 = 303,
00107     F15 = 304,
00108     F16 = 305,
00109     F17 = 306,
00110     F18 = 307,
00111     F19 = 308,
00112     F20 = 309,
00113     F21 = 310,
00114     F22 = 311,
00115     F23 = 312,
00116     F24 = 313,
00117     F25 = 314,
00118
00119     /* Keypad */
00120     KP_0 = 320,
00121     KP_1 = 321,
00122     KP_2 = 322,
00123     KP_3 = 323,
00124     KP_4 = 324,
00125     KP_5 = 325,
00126     KP_6 = 326,
00127     KP_7 = 327,
00128     KP_8 = 328,
00129     KP_9 = 329,
00130     KP_DECIMAL = 330,
00131     KP_DIVIDE = 331,
00132     KP_MULTIPLY = 332,
00133     KP_SUBTRACT = 333,
00134     KP_ADD = 334,
00135     KP_ENTER = 335,
00136     KP_EQUAL = 336,
00137
00138
00139     LeftShift = 340,
00140     LeftControl = 341,
00141     LeftAlt = 342,
00142     LeftSuper = 343,
00143     RightShift = 344,
00144     RightControl = 345,
00145     RightAlt = 346,
00146     RightSuper = 347,
00147     Menu = 348
00148     };
00149 }
00150 }
```

11.101 Vesper/src/Vesper/Input/MouseButtonCodes.h File Reference

Namespaces

- namespace [Vesper](#)
TEMPORARY.
- namespace [Vesper::Mouse](#)
Namespace for mouse button codes.

Typedefs

- using [Vesper::MouseCode](#) = [uint8_t](#)
Alias for mouse button code type.

Enumerations

- enum : MouseCode {

 Vesper::Mouse::Button0 = 0 , Vesper::Mouse::Button1 = 1 , Vesper::Mouse::Button2 = 2 , Vesper::Mouse::Button3
 = 3 ,
 Vesper::Mouse::Button4 = 4 , Vesper::Mouse::Button5 = 5 , Vesper::Mouse::Button6 = 6 , Vesper::Mouse::Button7
= 7 ,
 Vesper::Mouse::ButtonLast = Button7 , Vesper::Mouse::ButtonLeft = Button0 , Vesper::Mouse::ButtonRight
= Button1 , Vesper::Mouse::ButtonMiddle = Button2 }

Enumeration of mouse button codes.

11.102 MouseButtonCodes.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 namespace Vesper {
00004
00005     /// @brief Alias for mouse button code type.
00006     using MouseCode = uint8_t;
00007
00008     /// @namespace Vesper::Mouse
00009     /// @brief Namespace for mouse button codes.
00010     namespace Mouse {
00011
00012         /// @brief Enumeration of mouse button codes.
00013         enum : MouseCode
00014         {
00015             // From glfw3.h
00016             Button0 = 0,
00017             Button1 = 1,
00018             Button2 = 2,
00019             Button3 = 3,
00020             Button4 = 4,
00021             Button5 = 5,
00022             Button6 = 6,
00023             Button7 = 7,
00024
00025             ButtonLast = Button7,
00026             ButtonLeft = Button0,
00027             ButtonRight = Button1,
00028             ButtonMiddle = Button2,
00029         };
00030     }
00031 }
```

11.103 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`

11.103.1 Macro Definition Documentation**11.103.1.1 GLM_ENABLE_EXPERIMENTAL**

```
#define GLM_ENABLE_EXPERIMENTAL
```

11.104 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.

11.104.1 Class Documentation**11.104.1.1 struct Vesper::ParticleProps****Class Members**

| | | |
|--------------------|--|--|
| <code>vec4</code> | <code>ColorBegin = { 1.0f, 1.0f, 1.0f, 1.0f }</code> | |
| <code>vec4</code> | <code>ColorEnd = { 1.0f, 1.0f, 1.0f, 1.0f }</code> | |
| <code>float</code> | <code>LifeTime = 1.0f</code> | |
| <code>float</code> | <code>LifetimeVariation = 0.0f</code> | |
| <code>vec3</code> | <code>Position = { 0.0f, 0.0f, 0.0f }</code> | |
| <code>float</code> | <code>Rotation = 0.0f</code> | |
| <code>float</code> | <code>RotationVariation = 0.0f</code> | |
| <code>float</code> | <code>SizeBegin = 1.0f</code> | |

| | | | |
|-------|--|--|--|
| float | SizeEnd = 0.0f | | |
| float | SizeVariation = 0.0f | | |
| vec3 | Velocity = { 0.0f, 0.0f, 0.0f } | | |
| vec3 | VelocityVariation = { 0.0f, 0.0f, 0.0f } | | |

11.104.1.2 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 | | |

11.105 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 }

```

11.106 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.

11.107 Vesper/src/Vesper/Renderer/Buffer.h File Reference

Classes

- struct [Vesper::BufferElement](#)
Represents a single element in a buffer layout.
- class [Vesper::BufferLayout](#)
Represents the layout of a buffer, consisting of multiple BufferElements.
- class [Vesper::VertexBuffer](#)
Abstract base class for a vertex buffer.
- class [Vesper::IndexBuffer](#)
Abstract base class for an index buffer.

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](#) }

The different data types that can be used in shaders.

Functions

- static uint32_t **Vesper::ShaderDataTypeSize** (ShaderDataType type)
Returns the size in bytes of the given ShaderDataType.

11.108 Buffer.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 namespace Vesper {
00004
00005     /// @brief The different data types that can be used in shaders.
00006     enum class ShaderDataType {
00007         None = 0,
00008         Float, Float2, Float3, Float4,
00009         Mat3, Mat4,
00010         Int, Int2, Int3, Int4,
00011         Bool
00012     };
00013
00014     /// @brief Returns the size in bytes of the given ShaderDataType.
00015     ///
00016     /// @param type The ShaderDataType to get the size of.
00017     /// @return The size in bytes of the ShaderDataType.
00018     /// Will assert if the type is unknown.
00019     static uint32_t ShaderDataTypeSize(ShaderDataType type) {
00020         switch (type) {
00021             case ShaderDataType::Float:      return 4;
00022             case ShaderDataType::Float2:    return 4 * 2;
00023             case ShaderDataType::Float3:    return 4 * 3;
00024             case ShaderDataType::Float4:    return 4 * 4;
00025             case ShaderDataType::Mat3:     return 4 * 3 * 3;
00026             case ShaderDataType::Mat4:     return 4 * 4 * 4;
00027             case ShaderDataType::Int:      return 4;
00028             case ShaderDataType::Int2:     return 4 * 2;
00029             case ShaderDataType::Int3:     return 4 * 3;
00030             case ShaderDataType::Int4:     return 4 * 4;
00031             case ShaderDataType::Bool:     return 1;
00032         }
00033         VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00034         return 0;
00035     }
00036
00037
00038     /// @brief Represents a single element in a buffer layout.
00039     struct BufferElement {
00040         /// @brief The name of the buffer element.
00041         std::string Name;
00042         /// @brief The data type of the buffer element.
00043         ShaderDataType Type;
00044         /// @brief The size in bytes of the buffer element.
00045         uint32_t Size;
00046         /// @brief The offset in bytes of the buffer element from the start of the buffer.
00047         uint32_t Offset;
00048         /// @brief Whether the buffer element is normalized.
00049         bool Normalized;
00050
00051         BufferElement() {}
00052         /// @brief Constructs a BufferElement with the given type, name, and normalization flag.
00053         ///
00054         /// @param type The ShaderDataType of the buffer element.
00055         /// @param name The name of the buffer element.
00056         /// @param normalized Whether the buffer element is normalized.
00057         BufferElement(ShaderDataType type, const std::string& name, bool normalized = false)
00058             : Name(name), Type(type), Size(ShaderDataTypeSize(type)), Offset(0),
00059             Normalized(normalized)
00060         {}
00061
00062         /// @brief Returns the number of components in the buffer element based on its ShaderDataType.
00063         uint32_t GetComponentCount() const {
00064             switch (Type) {
00065                 case ShaderDataType::Float:      return 1;
00066                 case ShaderDataType::Float2:    return 2;
00067                 case ShaderDataType::Float3:    return 3;
00068                 case ShaderDataType::Float4:    return 4;
00069                 case ShaderDataType::Mat3:     return 3 * 3;
00070                 case ShaderDataType::Mat4:     return 4 * 4;
00071             }
00072         }
00073     };
00074 }
```

```

00071         case ShaderDataType::Int:           return 1;
00072         case ShaderDataType::Int2:          return 2;
00073         case ShaderDataType::Int3:          return 3;
00074         case ShaderDataType::Int4:          return 4;
00075         case ShaderDataType::Bool:         return 1;
00076     }
00077     VZ_CORE_ASSERT(false, "Unknown ShaderDataType!");
00078     return 0;
00079 }
00080
00081 /**
00082  * @class BufferLayout
00083  * @brief Represents the layout of a buffer, consisting of multiple BufferElements.
00084  */
00085 class BufferLayout {
00086 public:
00087     BufferLayout() {}
00088
00089     /// @brief Constructs a BufferLayout with the given list of BufferElements.
00090     /// @param elements The list of BufferElements that make up the layout.
00091     /// Automatically calculates offsets and stride.
00092     BufferLayout(const std::initializer_list<BufferElement>& elements)
00093         : m_Elements(elements), m_Stride(0)
00094     {
00095         CalculateOffsetsAndStride();
00096     }
00097
00098     /// @brief Returns the list of BufferElements in the layout.
00099     inline const std::vector<BufferElement>& GetElements() const { return m_Elements; }
00100     /// @brief Returns the stride (total size in bytes) of the buffer layout.
00101     inline uint32_t GetStride() const { return m_Stride; }
00102
00103     std::vector<BufferElement>::iterator begin() { return m_Elements.begin(); }
00104     std::vector<BufferElement>::const_iterator begin() const { return m_Elements.begin(); }
00105     std::vector<BufferElement>::iterator end() { return m_Elements.end(); }
00106     std::vector<BufferElement>::const_iterator end() const { return m_Elements.end(); }
00107
00108
00109
00110 private:
00111     /// @brief Calculates the offsets and stride for the buffer layout based on its elements.
00112     void CalculateOffsetsAndStride() {
00113         uint32_t offset = 0;
00114         m_Stride = 0;
00115         for (auto& element : m_Elements) {
00116             element.Offset = offset;
00117             offset += element.Size;
00118             m_Stride += element.Size;
00119         }
00120     }
00121
00122
00123 private:
00124     std::vector<BufferElement> m_Elements;
00125     uint32_t m_Stride = 0;
00126 };
00127
00128
00129 /**
00130  * @class VertexBuffer
00131  * @brief Abstract base class for a vertex buffer.
00132  */
00133
00134 public:
00135     virtual ~VertexBuffer() {}
00136
00137     virtual void Bind() const = 0;
00138     virtual void Unbind() const = 0;
00139
00140     virtual void SetLayout(const BufferLayout& layout) = 0;
00141     virtual const BufferLayout& GetLayout() const = 0;
00142
00143     virtual void SetData(const void* data, uint32_t size) = 0;
00144
00145     static Ref<VertexBuffer> Create(uint32_t size);
00146     static Ref<VertexBuffer> Create(float* vertices, uint32_t size);
00147 };
00148
00149 /**
00150  * @class IndexBuffer
00151  * @brief Abstract base class for an index buffer.
00152  */
00153
00154 /**
00155  * Currently only supports uint32_t indices
00156  */
00157
00158 public:
00159     virtual ~IndexBuffer() {}
00160     virtual void Bind() const = 0;

```

```

00158     virtual void Unbind() const = 0;
00159
00160     virtual uint32_t GetCount() const = 0;
00161
00162     static Ref<IndexBuffer> Create(uint32_t* indices, uint32_t count);
00163 };
00164 }
```

11.109 Vesper/src/Vesper/Renderer/Camera.h File Reference

```
#include <glm/glm.hpp>
```

Classes

- class [Vesper::Camera](#)

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.110 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 }
```

11.111 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`

11.111.1 Macro Definition Documentation

11.111.1.1 GLM_ENABLE_EXPERIMENTAL

```
#define GLM_ENABLE_EXPERIMENTAL
```

11.112 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.

11.113 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 aspectRatio, 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,
00024             m_ViewportHeight = height; UpdateProjection(); }
00025
00026         const glm::mat4& GetViewMatrix() const { return m_ViewMatrix; }
00027         const glm::mat4& GetViewProjection() const { return m_Projection * m_ViewMatrix; }
00028
00029         glm::vec3 GetUpDirection() const;
00030         glm::vec3 GetRightDirection() const;
00031         glm::vec3 GetForwardDirection() const;
00032         glm::quat GetOrientation() const;
00033
00034         const glm::vec3&GetPosition() const { return m_Position; }
00035         void SetPosition(const glm::vec3& position);
00036
00037         float GetPitch() const { return m_Pitch; }
00038         void SetPitch(float pitch) { m_Pitch = pitch; }
00039
00040         float GetYaw() const { return m_Yaw; }
00041         void SetYaw(float yaw) { m_Yaw = yaw; }
00042     private:
00043         void UpdateProjection();
00044         void UpdateView();
00045
00046         bool OnMouseScroll(MouseScrolledEvent& e);
00047
00048         void MousePan(const glm::vec2& delta);
00049         void MouseRotate(const glm::vec2& delta);
00050         void MouseZoom(float delta);
00051
00052         glm::vec3 CalculatePosition() const;
00053
00054         std::pair<float, float> PanSpeed() const;
00055         float RotationSpeed() const;
00056         float ZoomSpeed() const;
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 }
```

11.114 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.

11.115 Vesper/src/Vesper/Renderer/Framebuffer.h File Reference

```
#include "Vesper/Core/Base.h"
```

Classes

- struct [Vesper::FramebufferSpecification](#)
Specification for creating a [Framebuffer](#). [More...](#)
- class [Vesper::Framebuffer](#)
Abstract class representing a framebuffer.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.115.1 Class Documentation

11.115.1.1 struct Vesper::FramebufferSpecification

Specification for creating a [Framebuffer](#).

Class Members

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

11.116 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     /// @brief Specification for creating a Framebuffer.
00008     struct FramebufferSpecification
00009     {
00010         uint32_t Width, Height;
00011         // FramebufferFormat Format = FramebufferFormat::RGBA8;
00012         uint32_t Samples = 1;
00013
00014         bool SwapChainTarget = false;
00015     };
00016
00017
00018     /// @class Framebuffer
00019     /// @brief Abstract class representing a framebuffer.
00020     class Framebuffer
00021     {
00022     public:
00023         ~Framebuffer() = default;
00024
00025
00026         virtual void Bind() = 0;
00027         virtual void Unbind() = 0;
00028
00029         /// @brief Resizes the framebuffer to the given width and height.
00030         virtual void Resize(uint32_t width, uint32_t height) = 0;
00031
00032         /// @brief Returns the renderer ID of the color attachment texture.
00033         virtual uint32_t GetColorAttachmentRendererID() const = 0;
00034
00035         /// @brief Returns the specification used to create the framebuffer.
00036         virtual const FramebufferSpecification& GetSpecification() const = 0;
00037
00038         /// @brief Creates a framebuffer with the given specification.
00039         ///
00040         /// @param spec The specification for the framebuffer.
00041         /// @return A reference-counted pointer to the created framebuffer.
00042         static Ref<Framebuffer> Create(const FramebufferSpecification& spec);
00043     };
00044
00045 }
```

11.117 Vesper/src/Vesper/Renderer/GraphicsContext.h File Reference

```
#include "Vesper/Core/Base.h"
```

Classes

- class [Vesper::GraphicsContext](#)

Abstract class representing a graphics context.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.118 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 GraphicsContext
00008     /// @brief Abstract class representing a graphics context.
00009     class GraphicsContext
00010     {
00011     public:
00012         virtual ~GraphicsContext() {}
00013         /// @brief Initializes the graphics context.
00014         virtual void Init() = 0;
00015         /// @brief Swaps the front and back buffers.
00016         virtual void SwapBuffers() = 0;
00017
00018     };
00019
00020 }
```

11.119 Vesper/src/Vesper/Renderer/OrthographicCamera.cpp File Reference

```
#include "vzpch.h"
#include "OrthographicCamera.h"
#include <glm/gtc/matrix_transform.hpp>
```

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.120 Vesper/src/Vesper/Renderer/OrthographicCamera.h File Reference

```
#include <glm/glm.hpp>
```

Classes

- class [Vesper::OrthographicCamera](#)

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.121 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);
0010
0011         void SetProjection(float left, float right, float bottom, float top);
0012
0013         void SetPosition(const glm::vec3& position) { m_Position = position; RecalculateViewMatrix(); }
0014
0015         const glm::vec3& GetPosition() const { return m_Position; }
0016
0017         void SetRotation(float rotation) { m_Rotation = rotation; RecalculateViewMatrix(); }
0018         const float GetRotation() const { return m_Rotation; }
0019
0020         const glm::mat4& GetProjectionMatrix() const { return m_ProjectionMatrix; }
0021         const glm::mat4& GetViewMatrix() const { return m_ViewMatrix; }
0022         const glm::mat4& GetViewProjectionMatrix() const { return m_ViewProjectionMatrix; }
0023
0024     private:
0025         void RecalculateViewMatrix();
0026
0027     private:
0028         glm::mat4 m_ProjectionMatrix;
0029         glm::mat4 m_ViewMatrix;
0030         glm::mat4 m_ViewProjectionMatrix;
0031
0032         glm::vec3 m_Position = { 0.0f, 0.0f, 0.0f };
0033         float m_Rotation = 0.0f;
0034     };

```

11.122 Vesper/src/Vesper/Renderer/OrthographicCameraController.cpp File Reference

```

#include "vzpch.h"
#include "OrthographicCameraController.h"
#include "Vesper/Input/Input.h"
#include "Vesper/Input/KeyCode.h"
#include <imgui.h>

```

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.123 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.

11.124 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 }
```

11.125 Vesper/src/Vesper/Renderer/RenderCommand.cpp File Reference

```
#include "vzpch.h"
#include "RenderCommand.h"
#include "RenderAPI/OpenGL/OpenGLRendererAPI.h"
```

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.126 Vesper/src/Vesper/Renderer/RenderCommand.h File Reference

```
#include "RendererAPI.h"
```

Classes

- class [Vesper::RenderCommand](#)

A static class that provides an interface for issuing rendering commands.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.127 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     /// @brief A static class that provides an interface for issuing rendering commands.
00009     class RenderCommand
00010     {
00011     public:
00012
00013         /// @brief Initializes the rendering API.
00014         inline static void Init()
00015         {
00016             s_RendererAPI->Init();
00017         }
00018
00019         /// @brief Sets the viewport dimensions for the renderer.
00020         inline static void SetViewport(uint32_t x, uint32_t y, uint32_t width, uint32_t height)
00021         {
00022             s_RendererAPI->SetViewport(x, y, width, height);
00023         }
00024
00025         /// @brief Sets the clear color for the renderer.
00026         inline static void SetClearColor(const glm::vec4& color)
00027         {
00028             s_RendererAPI->SetClearColor(color);
00029         }
00030
00031         /// @brief Clears the rendering buffers.
00032         inline static void Clear()
00033         {
00034             s_RendererAPI->Clear();
00035         }
00036
00037         /// @brief Draws indexed geometry using the specified vertex array and index count.
00038         inline static void DrawIndexed(const Ref<VertexArray>& vertexArray, uint32_t indexCount = 0)
00039         {
00040             s_RendererAPI->DrawIndexed(vertexArray, indexCount);
00041         }
00042
00043     private:
00044         static RendererAPI* s_RendererAPI;
00045     };
00046
00047 }
```

11.128 Vesper/src/Vesper/Renderer/Renderer.cpp File Reference

```

#include "vzpch.h"
#include "Renderer.h"
#include "RenderCommand.h"
#include "Renderer2D.h"
#include "RenderAPI/OpenGL/OpenGLShader.h"
```

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.129 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](#)
The main renderer class responsible for managing rendering operations.
- struct [Vesper::Renderer::SceneData](#)
Scene data structure containing view-projection matrix. [More...](#)

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.129.1 Class Documentation

11.129.1.1 struct Vesper::Renderer::SceneData

[Scene](#) data structure containing view-projection matrix.

Class Members

| | | |
|------|--------------------------------------|--|
| mat4 | ViewProjectionMatrix | The combined view-projection matrix for the scene. |
|------|--------------------------------------|--|

11.130 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
0010     /// @brief The main renderer class responsible for managing rendering operations.
0011     class Renderer {
0012
0013         public:
0014
0015             /// @brief Initializes the renderer.
0016             /// @note This initializes both the RenderCommand and Renderer2D systems.
0017             static void Init();
0018
0019             /// @brief Handles window resize events by resizing the viewport.
0020             static void OnWindowResize(uint32_t width, uint32_t height);
0021
0022             /// @brief Begins a new scene with the given orthographic camera.
0023             /// @param camera The orthographic camera defining the view and projection for the scene.
0024             /// @todo Support for other camera types.
0025             static void BeginScene(OrthographicCamera& camera);
0026 }
```

```

00027     /// @brief Ends the current scene.
00028     /// @note Currently does nothing as scene closure is automatic, but could be useful for a
00029     render interface
00030
00031     static void EndScene();
00032
00033     /// @brief Submits a draw call with the specified shader, vertex array, and transform.
00034     ///
00035     /// @param shader The shader to use for rendering.
00036     /// @param vertexArray The vertex array to draw.
00037     /// @param transform The transformation matrix to apply.
00038     /// @todo Set up a command list to encapsulate this for draw order control
00039     static void Submit(const Ref<Shader>& shader, const Ref<VertexArray>& vertexArray, const
00040     glm::mat4& transform = glm::mat4(1.0f));
00041
00042     /// @brief Retrieves the current rendering API.
00043     inline static RendererAPI::API GetAPI() { return RendererAPI::GetAPI(); }
00044
00045     private:
00046         /// @brief Scene data structure containing view-projection matrix.
00047         struct SceneData {
00048             /// @brief The combined view-projection matrix for the scene.
00049             glm::mat4 ViewProjectionMatrix;
00050         };
00051
00052     static SceneData* s_SceneData;
00053 };

```

11.131 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](#)

11.131.1 Class Documentation

11.131.1.1 struct Vesper::QuadVertex

Class Members

| | | |
|-------|--------------|--|
| vec4 | Color | |
| vec3 | Position | |
| vec2 | TexCoord | |
| float | TexIndex | |
| float | TilingFactor | |

11.131.1.2 struct Vesper::Renderer2DData::CameraData

Class Members

| | | |
|------|----------------|--|
| mat4 | ViewProjection | |
|------|----------------|--|

11.132 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](#)
A 2D renderer for drawing quads and sprites.
- struct [Vesper::Renderer2D::Statistics](#)
2D Renderer Statistics

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.133 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     /// @class Renderer2D
00015     /// @brief A 2D renderer for drawing quads and sprites.
00016     class Renderer2D
00017     {
00018         public:
00019             /// @brief Initializes the 2D renderer.
00020             static void Init();
00021             /// @brief Shuts down the 2D renderer.
00022             static void Shutdown();
00023
00024             /// @brief Begins a new scene with the given camera and transform.
00025             ///
00026             /// @param camera The camera to use for the scene.
00027             /// @param transform The transform matrix for the camera.
00028             static void BeginScene(const Camera& camera, const glm::mat4& transform);
00029
00030             /// @brief Begins a new scene with the given editor camera.
00031             ///
00032             /// @param camera The editor camera to use for the scene.
00033             static void BeginScene(const EditorCamera& camera);
00034
00035             /// @brief Begins a new scene with the given orthographic camera.
00036             ///
00037             /// @param camera The orthographic camera to use for the scene.
00038             /// @todo Remove once we have a proper scene system
00039             static void BeginScene(const OrthographicCamera& camera);
00040
00041             /// @brief Ends the current scene.
00042             static void EndScene();
00043
00044             /// @brief Flushes the current batch of rendering commands.
00045             static void Flush();
00046
00047             /// @brief Draws a colored quad with the given transform and color.
00048             ///
00049             /// @param transform The transform matrix for the quad.
00050             /// @param color The color of the quad.
00051             static void DrawQuad(const glm::mat4& transform, const glm::vec4& color);
00052
00053             /// @brief Draws a colored quad at the specified position and size.
00054             ///
00055             /// @param position The position of the quad in 2D space.
00056             /// @param size The size of the quad.
00057             /// @param color The color of the quad.
00058             static void DrawQuad(const glm::vec2& position, const glm::vec2& size, const glm::vec4&
00059             color);
00060             /// @brief Draws a colored quad at the specified position and size.
00061             ///
00062             /// @param position The position of the quad in 3D space.
00063             /// @param size The size of the quad.
00064             /// @param color The color of the quad.
00065             static void DrawQuad(const glm::vec3& position, const glm::vec2& size, const glm::vec4&
00066             color);
00067             /// @brief Draws a textured quad with the given transform, texture, tiling factor, and tint
00068             color.
00069             ///
00070             /// @param transform The transform matrix for the quad.
00071             /// @param texture The texture to apply to the quad.
00072             /// @param tilingFactor The tiling factor for the texture.
00073             /// @param tintColor The tint color to apply to the texture.
00074             static void DrawQuadWithTexture(const glm::mat4& transform, const Ref<Texture2D>& texture,
00075             float tilingFactor, const glm::vec4 tintColor);
00076
00077             /// @brief Draws a textured quad at the specified position and size.
00078             ///
00079             /// @param position The position of the quad in 2D space.
00080             /// @param size The size of the quad.

```

```

00079     /// @param texture The texture to apply to the quad.
00080     /// @param tilingFactor The tiling factor for the texture.
00081     /// @param tintColor The tint color to apply to the texture.
00082     static void DrawQuadWithTexture(const glm::vec2& position, const glm::vec2& size, const
00083     Ref<Texture2D>& texture, float tilingFactor, const glm::vec4 tintColor);
00084     /// @brief Draws a textured quad at the specified position and size.
00085     ///
00086     /// @param position The position of the quad in 3D space.
00087     /// @param size The size of the quad.
00088     /// @param texture The texture to apply to the quad.
00089     /// @param tilingFactor The tiling factor for the texture.
00090     /// @param tintColor The tint color to apply to the texture.
00091     static void DrawQuadWithTexture(const glm::vec3& position, const glm::vec2& size, const
00092     Ref<Texture2D>& texture, float tilingFactor, const glm::vec4 tintColor);
00093     /// @brief Draws a textured quad with the given transform, subtexture, tiling factor, and tint
00094     /// color.
00095     /// @param transform The transform matrix for the quad.
00096     /// @param subtexture The subtexture to apply to the quad.
00097     /// @param tilingFactor The tiling factor for the texture.
00098     /// @param tintColor The tint color to apply to the texture.
00099     static void DrawQuadWithTexture(const glm::mat4& transform, const Ref<SubTexture2D>&
00100     subtexture, float tilingFactor, const glm::vec4 tintColor);
00101     /// @brief Draws a textured quad at the specified position and size.
00102     ///
00103     /// @param position The position of the quad in 2D space.
00104     /// @param size The size of the quad.
00105     /// @param subtexture The subtexture to apply to the quad.
00106     /// @param tilingFactor The tiling factor for the texture.
00107     /// @param tintColor The tint color to apply to the texture.
00108     static void DrawQuadWithTexture(const glm::vec2& position, const glm::vec2& size, const
00109     Ref<SubTexture2D>& subtexture, float tilingFactor, const glm::vec4 tintColor);
00110     /// @brief Draws a textured quad at the specified position and size.
00111     ///
00112     /// @param position The position of the quad in 3D space.
00113     /// @param size The size of the quad.
00114     /// @param subtexture The subtexture to apply to the quad.
00115     /// @param tilingFactor The tiling factor for the texture.
00116     /// @param tintColor The tint color to apply to the texture.
00117     static void DrawQuadWithTexture(const glm::vec3& position, const glm::vec2& size, const
00118     Ref<SubTexture2D>& subtexture, float tilingFactor, const glm::vec4 tintColor);
00119     /// @brief Draws a rotated colored quad with the given transform and color.
00120     ///
00121     /// @param transform The transform matrix for the quad.
00122     /// @param color The color of the quad.
00123     static void DrawQuadRotated(const glm::mat4& transform, const glm::vec4& color);
00124     /// @brief Draws a rotated colored quad at the specified position, size, and rotation.
00125     ///
00126     /// @param position The position of the quad in 2D space.
00127     /// @param size The size of the quad.
00128     /// @param rotationRads The rotation of the quad in radians.
00129     /// @param color The color of the quad.
00130     static void DrawQuadRotated(const glm::vec2& position, const glm::vec2& size, float
00131     rotationRads, const glm::vec4& color);
00132     /// @brief Draws a rotated colored quad at the specified position, size, and rotation.
00133     ///
00134     /// @param position The position of the quad in 3D space.
00135     /// @param size The size of the quad.
00136     /// @param rotationRads The rotation of the quad in radians.
00137     /// @param color The color of the quad.
00138     static void DrawQuadRotated(const glm::vec3& position, const glm::vec2& size, float
00139     rotationRads, const glm::vec4& color);
00140     /// @brief Draws a rotated textured quad with the given transform, texture, tiling factor, and
00141     /// tint color.
00142     ///
00143     /// @param transform The transform matrix for the quad.
00144     /// @param texture The texture to apply to the quad.
00145     /// @param tilingFactor The tiling factor for the texture.
00146     /// @param tintColor The tint color to apply to the texture.
00147     static void DrawQuadRotatedWithTexture(const glm::mat4& transform, const Ref<Texture2D>&
00148     texture, float tilingFactor, const glm::vec4 tintColor);
00149     /// @brief Draws a rotated textured quad at the specified position, size, and rotation.
00150     ///
00151     /// @param position The position of the quad in 2D space.
00152     /// @param size The size of the quad.
00153     /// @param rotationRads The rotation of the quad in radians.
00154     /// @param texture The texture to apply to the quad.
00155     /// @param tilingFactor The tiling factor for the texture.

```

```

00156     /// @param tintColor The tint color to apply to the texture.
00157     static void DrawQuadRotatedWithTexture(const glm::vec2& position, const glm::vec2& size, const
00158     Ref<Texture2D>& texture, float rotationRads, float tilingFactor, const glm::vec4 tintColor);
00159     /// @brief Draws a rotated textured quad at the specified position, size, and rotation.
00160     ///
00161     /// @param position The position of the quad in 3D space.
00162     /// @param size The size of the quad.
00163     /// @param rotationRads The rotation of the quad in radians.
00164     /// @param texture The texture to apply to the quad.
00165     /// @param tilingFactor The tiling factor for the texture.
00166     /// @param tintColor The tint color to apply to the texture.
00167     static void DrawQuadRotatedWithTexture(const glm::vec3& position, const glm::vec2& size, const
00168     Ref<Texture2D>& texture, float rotationRads, float tilingFactor, const glm::vec4 tintColor);
00169     /// @brief Draws a rotated textured quad with the given transform, subtexture, tiling factor,
00170     and tint color.
00171     ///
00172     /// @param transform The transform matrix for the quad.
00173     /// @param subtexture The subtexture to apply to the quad.
00174     /// @param tilingFactor The tiling factor for the texture.
00175     /// @param tintColor The tint color to apply to the texture.
00176     static void DrawQuadRotatedWithTexture(const glm::mat4& transform, const Ref<SubTexture2D>&
00177     subtexture, float tilingFactor, const glm::vec4 tintColor);
00178     /// @brief Draws a rotated textured quad at the specified position, size, and rotation.
00179     ///
00180     /// @param position The position of the quad in 2D space.
00181     /// @param size The size of the quad.
00182     /// @param rotationRads The rotation of the quad in radians.
00183     /// @param subtexture The subtexture to apply to the quad.
00184     /// @param tilingFactor The tiling factor for the texture.
00185     /// @param tintColor The tint color to apply to the texture.
00186     static void DrawQuadRotatedWithTexture(const glm::vec2& position, const glm::vec2& size, const
00187     Ref<SubTexture2D>& subtexture, float rotationRads, float tilingFactor, const glm::vec4 tintColor);
00188     /// @brief Draws a rotated textured quad at the specified position, size, and rotation.
00189     ///
00190     /// @param position The position of the quad in 3D space.
00191     /// @param size The size of the quad.
00192     /// @param rotationRads The rotation of the quad in radians.
00193     /// @param subtexture The subtexture to apply to the quad.
00194     /// @param tilingFactor The tiling factor for the texture.
00195     /// @param tintColor The tint color to apply to the texture.
00196     static void DrawQuadRotatedWithTexture(const glm::vec3& position, const glm::vec2& size, const
00197     Ref<SubTexture2D>& subtexture, float rotationRads, float tilingFactor, const glm::vec4 tintColor);
00198     //static void DrawSprite(const glm::mat4& transform, const SpriteRendererComponent& src, int
00199     entityID);
00200     //static void DrawSprite(const glm::mat4& transform, const SubTextureComponent& stc, int
00201     entityID);
00202     /// @brief Returns a reference to the default white texture that allows for coloring.
00203     static Ref<Texture2D> GetWhiteTexture();
00204     /// 2D Renderer Statistics
00205     struct Statistics {
00206         /// @brief The number of draw calls being made
00207         uint32_t DrawCalls = 0;
00208         /// @brief The number of quads being drawn
00209         uint32_t QuadCount = 0;
00210         /// @brief Calculates the total number of vertices drawn
00211         uint32_t GetTotalVertexCount() { return QuadCount * 4; }
00212         /// @brief Calculates the total number of indices drawn
00213         uint32_t GetTotalIndexCount() { return QuadCount * 6; }
00214     };
00215     /// @brief Resets the rendering statistics.
00216     static void ResetStats();
00217     /// @brief Retrieves the current rendering statistics.
00218     static Statistics GetStats();
00219
00220     private:
00221         static void FlushAndReset();
00222         static void StartBatch();
00223     };
00224 }
```

11.134 Vesper/src/Vesper/Renderer/RendererAPI.cpp File Reference

```
#include "vzpch.h"
#include "RendererAPI.h"
```

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.135 Vesper/src/Vesper/Renderer/RendererAPI.h File Reference

```
#include <glm/glm.hpp>
#include "VertexArray.h"
```

Classes

- class [Vesper::RendererAPI](#)

An abstract class defining the interface for a rendering API.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.136 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     /// @brief An abstract class defining the interface for a rendering API.
00011     class RendererAPI {
00012         public:
00013             /// @brief API
00014             enum class API {
00015                 None = 0,
00016                 OpenGL = 1,
00017             };
00018
00019         public:
00020             virtual ~RendererAPI() = default;
00021             /// @brief Initializes the rendering API.
00022             virtual void Init() = 0;
00023             /// @brief Sets the viewport dimensions.
00024             virtual void SetViewport(uint32_t x, uint32_t y, uint32_t width, uint32_t height) = 0;
00025             /// @brief Sets the clear color for the rendering API.
00026             virtual void SetClearColor(const glm::vec4& color) = 0;
00027             /// @brief Clears the rendering buffers.
```

```

00028     virtual void Clear() = 0;
00029
00030     /// @brief Draws indexed geometry using the provided vertex array.
00031     virtual void DrawIndexed(const Ref<VertexArray>& vertexArray, uint32_t indexCount = 0) = 0;
00032
00033     /// @brief Returns the current rendering API.
00034     inline static API GetAPI() { return s_API; }
00035
00036     private:
00037     static API s_API;
00038
00039
00040 }
```

11.137 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.

11.138 Vesper/src/Vesper/Renderer/Shader.h File Reference

```
#include <string>
#include <unordered_map>
#include <glm/glm.hpp>
```

Classes

- class [Vesper::Shader](#)
An abstraction for a shader program.
- class [Vesper::ShaderLibrary](#)
A library for managing and storing shaders.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.139 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     /// @brief An abstraction for a shader program.
00011     class Shader
00012     {
00013         public:
00014
00015         virtual ~Shader() = default;
00016
00017         /// @brief connects the shader program for use.
00018         virtual void Bind() const = 0;
00019         /// @brief disconnects the shader program.
00020         virtual void Unbind() const = 0;
00021
00022         /// @brief Sets a 4x4 matrix uniform in the shader.
00023         virtual void SetMat4(const std::string& name, const glm::mat4& value) = 0;
00024         /// @brief Sets a 4-component float vector uniform in the shader.
00025         virtual void SetFloat4(const std::string& name, const glm::vec4& value) = 0;
00026         /// @brief Sets a 3-component float vector uniform in the shader.
00027         virtual void SetFloat3(const std::string& name, const glm::vec3& value) = 0;
00028         /// @brief Sets a single float uniform in the shader.
00029         virtual void SetFloat(const std::string& name, float value) = 0;
00030         /// @brief Sets a single integer uniform in the shader.
00031         virtual void SetInt(const std::string& name, int value) = 0;
00032         /// @brief Sets an array of integers uniform in the shader.
00033         virtual void SetIntArray(const std::string& name, int* values, uint32_t count) = 0;
00034
00035
00036         static Ref<Shader> Create(const std::string& name, const std::string& vertexSrc, const
00037             std::string& fragmentSrc);
00038         static Ref<Shader> Create(const std::string& filepath);
00039
00040         virtual const std::string& GetName() const = 0;
00041     };
00042
00043     /// @class ShaderLibrary
00044     /// @brief A library for managing and storing shaders.
00045     class ShaderLibrary
00046     {
00047         public:
00048             /// @brief Adds a shader to the library with the specified name.
00049             void Add(const std::string& name, const Ref<Shader>& shader);
00050             /// @brief Adds a shader to the library using its own name.
00051             void Add(const Ref<Shader>& shader);
00052             /// @brief Loads a shader from the specified filepath and adds it to the library.
00053             Ref<Shader> Load(const std::string& filepath);
00054             /// @brief Loads a shader from the specified filepath and adds it to the library with the
00055             /// given name.
00056             Ref<Shader> Load(const std::string& name, const std::string& filepath);
00057             /// @brief Retrieves a shader from the library by name.
00058             Ref<Shader> Get(const std::string& name);
00059             /// @brief Checks if a shader with the specified name exists in the library.
00060             bool Exists(const std::string& name) const;
00061
00062         private:
00063             std::unordered_map<std::string, Ref<Shader>> m_Shaders;
00064     };
00065
00066 }

```

11.140 Vesper/src/Vesper/Renderer/SubTexture2D.cpp File Reference

```

#include "vzpch.h"
#include "SubTexture2D.h"

```

Namespaces

- namespace **Vesper**

TEMPORARY.

11.141 Vesper/src/Vesper/Renderer/SubTexture2D.h File Reference

```
#include <glm/glm.hpp>
#include "Texture.h"
```

Classes

- class **Vesper::SubTexture2D**

Represents a sub-region of a 2D texture, useful for sprite sheets.

Namespaces

- namespace **Vesper**

TEMPORARY.

11.142 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     /// @brief Represents a sub-region of a 2D texture, useful for sprite sheets.
00012     class SubTexture2D
00013     {
00014         public:
00015             /// @brief Constructs a SubTexture2D from the given texture and texture coordinates.
00016             ///
00017             /// @param texture The texture from which the sub-texture is derived.
00018             /// @param min The minimum texture coordinates (bottom-left).
00019             /// @param max The maximum texture coordinates (top-right).
00020             SubTexture2D(const Ref<Texture2D>& texture, const glm::vec2& min, const glm::vec2& max);
00021
00022             /// @brief Returns the underlying texture of the sub-texture.
00023             const Ref<Texture2D> GetTexture() { return m_Texture; }
00024             /// @brief Returns the texture coordinates of the sub-texture.
00025             glm::vec2* GetTexCoords() { return m_TexCoords; }
00026
00027             /// @brief Creates a SubTexture2D from a grid of cells within the given texture.
00028             ///
00029             /// @param texture The texture from which the sub-texture is derived.
00030             /// @param coords The coordinates of the cell in the grid.
00031             /// @param cellSize The size of each cell in the grid.
00032             /// @param spriteSize The size of the sprite in cells (default is 1x1).
00033             static Ref<SubTexture2D> CreateFromCoords(const Ref<Texture2D>& texture, const glm::vec2&
00034                 coords, const glm::vec2& cellSize, const glm::vec2& spriteSize = {1, 1});
00035         private:
00036             /// @brief The underlying texture of the sub-texture.
00037             Ref<Texture2D> m_Texture;
00038             /// @brief The texture coordinates of the sub-texture.
00039             glm::vec2 m_TexCoords[4];
00040     };
00041 }
```

11.143 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.

11.144 Vesper/src/Vesper/Renderer/Texture.h File Reference

Classes

- class [Vesper::Texture](#)
An abstraction for a texture.
- class [Vesper::Texture2D](#)
An abstraction for a 2D texture.
- class [Vesper::TextureLibrary](#)
A library for managing and storing textures.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.145 Texture.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 namespace Vesper {
00004
00005     class Texture2D;
00006
00007     /// @class Texture
00008     /// @brief An abstraction for a texture.
00009     class Texture
00010     {
00011         public:
00012             virtual ~Texture() = default;
00013
00014             /// @brief Returns the width of the texture.
00015             virtual uint32_t GetWidth() const = 0;
00016             /// @brief Returns the height of the texture.
00017             virtual uint32_t GetHeight() const = 0;
00018             /// @brief Returns the renderer ID of the texture.
00019             virtual uint32_t GetRendererID() const = 0;
00020
00021             /// @brief Binds the texture to the specified slot for use.
00022             virtual void Bind(uint32_t slot = 0) const = 0;
00023 }
```

```

00024     /// @brief Sets the data of the texture.
00025     ///
00026     /// @param data Pointer to the data to be set.
00027     /// @param size Size of the data in bytes.
00028     virtual void SetData(void* data, uint32_t size) = 0;
00029
00030     virtual bool operator==(const Texture2D& other) const = 0;
00031     virtual std::string GetName() const = 0;
00032 };
00033
00034 /// @class Texture2D
00035 /// @brief An abstraction for a 2D texture.
00036 class Texture2D : public Texture
00037 {
00038 public:
00039     static Ref<Texture2D> Create(uint32_t width, uint32_t height);
00040     static Ref<Texture2D> Create(const std::string& path);
00041 };
00042
00043
00044 /// @class TextureLibrary
00045 /// @brief A library for managing and storing textures.
00046 ///
00047 /// No current serialization implemented.
00048 class TextureLibrary
00049 {
00050 public:
00051     /// @brief Adds a texture to the library with the specified name.
00052     void Add(const std::string& name, const Ref<Texture2D>& texture);
00053     /// @brief Adds a texture to the library using its own name.
00054     void Add(const Ref<Texture2D>& texture);
00055     /// @brief Loads a texture from the specified filepath and adds it to the library.
00056     Ref<Texture2D> Load(const std::string& filepath);
00057     /// @brief Loads a texture from the specified filepath and adds it to the library with the
00058     /// given name.
00059     Ref<Texture2D> Load(const std::string& name, const std::string& filepath);
00060     /// @brief Retrieves a texture from the library by name.
00061     Ref<Texture2D> Get(const std::string& name) const;
00062     /// @brief Checks if a texture with the specified name exists in the library.
00063     bool Exists(const std::string& name) const;
00064 private:
00065     std::unordered_map<std::string, Ref<Texture2D>> m_Textures;
00066 };
00067 }
```

11.146 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.

11.147 Vesper/src/Vesper/Renderer/UniformBuffer.h File Reference

```
#include "Vesper/Core/Base.h"
```

Classes

- class [Vesper::UniformBuffer](#)

An abstraction for a uniform buffer object (UBO).

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.148 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     /// @brief An abstraction for a uniform buffer object (UBO).
00009     class UniformBuffer
0010     {
0011     public:
0012         virtual ~UniformBuffer() {}
0013         virtual void SetData(const void* data, uint32_t size, uint32_t offset = 0) = 0;
0014
0015         static Ref<UniformBuffer> Create(uint32_t size, uint32_t binding);
0016     };
0017
0018 }
```

11.149 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.

11.150 Vesper/src/Vesper/Renderer/VertexArray.h File Reference

```
#include <memory>
#include "Vesper/Renderer/Buffer.h"
```

Classes

- class [Vesper::VertexArray](#)
An abstraction for a vertex array object (VAO).

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.151 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     /// @class VertexArray
00009     /// @brief An abstraction for a vertex array object (VAO).
00010     class VertexArray
00011     {
00012     public:
00013         virtual ~VertexArray() {}
00014
00015         virtual void Bind() const = 0;
00016         virtual void Unbind() const = 0;
00017
00018         /// @brief Adds a vertex buffer to the vertex array.
00019         virtual void AddVertexBuffer(const Ref<VertexBuffer>& vertexBuffer) = 0;
00020         /// @brief Sets the index buffer for the vertex array.
00021         virtual void SetIndexBuffer(const Ref<IndexBuffer>& indexBuffer) = 0;
00022
00023         virtual const std::vector<Ref<VertexBuffer>>& GetVertexBuffers() = 0;
00024         virtual const Ref<IndexBuffer>& GetIndexBuffer() const = 0;
00025
00026     static Ref<VertexArray> Create();
00027 };
00028 }
```

11.152 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](#)
Universally Unique Identifier.
- struct [Vesper::UUIDComponent](#)
Component that holds a [UUID](#).
- struct [Vesper::NameComponent](#)
Component that holds the name of an entity.
- struct [Vesper::TransformComponent](#)
Component that holds the transform of an entity.
- struct [Vesper::SpriteRendererComponent](#)
Component that holds sprite rendering data.
- struct [Vesper::SubTextureComponent](#)
Component that holds sub-texture data for sprites.
- struct [Vesper::TextureAnimationComponent](#)
Animates through a series of sub textures.
- struct [Vesper::CameraComponent](#)
Component that holds camera data.
- struct [Vesper::NativeScriptComponent](#)
Component that holds scripting data for an entity.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

Macros

- #define [GLM_ENABLE_EXPERIMENTAL](#)

11.152.1 Macro Definition Documentation

11.152.1.1 GLM_ENABLE_EXPERIMENTAL

```
#define GLM_ENABLE_EXPERIMENTAL
```

11.153 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     /// @brief Universally Unique Identifier
00015     ///
00016     /// A simple wrapper around a string representing a UUID.
00017     struct UUID {
00018         /// @brief The string representation of the UUID.
00019         std::string ID;
00020
00021         UUID() { ID = Random::UUID(); }
00022         UUID(const std::string& id)
00023             : ID{ id } {}
00024
00025         operator std::string& () { return ID; }
00026         operator const std::string& () const { return ID; }
00027     };
00028
00029     /// @brief Component that holds a UUID.
00030     struct UUIDComponent {
00031         /// @brief The UUID of the owning entity.
00032         UUID ID;
00033
00034         UUIDComponent()
00035             : ID() {}
00036
00037         UUIDComponent(const UUIDComponent&) = default;
00038         UUIDComponent(const std::string& id)
00039             : ID{ id } {}
00040
00041     };
00042
00043
00044     /// @brief Component that holds the name of an entity.
00045     struct NameComponent
00046     {
00047         /// @brief The name of the owning entity.
00048         std::string Name;
00049
00050         NameComponent() = default;
00051         NameComponent(const NameComponent&) = default;
00052         NameComponent(const std::string& name)
00053             : Name(name) {}
00054
00055         operator std::string& () { return Name; }
00056         operator const std::string& () const { return Name; }
00057         std::string& GetName() { return Name; }
00058     };
00059
00060
00061     /// @brief Component that holds the transform of an entity.
00062     struct TransformComponent
00063     {
00064         /// @brief Translation (position) vector.
00065         glm::vec3 Translation = { 0.0f, 0.0f, 0.0f };
00066         /// @brief Rotation vector (in radians).
00067         glm::vec3 Rotation = { 0.0f, 0.0f, 0.0f };
00068         /// @brief Scale vector.
00069         glm::vec3 Scale = { 1.0f, 1.0f, 1.0f };
00070
00071         TransformComponent() = default;
00072         TransformComponent(const TransformComponent&) = default;
00073         /// @brief Constructor that initializes the translation with a 3D vector.
00074         TransformComponent(const glm::vec3& translation)
00075             : Translation(translation) {}
00076
00077         /// @brief Calculates and returns the transformation matrix.
00078         glm::mat4 GetTransform() const
00079         {
00080             glm::mat4 rotation = glm::toMat4(glm::quat(Rotation));
00081             return glm::translate(glm::mat4(1.0f), Translation)

```

```
00083         * rotation
00084         * glm::scale(glm::mat4(1.0f), Scale);
00085     }
00086 };
00087
00088 /// @brief Component that holds sprite rendering data.
00089 struct SpriteRendererComponent
00090 {
00091     /// @brief Color of the sprite.
00092     glm::vec4 Color{ 1.0f, 1.0f, 1.0f, 1.0f };
00093     /// @brief Texture of the sprite.
00094     Ref<Texture2D> Texture = nullptr;
00095     /// @brief Tiling factor for the texture.
00096     float TilingFactor = 1.0f;
00097
00098     SpriteRendererComponent() = default;
00099     SpriteRendererComponent(const SpriteRendererComponent&) = default;
00100     /// @brief Constructor that initializes the color.
00101     SpriteRendererComponent(const glm::vec4& color)
00102         : Color(color) {}
00103     }
00104
00105     operator glm::vec4& () { return Color; }
00106     operator const glm::vec4& () const { return Color; }
00107
00108     /// @brief Returns the color of the sprite.
00109     glm::vec4& GetColor() { return Color; }
00110
00111     /// @brief whether the texturing is enabled
00112     bool TextureEnabled = false;
00113
00114     /// @brief whether the sprite should always face the camera
00115     ///
00116     /// WIP
00117     bool Billboard = false;
00118 };
00119
00120 /// @brief Component that holds sub-texture data for sprites.
00121 struct SubTextureComponent
00122 {
00123     /// @brief The sub-texture reference.
00124     Ref<SubTexture2D> SubTexture;
00125     /// @brief Tiling factor for the sub-texture.
00126     glm::vec2 TilingFactor = { 1.0f, 1.0f };
00127     /// @brief Offset for the sub-texture.
00128     glm::vec2 Offset = { 0.0f, 0.0f };
00129
00130
00131     SubTextureComponent() = default;
00132
00133     /// @brief Constructor that initializes the sub-texture from a full texture.
00134     SubTextureComponent(const Ref<Texture2D>& texture)
00135         : SubTexture(SubTexture2D::CreateFromCoords(texture, { 0, 0 }, { texture->GetWidth(),
00136             texture->GetHeight() })) {
00137         }
00138         /// @brief Constructor that initializes the sub-texture directly.
00139         SubTextureComponent(const Ref<SubTexture2D>& subTexture)
00140             : SubTexture(subTexture) {
00141             }
00142             /// @brief Copy constructor.
00143             void SetTexture(const Ref<Texture2D>& texture) {
00144                 SubTexture = SubTexture2D::CreateFromCoords(texture, { 0, 0 }, { texture->GetWidth(),
00145                     texture->GetHeight() });
00146                 }
00147
00148             /// @brief Sets the tiling factor for the sub-texture.
00149             void SetTilingFactor(const glm::vec2& tiling) {
00150                 TilingFactor = tiling;
00151                 }
00152
00153             /// @brief Sets the offset for the sub-texture.
00154             void SetOffset(const glm::vec2& offset) {
00155                 Offset = offset;
00156                 }
00157
00158         operator Ref<SubTexture2D>& () { return SubTexture; }
00159         operator const Ref<SubTexture2D>& () const { return SubTexture; }
00160         Ref<SubTexture2D>& GetSubTexture() { return SubTexture; }
00161     };
00162
00163     /// @brief Animates through a series of sub textures
00164     ///
00165     /// (can be used with full textures)
00166     struct TextureAnimationComponent
00167     {
00168         /// @brief The list of sub-textures for the animation.
00169         std::vector<Ref<SubTexture2D>> SubTextures;
```

```

00168     /// @brief The current frame index.
00169     uint32_t CurrentFrame = 0;
00170     /// @brief Time per frame in seconds.
00171     float FrameTime = 0.6f; // Time per frame in seconds
00172     /// @brief Accumulated time for frame switching.
00173     float TimeAccumulator = 0.0f; // per-instance accumulator
00174
00175     TextureAnimationComponent() = default;
00176     TextureAnimationComponent(const TextureAnimationComponent&) = default;
00177     /// @brief Constructor that initializes the sub-textures and frame time.
00178     ///
00179     /// @param subTextures Vector of sub-textures for the animation.
00180     /// @param frameTime Time per frame in seconds.
00181     TextureAnimationComponent(const std::vector<Ref<SubTexture2D>& subTextures, float frameTime)
00182         : SubTextures(subTextures), FrameTime(frameTime) {
00183     }
00184     operator std::vector<Ref<SubTexture2D>& () { return SubTextures; }
00185     operator const std::vector<Ref<SubTexture2D>& () const { return SubTextures; }
00186
00187     std::vector<Ref<SubTexture2D>& GetSubTextures() { return SubTextures; }
00188
00189     /// @brief Returns the index of the current frame.
00190     uint32_t GetCurrentFrame() const { return CurrentFrame; }
00191
00192     /// @brief Updates the animation based on the elapsed time.
00193     ///
00194     /// @param deltaTime The elapsed time since the last update.
00195     void Update(float deltaTime) {
00196         if (SubTextures.empty() || FrameTime <= 0.0f)
00197             return;
00198
00199         TimeAccumulator += deltaTime;
00200         while (TimeAccumulator >= FrameTime) {
00201             CurrentFrame = (CurrentFrame + 1) % static_cast<uint32_t>(SubTextures.size());
00202             TimeAccumulator -= FrameTime;
00203         }
00204     }
00205 };
00206
00207     /// @brief Component that holds camera data.
00208     struct CameraComponent
00209     {
00210         /// @brief The scene camera.
00211         SceneCamera Camera;
00212
00213         /// @brief Whether this camera is the primary camera.
00214         ///
00215         /// If multiple cameras exist, the primary camera for rendering will be the first one found
00216         /// marked as primary.
00217         bool Primary = true;
00218
00219         /// @brief Whether the aspect ratio is fixed.
00220         bool FixedAspectRatio = false;
00221
00222         CameraComponent() = default;
00223         CameraComponent(const CameraComponent&) = default;
00224     };
00225
00226     class ScriptableEntity;
00227     class Timestep;
00228
00229     /// @brief Component that holds scripting data for an entity.
00230     struct NativeScriptComponent
00231     {
00232         /// @brief Pointer to the instance of the scriptable entity.
00233         ScriptableEntity* Instance = nullptr;
00234
00235         /// @brief Function pointer to instantiate the script.
00236         ScriptableEntity* (*InstantiateScript)();
00237         /// @brief Function pointer to destroy the script.
00238         void (*DestroyScript)(NativeScriptComponent*);
00239
00240         /// @brief Binds a script type to this component.
00241         ///
00242         /// @tparam T The type of the script to bind.
00243         template<typename T>
00244         void Bind()
00245         {
00246             InstantiateScript = []() { return static_cast<ScriptableEntity*> (new T()); };
00247             DestroyScript = [](NativeScriptComponent* nsc) { delete nsc->Instance; nsc->Instance =
00248                 nullptr; };
00249         };
00250
00251     };

```

11.154 Vesper/src/Vesper/Scene/Entity.cpp File Reference

```
#include "vzpch.h"
#include "Entity.h"
```

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.155 Vesper/src/Vesper/Scene/Entity.h File Reference

```
#include "entt.hpp"
#include "Scene.h"
```

Classes

- class [Vesper::Entity](#)

Represents an entity in a scene.

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.156 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     /// @brief Represents an entity in a scene.
00011     ///
00012     /// Contains a handle to an entity (entt::entity) in a scene and provides methods to manipulate
00013     /// its components.
00014     class Entity
00015     {
00016     public:
00017         Entity() = default;
00018
00019         /// @brief Constructs an Entity with the given handle and scene.
00020         ///
00021         /// @param handle The handle to the entity.
00022         /// @param scene The scene to which the entity belongs.
00023         Entity(entt::entity handle, Scene* scene);
00024
00025         /// @brief Copy constructor for Entity.
00026         Entity(const Entity& other) = default;
```

```

00027
00028     /// Checks if the entity has a component of type T.
00029     ///
00030     /// @param T The type of component to check for.
00031     /// @return true if the entity has the component, false otherwise.
00032     template<typename T>
00033     bool HasComponent() const
00034     {
00035         return m_Scene->m_Registry.all_of<T>(m_EntityID);
00036     }
00037
00038     /// Adds a component of type T to the entity with the provided arguments Otherwise, asserts if
00039     /// the entity already has the component.
00040     ///
00041     /// @param T The type of component to add.
00042     /// @param Args The types of arguments to forward to the component's constructor.
00043     /// @param args The arguments to forward to the component's constructor.
00044     /// @return A reference to the newly added component.
00045     template<typename T, typename... Args>
00046     T& AddComponent(Args&&... args)
00047     {
00048         VZ_CORE_ASSERT(!HasComponent<T>(), "Entity already has component!");
00049         T& component = m_Scene->m_Registry.emplace<T>(m_EntityID, std::forward<Args>(args)...);
00050         m_Scene->OnComponentAdded<T>(*this, component);
00051         return component;
00052     }
00053
00054     /// Adds or replaces a component of type T to the entity with the provided arguments.
00055     ///
00056     /// @param T The type of component to add or replace.
00057     /// @param Args The types of arguments to forward to the component's constructor.
00058     /// @param args The arguments to forward to the component's constructor.
00059     /// @return A reference to the newly added or replaced component.
00060     template<typename T, typename... Args>
00061     T& AddOrReplaceComponent(Args&&... args)
00062     {
00063         return m_Scene->m_Registry.emplace_or_replace<T>(m_EntityID, std::forward<Args>(args)...);
00064     }
00065
00066     /// Retrieves a reference to the component of type T attached to the entity if it exists.
00067     /// Otherwise, asserts.
00068     ///
00069     /// @param T The type of component to retrieve.
00070     /// @return A reference to the component.
00071     template<typename T>
00072     T& GetComponent()
00073     {
00074         VZ_CORE_ASSERT(HasComponent<T>(), "Entity does not have component!");
00075         return m_Scene->m_Registry.get<T>(m_EntityID);
00076     }
00077
00078     /// Retrieves a reference to the component of type T attached to the entity if it exists.
00079     /// Otherwise, adds the component with the provided arguments and returns it.
00080     ///
00081     /// @param T The type of component to add.
00082     /// @param Args The types of arguments to forward to the component's constructor.
00083     /// @param args The arguments to forward to the component's constructor.
00084     /// @return A reference to the newly added component.
00085     template<typename T, typename... Args>
00086     T& GetOrAddComponent(Args&&... args)
00087     {
00088         if (HasComponent<T>())
00089             return GetComponent<T>();
00090         else
00091             return AddComponent<T>(std::forward<Args>(args)...);
00092     }
00093
00094     /// Removes the component of type T from the entity if it exists. Otherwise, asserts.
00095     ///
00096     /// @param T The type of component to remove.
00097     template<typename T>
00098     void RemoveComponent()
00099     {
00100         VZ_CORE_ASSERT(HasComponent<T>(), "Entity does not have component!");
00101         m_Scene->m_Registry.remove<T>(m_EntityID);
00102     }
00103
00104     /// Retrieves a const reference to the UUID of the entity.
00105     ///
00106     /// @return A const reference to the UUID of the entity.
00107     const UUID& GetID()
00108     {
00109         return GetComponent<UUIDComponent>().ID;
00110     }
00111
00112     /// Retrieves a const reference to the name of the entity.
00113     ///

```

```

00111     /// @return A const reference to the name of the entity.
00112     const std::string& GetName()
00113     {
00114         return GetComponent<NameComponent>().Name;
00115     }
00116
00117     operator bool() const { return m_EntityID != entt::null; }
00118     operator entt::entity() const { return m_EntityID; }
00119     operator uint32_t() const { return (uint32_t)m_EntityID; }
00120     bool operator==(const Entity& other) const { return m_EntityID == other.m_EntityID &&
00121     m_Scene == other.m_Scene; }
00122     bool operator!=(const Entity& other) const { return !(this == other); }
00123
00124     private:
00125         /// @brief The unique identifier of the entity within the scene.
00126         entt::entity m_EntityID {entt::null};
00127         /// @brief Pointer to the scene that contains the entity.
00128         Scene* m_Scene = nullptr;
00129     };
00130
00131
00132 }
```

11.157 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.

11.158 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.

11.159 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 }
```

11.160 Vesper/src/Vesper/Scene/SceneCamera.cpp File Reference

```
#include "vzpch.h"
#include "SceneCamera.h"
#include <glm/gtc/matrix_transform.hpp>
```

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.161 Vesper/src/Vesper/Scene/SceneCamera.h File Reference

```
#include "Vesper/Renderer/Camera.h"
```

Classes

- class [Vesper::SceneCamera](#)

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.162 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;
00021                 RecalculateProjection(); }
00022             float GetPerspectiveNearClip() const { return m_PerspectiveNear; }
00023             void SetPerspectiveNearClip(float nearClip) { m_PerspectiveNear = nearClip;
00024                 RecalculateProjection(); }
00025             float GetPerspectiveFarClip() const { return m_PerspectiveFar; }
00026             void SetPerspectiveFarClip(float farClip) { m_PerspectiveFar = farClip;
00027                 RecalculateProjection(); }
00028
00029             float GetOrthographicSize() const { return m_OrthographicSize; }
00030             void SetOrthographicSize(float size) { m_OrthographicSize = size; RecalculateProjection(); }
00031             float GetOrthographicNearClip() const { return m_OrthographicNear; }
00032             void SetOrthographicNearClip(float nearClip) { m_OrthographicNear = nearClip;
00033                 RecalculateProjection(); }
00034             float GetOrthographicFarClip() const { return m_OrthographicFar; }
00035             void SetOrthographicFarClip(float farClip) { m_OrthographicFar = farClip;
00036                 RecalculateProjection(); }
00037
00038         private:
00039             ProjectionType GetProjectionType() const { return m_ProjectionType; }
00040             void SetProjectionType(ProjectionType type) { m_ProjectionType = type;
00041                 RecalculateProjection(); }
00042             private:
00043                 ProjectionType m_ProjectionType = ProjectionType::Orthographic;
00044
00045                 float m_PerspectiveFOV = glm::radians(45.0f);
00046                 float m_PerspectiveNear = 0.01f, m_PerspectiveFar = 1000.0f;
00047
00048                 float m_OrthographicSize = 10.0f;
00049                 float m_OrthographicNear = -1.0f, m_OrthographicFar = 1.0f;
00050             };
00051 }
```

11.163 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 \(YAML::Emitter &out, Entity entity\)](#)

11.164 Vesper/src/Vesper/Scene/SceneSerializer.h File Reference

```
#include "Vesper/Core/Base.h"
#include "Scene.h"
```

Classes

- class [Vesper::SceneSerializer](#)

Namespaces

- namespace [Vesper](#)

TEMPORARY.

11.165 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);
0010
0011         void Serialize(const std::string& filepath);
0012         void SerializeRuntime(const std::string& filepath);
0013
0014         bool Deserialize(const std::string& filepath);
0015         bool DeserializeRuntime(const std::string& filepath);
0016     private:
0017         Ref<Scene> m_Scene;
0018     };
0019 }
```

11.166 Vesper/src/Vesper/Scene/ScriptableEntity.h File Reference

```
#include "Entity.h"
```

Classes

- class [Vesper::ScriptableEntity](#)

Base class for scriptable entities within a scene.

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.167 ScriptableEntity.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "Entity.h"
00004
00005 namespace Vesper {
00006
00007     /// @brief Base class for scriptable entities within a scene.
00008     ///
00009     /// A ScriptableEntity allows for custom behavior to be defined for entities in a scene, not
00010     /// already provided as standardized components.
00011     class ScriptableEntity
00012     {
00013     public:
00014         virtual ~ScriptableEntity() {};
00015
00016         /// @brief Retrieves a reference to a component of type T attached to the entity.
00017         ///
00018         /// @tparam T The type of the component to retrieve.
00019         /// @return Reference to the component of type T.
00020         /// Will assert if the component does not exist.
00021         template<typename T>
```

```

00021     T& GetComponent()
00022     {
00023         return m_Entity.GetComponent<T>();
00024     }
00025
00026     protected:
00027     /// @brief Sets the entity associated with this scriptable entity.
00028     ///
00029     /// Meant for internal use by the Scene class.
00030     /// Should be overridden in derived classes if additional setup is required when the entity is
00031     set.
00032     virtual void OnCreate() {}
00033
00034     /// @brief Called when the entity is destroyed.
00035     ///
00036     /// Meant for internal use by the Scene class.
00037     /// Should be overridden in derived classes to handle cleanup.
00038     virtual void OnDestroy() {}
00039
00040     /// @brief Called every frame to update the entity.
00041     ///
00042     /// @param ts The timestep representing the time elapsed since the last update.
00043     /// Custom behavior should be implemented here in derived classes.
00044     virtual void OnUpdate(Timestep ts) {}
00045
00046     private:
00047     Entity m_Entity;
00048     friend class Scene;
00049 };
00050 }
```

11.168 Vesper/src/Vesper/Utils/PlatformUtils.h File Reference

```
#include <string>
```

Classes

- class [Vesper::FileDialogs](#)
Cross-platform file dialog utilities.
- class [Vesper::FileSystem](#)

Namespaces

- namespace [Vesper](#)
TEMPORARY.

11.169 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     /// @brief Cross-platform file dialog utilities.
00009     class FileDialogs
00010     {
00011     public:
00012         /// @brief Opens a file dialog to select a file to open.
00013         ///
00014         /// @param filter The file type filter for the dialog.
00015         /// @return The selected file path or an empty string if cancelled.
00016 }
```

```
00016     static std::string OpenFile(const char* filter);
00017
00018     /// @brief Opens a file dialog to select a location to save a file.
00019     ///
00020     /// @param filter The file type filter for the dialog.
00021     /// @return The selected file path or an empty string if cancelled.
00022     static std::string SaveFile(const char* filter);
00023 };
00024
00025
00026
00027 class FileSystem
00028 {
00029 public:
00030     static void Initialize();
00031     static std::string GetCurrentWorkingDirectory();
00032     static std::string GetAbsolutePath(const std::string& relativePath);
00033     static std::string GetTravelingUpPath(const std::string& path);
00034     static bool IsInitialized() { return m_Initialized; }
00035
00036     static bool m_Initialized;
00037     static std::string m_RootEngineDirectory;
00038     static std::string m_RootEditorDirectory;
00039     static std::string m_ResourcesDirectory;
00040     static std::string m_AssetsDirectory;
00041     static std::string m_ProjectsDirectory;
00042     static std::string m_CurrentProjectDirectory;
00043 };
00044
00045
00046 };
```

11.170 Vesper/src/vzpch.cpp File Reference

```
#include "vzpch.h"
```

11.171 Vesper/src/vzpch.h File Reference

Precompiled header for the [Vesper](#) engine.

```
#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"
```

11.171.1 Detailed Description

Precompiled header for the [Vesper](#) engine.

Author

Damon S. Green II

11.172 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 /// @file vzpch.h
00023 /// @author Damon S. Green II
00024 /// @brief Precompiled header for the Vesper engine.
00025
00026
00027 #ifdef VZ_PLATFORM_WINDOWS
00028     #include <windows.h>
00029 #endif
```