engineOne

Generated by Doxygen 1.14.0

1 Class Index 1

1.1 Class List . . . 1

2 File Index 3

2.1 File List . . . 3

3 Class Documentation 5

3.1 Application Class Reference . . . 5

3.1.1 Detailed Description . . . 5

3.1.2 Constructor & Destructor Documentation . . . 5

3.1.2.1 Application() . . . 5

3.1.3 Member Function Documentation . . . 6

3.1.3.1 Init() . . . 6

3.2 CameraFPS Class Reference . . . 6

3.3 GLLoader Class Reference . . . 7

3.4 IndexBuffer Class Reference . . . 7

3.5 Mesh Struct Reference . . . 7

3.6 RenderContext Class Reference . . . 8

3.7 RenderContextCreateInfo Struct Reference . . . 8

3.8 Shader Class Reference . . . 9

3.9 ShaderProgram Class Reference . . . 9

3.10 Texture2D Class Reference . . . 10

3.11 Texture2DDataCreateInfo Struct Reference . . . 10

3.12 Timer Class Reference . . . 11

3.13 Vertex Struct Reference . . . 11

3.14 VertexArray Class Reference . . . 11

3.15 VertexBuffer Class Reference . . . 12

3.16 Window Class Reference . . . 12

4 File Documentation 13

4.1 Application.h . . . 13

4.2 Buffer.h . . . 14

4.3 Camera.h . . . 15

4.4 GLLoader.h . . . 15

4.5 glTypes.h . . . 16

4.6 glUtils.h . . . 16

4.7 Input.h . . . 17

4.8 keyCodes.h . . . 17

4.9 RenderContext.h . . . 19

4.10 Shader.h . . . 20

4.11 Texture.h . . . 21

4.12 Timer.h . . . 23

4.13 utils.h . . . 23

4.14 VertexArray.h 23
4.15 Window.h 24

Index **25**

Chapter 1

Class Index

1.1 Class List

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

Application
    Class for runnning engine Application 5
CameraFPS 6
GLLoader 7
IndexBuffer 7
Mesh 7
RenderContext 8
RenderContextCreateInfo 8
Shader 9
ShaderProgram 9
Texture2D 10
Texture2DDataCreateInfo 10
Timer 11
Vertex 11
VertexArray 11
VertexBuffer 12
Window 12

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

engineOne/src/Core/Application.h 13
engineOne/src/Core/Buffer.h 14
engineOne/src/Core/Camera.h 15
engineOne/src/Core/GLLoader.h 15
engineOne/src/Core/glTypes.h 16
engineOne/src/Core/glUtils.h 16
engineOne/src/Core/Input.h 17
engineOne/src/Core/keyCodes.h 17
engineOne/src/Core/RenderContext.h 19
engineOne/src/Core/Shader.h 20
engineOne/src/Core/Texture.h 21
engineOne/src/Core/Timer.h 23
engineOne/src/Core/utils.h 23
engineOne/src/Core/VertexArray.h 23
engineOne/src/Core/Window.h 24

Chapter 3

Class Documentation

3.1 Application Class Reference

class for runnning engine Application

`#include <Application.h>`

Public Member Functions

- Application (HINSTANCE hInstance, const std::string &appName) noexcept
  constructer for creating Application object
- **Application** (const Application &)=delete
- Application & **operator=** (const Application &)=delete
- bool Init () noexcept
  Initialize the App class.
- void **Run** () noexcept
  Run the application.

3.1.1 Detailed Description

class for runnning engine Application

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Application()

```
Application::Application (
            HINSTANCE hInstance,
            const std::string & appName)  [noexcept]
```

constructer for creating Application object

Parameters

hInstance	HINSTANCE from WinMain
appName	Name of the application which will be used to set window name

3.1.3 Member Function Documentation

3.1.3.1 Init()

```
bool Application::Init ()  [noexcept]
```

Initialize the App class.

Returns

returns true if initalizetion succeeds and false otherwise

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

- engineOne/src/Core/Application.h
- engineOne/src/Core/Application.cpp

3.2 CameraFPS Class Reference

Public Member Functions

- **CameraFPS** (float fov=60.0f, float aspect=4.0f/3.0f, float zNear=0.1f, float zFar=100.0f) noexcept
- void **MoveTo** (const glm::vec3 &position) noexcept
- void **Translate** (const glm::vec3 &displacement) noexcept
- void **Yaw** (float angle) noexcept
- void **Pitch** (float angle) noexcept
- void **Roll** (float angle) noexcept
- void **MoveForward** (float distance) noexcept
- void **MoveRight** (float distance) noexcept
- void **MoveUp** (float distance) noexcept
- void **UpdateProjection** (float fov, float aspect, float zNear, float zFar) noexcept
- const glm::mat4 & **GetViewMatrix** () const noexcept
- const glm::mat4 & **GetProjectionMatrix** () const noexcept
- void **RestPosAndOrient** () noexcept

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

- engineOne/src/Core/Camera.h
- engineOne/src/Core/Camera.cpp

3.3 GLLoader Class Reference

Public Member Functions

- **GLLoader** (HINSTANCE hInstance) noexcept
- **GLLoader** (const GLLoader &)=delete
- GLLoader & **operator=** (const GLLoader &)=delete
- **GLLoader** (GLLoader &&)=delete
- GLLoader & **operator=** (GLLoader &&)=delete
- bool **isLoaded** () const noexcept

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

- engineOne/src/Core/GLLoader.h

3.4 IndexBuffer Class Reference

Public Member Functions

- **IndexBuffer** (const void *data, unsigned int size, BufferUsage usage=DefaultBufferUsage) noexcept
- **IndexBuffer** (IndexBuffer &&other) noexcept
- IndexBuffer & **operator=** (IndexBuffer &&other) noexcept
- **IndexBuffer** (const IndexBuffer &)=delete
- IndexBuffer & **operator=** (const IndexBuffer &)=delete
- void **Bind** () const noexcept
- void **Unbind** () const noexcept
- void **SetData** (const void *data, unsigned int size, BufferUsage usage=DefaultBufferUsage) const noexcept
- unsigned int **getID** () const noexcept

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

- engineOne/src/Core/Buffer.h
- engineOne/src/Core/Buffer.cpp

3.5 Mesh Struct Reference

Public Attributes

- std::vector< Vertex > **vertices**
- std::vector< uint32_t > **indices**

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

- engineOne/src/Core/Application.h

3.6 RenderContext Class Reference

Public Member Functions

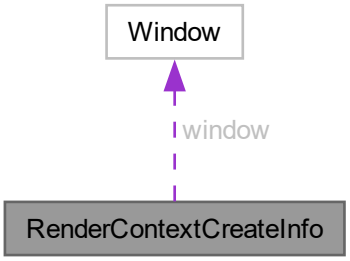
- **RenderContext** (const RenderContextCreateInfo &createInfo)
- **RenderContext** (Window &window, int majorVersion=4, int minorVersion=5, bool debug=true)
- **RenderContext** (const RenderContext &)=delete
- RenderContext & **operator=** (const RenderContext &)=delete
- bool **MakeCurrent** (Window &window) noexcept
- void **clearColor** (float r, float g, float b, float a) noexcept
- void **Present** () noexcept
- bool **IsNull** () const noexcept
- **operator bool** () const noexcept

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

- engineOne/src/Core/RenderContext.h
- engineOne/src/Core/RenderContext.cpp

3.7 RenderContextCreateInfo Struct Reference

Collaboration diagram for RenderContextCreateInfo:



Public Member Functions

- **RenderContextCreateInfo** (Window &win, int maj=4, int min=5, bool debug=true, int color=32, int depth=24, int stencil=8, int sampleCount=0)

Public Attributes

- Window & **window**
- int **majorVersion** = 4
- int **minorVersion** = 5
- bool **debugContext** = true
- int **colorBits** = 32
- int **depthBits** = 24
- int **stencilBits** = 8
- int **samples** = 0

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

- engineOne/src/Core/RenderContext.h

3.8 Shader Class Reference

Public Member Functions

- **Shader** (ShaderType type) noexcept
- **Shader** (ShaderType type, std::string &data, ShaderLoadOption loadOption)
- **Shader** (Shader &&) noexcept
- **Shader** (const Shader &)=delete
- Shader & **operator=** (const Shader &)=delete
- Shader & **operator=** (Shader &&) noexcept
- bool **loadFromFile** (const std::string &filePath)
- bool **loadFromString** (const std::string &shaderSrc)
- ShaderType **getType** () const
- unsigned int **getID** () const

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

- engineOne/src/Core/Shader.h
- engineOne/src/Core/Shader.cpp

3.9 ShaderProgram Class Reference

Public Member Functions

- **ShaderProgram** (ShaderProgram &&) noexcept
- **ShaderProgram** (const std::string &vertexPath, const std::string &fragmentPath)
- **ShaderProgram** (Shader &vertexShader, Shader &fragmentShader)
- **ShaderProgram** (const ShaderProgram &)=delete
- ShaderProgram & **operator=** (const ShaderProgram &)=delete
- ShaderProgram & **operator=** (ShaderProgram &&) noexcept
- void **attachShader** (const Shader &shader)
- bool **link** ()
- void **Bind** () const
- void **Unbind** () const

- unsigned int **getID** () const
- bool **isValid** () const
- bool **checkLinkStatus** ()
- void **SetUniform1i** (const std::string &uniformName, int v) noexcept
- void **SetUniform1f** (const std::string &uniformName, float v) noexcept
- void **SetUniform2f** (const std::string &uniformName, float v0, float v1) noexcept
- void **SetUniform3f** (const std::string &uniformName, float v0, float v1, float v2) noexcept
- void **SetUniform4f** (const std::string &uniformName, float v0, float v1, float v2, float v3) noexcept
- void **SetUniformVec2** (const std::string &uniformName, const glm::vec2 &vec) noexcept
- void **SetUniformVec3** (const std::string &uniformName, const glm::vec3 &vec) noexcept
- void **SetUniformVec4** (const std::string &uniformName, const glm::vec4 &vec) noexcept
- void **SetUniformMat2** (const std::string &uniformName, const glm::mat2 &mat) noexcept
- void **SetUniformMat3** (const std::string &uniformName, const glm::mat3 &mat) noexcept
- void **SetUniformMat4** (const std::string &uniformName, const glm::mat4 &mat) noexcept

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

- engineOne/src/Core/Shader.h
- engineOne/src/Core/Shader.cpp

3.10 Texture2D Class Reference

Public Member Functions

- **Texture2D** (int width, int height, void ∗data, pixelDataType type, TextureInternalFormat internalFormat, int mipLevels=1) noexcept
- **Texture2D** (const Texture2DDataCreateInfo &createInfo) noexcept
- **Texture2D** (const Texture2D &)=delete
- Texture2D & **operator=** (const Texture2D &)=delete
- **Texture2D** (Texture2D &&other) noexcept
- Texture2D & **operator=** (Texture2D &&other) noexcept
- void **SubImage** (int width, int height, const void ∗data, pixelDataType type, TextureBaseFormat baseFormat, int level=0, int xOfffset=0, int yOffset=0) const noexcept
- void **Bind** (unsigned int slot=0) const noexcept
- void **Unbind** (unsigned int slot=0) const noexcept

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

- engineOne/src/Core/Texture.h
- engineOne/src/Core/Texture.cpp

3.11 Texture2DDataCreateInfo Struct Reference

Public Attributes

- int **width**
- int **height**
- void ∗ **data**
- pixelDataType **type**
- TextureInternalFormat **internalFormat**
- int **mipLevels** = 1

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

- engineOne/src/Core/Texture.h

3.12 Timer Class Reference

Public Member Functions

- **Timer** (const Timer &) noexcept=delete
- Timer & **operator=** (const Timer &) noexcept=delete
- **Timer** (Timer &&) noexcept=default
- Timer & **operator=** (Timer &&) noexcept=default
- void **reset** ()
- void **resetFrequency** () noexcept
- double **elapsed** () const noexcept
- double **elapsedAndReset** () noexcept

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

- engineOne/src/Core/Timer.h

3.13 Vertex Struct Reference

Public Attributes

- glm::vec3 **position**
- float **u**
- float **v**

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

- engineOne/src/Core/Application.h

3.14 VertexArray Class Reference

Public Member Functions

- **VertexArray** (VertexArray &&other) noexcept
- VertexArray & **operator=** (VertexArray &&other) noexcept
- **VertexArray** (const VertexArray &)=delete
- VertexArray & **operator=** (const VertexArray &)=delete
- void **Bind** () const noexcept
- void **Unbind** () const noexcept
- void **addAttribute** (unsigned int index, int size, GLType type, bool normalized, int stride, const void ∗pointer) const noexcept
- unsigned int **getID** () const noexcept

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

- engineOne/src/Core/VertexArray.h
- engineOne/src/Core/VertexArray.cpp

3.15 VertexBuffer Class Reference

Public Member Functions

- **VertexBuffer** (const void *data, unsigned int size, BufferUsage usage=DefaultBufferUsage) noexcept
- **VertexBuffer** (VertexBuffer &&other) noexcept
- VertexBuffer & **operator=** (VertexBuffer &&other) noexcept
- **VertexBuffer** (const VertexBuffer &)=delete
- VertexBuffer & **operator=** (const VertexBuffer &)=delete
- void **Bind** () const noexcept
- void **Unbind** () const noexcept
- void **SetData** (const void *data, unsigned int size, BufferUsage usage=DefaultBufferUsage) const noexcept
- unsigned int **getID** () const noexcept

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

- engineOne/src/Core/Buffer.h
- engineOne/src/Core/Buffer.cpp

3.16 Window Class Reference

Public Member Functions

- **Window** (HINSTANCE hInstance, const std::string &windowClassName, const std::string &title, int width, int height, DWORD windowStyle)
- void **ProcessMessages** () noexcept
- bool **ShouldClose** () const noexcept
- HDC **GetDeviceContext** () const noexcept
- int **GetWidth** () const noexcept
- int **GetHeight** () const noexcept
- float **GetAspectRatio** () const noexcept
- void **Close** () noexcept
- bool **isCreated** () const noexcept
- **operator bool** () const noexcept

Static Public Member Functions

- static LRESULT CALLBACK **StaticWndProc** (HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) noexcept

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

- engineOne/src/Core/Window.h
- engineOne/src/Core/Window.cpp

Chapter 4

File Documentation

4.1 Application.h

```
00001 #pragma once
00002 #include"Window.h"
00003 #include <memory>
00004 #include "Buffer.h"
00005 #include "Shader.h"
00006 #include "VertexArray.h"
00007 #include "Camera.h"
00008 #include "Timer.h"
00009 #include "Texture.h"
00010 #include "RenderContext.h"
00011 #include "GLLoader.h"
00012
00013 struct Vertex
00014 {
00015     glm::vec3 position; // 3D position
00016     float u, v;     // Texture coordinates
00017 };
00018
00019 struct Mesh
00020 {
00021     std::vector<Vertex> vertices;
00022     std::vector<uint32_t> indices;
00023 };
00024
00028 class Application
00029 {
00030 public:
00036     Application(HINSTANCE hInstance, const std::string& appName) noexcept;
00037
00038     Application(const Application&) = delete;
00039     Application& operator=(const Application&) = delete;
00040
00041     virtual ~Application() noexcept = default;
00042
00047     bool Init() noexcept;
00051     void Run() noexcept;
00052 private:
00057     bool InitGraphics() noexcept;
00062     bool InitResources() noexcept;
00063
00068     void ProcessInput(float deltaTime) noexcept;
00069
00074     void Update(float deltaTime) noexcept;
00078     void Render() noexcept;
00079
00080 private:
00086     static bool RegisterWindowClass(HINSTANCE hInstance) noexcept;
00087
00099     static void APIENTRY OpenGLDebugCallback(
00100         GLenum source, GLenum type, GLuint id, GLenum severity,
00101         GLsizei length, const GLchar* message, const void* userParam);
00102 private:
00103     inline static constexpr const char* s_WindowClassName = "MyEngineWindowClass";
00104 private:
00105     //in the order of initialization and reverse order of destruction
00106     HINSTANCE m_hInstance;
00107     std::string m_ApplicationName;
```

```
00108     CameraFPS m_Camera;
00109     Timer m_Timer;
00110
00111     GLLoader m_OpenGLLoader;
00112     std::unique_ptr<Window> m_Window;
00113     std::unique_ptr<RenderContext> m_RenderContext;
00114     std::unique_ptr<ShaderProgram> m_ShaderProgram;
00115
00116     Mesh m_Mesh;
00117
00118     std::unique_ptr<VertexArray> m_VAO;
00119     std::unique_ptr<VertexBuffer> m_VBO;
00120     std::unique_ptr<IndexBuffer> m_EBO;
00121     std::unique_ptr<Texture2D> m_Texture,m_AlphaTexture;
00122 };
00123
```

4.2 Buffer.h

```
00001 #pragma once
00002 #include <glad/gl.h>
00003 #include "utils.h"
00004 enum class BufferUsage : GLenum
00005 {
00006     Static  =  GL_STATIC_DRAW,
00007     Dynamic = GL_DYNAMIC_DRAW,
00008     Stream  = GL_STREAM_DRAW
00009 };
00010
00011 //set default value to Static
00012
00013 constexpr BufferUsage DefaultBufferUsage = BufferUsage::Static;
00014
00015
00016 constexpr GLenum BufferUsageToGLenum(BufferUsage usage) noexcept
00017 {
00018     return enumValue(usage);
00019 }
00020
00021 class VertexBuffer
00022 {
00023 public:
00024     VertexBuffer() noexcept;
00025     VertexBuffer(const void* data, unsigned int size, BufferUsage usage = DefaultBufferUsage)
      noexcept;
00026     //move constructor
00027     VertexBuffer(VertexBuffer&& other) noexcept;
00028     ~VertexBuffer();
00029
00030     //move assignment operator
00031     VertexBuffer& operator=(VertexBuffer&& other) noexcept;
00032     // delete copy  constructor and copy assignment operator
00033     VertexBuffer(const VertexBuffer&) = delete;
00034     VertexBuffer& operator=(const VertexBuffer&) = delete;
00035
00036     void Bind() const noexcept;
00037     void Unbind() const noexcept;
00038     void SetData(const void* data, unsigned int size, BufferUsage usage = DefaultBufferUsage) const
      noexcept;
00039
00040     inline unsigned int getID() const noexcept { return m_ID; }
00041
00042 private:
00043     unsigned int m_ID;
00044 };
00045
00046
00047 class IndexBuffer
00048 {
00049 public:
00050     IndexBuffer() noexcept;
00051     IndexBuffer(const void* data, unsigned int size, BufferUsage usage = DefaultBufferUsage) noexcept;
00052     //move constructor
00053     IndexBuffer(IndexBuffer&& other) noexcept;
00054     ~IndexBuffer();
00055
00056     //move assignment operator
00057     IndexBuffer& operator=(IndexBuffer&& other) noexcept;
00058     // delete copy  constructor and copy assignment operator
00059     IndexBuffer(const IndexBuffer&) = delete;
00060     IndexBuffer& operator=(const IndexBuffer&) = delete;
00061
00062     void Bind() const noexcept;
```

```
00063     void Unbind() const noexcept;
00064     void SetData(const void* data, unsigned int size, BufferUsage usage = DefaultBufferUsage) const
      noexcept;
00065
00066     inline unsigned int getID() const noexcept { return m_ID; }
00067
00068 private:
00069     unsigned int m_ID;
00070 };
00071
```

4.3 Camera.h

```
00001 #pragma once
00002 #include<glm/glm.hpp>
00003
00004 class CameraFPS
00005 {
00006 public:
00007
00008     CameraFPS(float fov = 60.0f, float aspect = 4.0f/3.0f, float zNear = 0.1f, float zFar = 100.0f)
      noexcept;
00009
00010     void MoveTo(const glm::vec3& position) noexcept;
00011     void Translate(const glm::vec3& displacement) noexcept;
00012
00013     void Yaw(float angle) noexcept;   // rotate around the up vector
00014     void Pitch(float angle) noexcept; // rotate around the right vector
00015     void Roll(float angle) noexcept;  // rotate around the front vector
00016
00017     void MoveForward(float distance) noexcept;
00018     void MoveRight(float distance) noexcept;
00019     void MoveUp(float distance) noexcept;
00020
00021     void UpdateProjection(float fov, float aspect, float zNear, float zFar) noexcept;
00022
00023     const glm::mat4& GetViewMatrix() const noexcept { return m_View; }
00024     const glm::mat4& GetProjectionMatrix() const noexcept { return m_Projection; }
00025
00026     void RestPosAndOrient() noexcept;
00027 private:
00028     void RecalculateView() noexcept;
00029 private:
00030     glm::mat4 m_View, m_Projection;
00031     glm::vec3 m_Position;
00032     glm::vec3 m_Front;
00033     glm::vec3 m_Up;
00034     glm::vec3 m_Right;
00035
00036 };
00037
```

4.4 GLLoader.h

```
00001 #pragma once
00002
00003 #include<Windows.h>
00004
00005 bool loadModernOpenGLFunctions(HINSTANCE hInstance);
00006 void unloadModernOpenGLFunctions();
00007
00008 class GLLoader
00009 {
00010 public:
00011     inline GLLoader(HINSTANCE hInstance) noexcept
00012         :
00013         m_IsLoaded(loadModernOpenGLFunctions(hInstance))
00014     {}
00015
00016     ~GLLoader() noexcept { unloadModernOpenGLFunctions(); }
00017
00018     GLLoader(const GLLoader&) = delete;
00019     GLLoader& operator=(const GLLoader&) = delete;
00020     GLLoader(GLLoader&&) = delete;
00021     GLLoader& operator=(GLLoader&&) = delete;
00022
00023     bool isLoaded() const noexcept { return m_IsLoaded; }
00024 private:
```

```
00025     bool m_IsLoaded;
00026 };
00027
00028
00029
```

4.5 glTypes.h

```
00001 #pragma once
00002 #include<glad/gl.h>
00003 #include "utils.h"
00004
00005 enum class GLType : GLenum
00006 {
00007     Float = GL_FLOAT,
00008     Double = GL_DOUBLE,
00009     Int = GL_INT,
00010     UnsignedInt = GL_UNSIGNED_INT,
00011     Short = GL_SHORT,
00012     UnsignedShort = GL_UNSIGNED_SHORT,
00013     Byte = GL_BYTE,
00014     UnsignedByte = GL_UNSIGNED_BYTE
00015 };
00016
00017
00018 constexpr GLenum GLTypeToGLenum(GLType type) noexcept
00019 {
00020     return enumValue(type);
00021 }
00022
00023
00024
00025 inline GLboolean boolToGLboolean(bool value) noexcept
00026 {
00027     return static_cast<GLboolean>(value);
00028 }
00029
00030 inline bool GLbooleanTobool(GLboolean value) noexcept
00031 {
00032     return  static_cast<bool>(value);
00033 }
```

4.6 glUtils.h

```
00001 #pragma once
00002 #include<glad/gl.h>
00003
00004 inline void SafeDeleteGLBuffer(GLuint& id)
00005 {
00006     if (id != 0)
00007     {
00008         glDeleteBuffers(1, &id);
00009         id = 0;
00010     }
00011 }
00012
00013 inline void SafeDeleteGLProgram(GLuint& id)
00014 {
00015     if (id != 0)
00016     {
00017         glDeleteProgram(id);
00018         id = 0;
00019     }
00020 }
00021
00022 inline void SafeDeleteGLShader(GLuint& id)
00023 {
00024     if (id != 0)
00025     {
00026         glDeleteShader(id);
00027         id = 0;
00028     }
00029 }
00030
00031 inline void SafeDeleteGLTexture(GLuint& id)
00032 {
00033     if (id != 0)
00034     {
```

```
00035         glDeleteTextures(1, &id);
00036         id = 0;
00037     }
00038 }
00039
00040 inline void SafeDeleteGLVertexArray(GLuint& id)
00041 {
00042     if (id != 0)
00043     {
00044         glDeleteVertexArrays(1, &id);
00045         id = 0;
00046     }
00047 }
00048
```

4.7 Input.h

```
00001 #pragma once
00002 #include<Windows.h>
00003 #include<winuser.h>
00004
00005 inline bool isKeyPressed(int vKey)
00006 {
00007     return (GetAsyncKeyState(vKey) & (short)0x8000) != 0;
00008 }
```

4.8 keyCodes.h

```
00001 #pragma once
00002
00003 namespace keyCode
00004 {
00005     // Mouse buttons
00006     constexpr int mouse_left = 0x01;
00007     constexpr int mouse_right = 0x02;
00008     constexpr int mouse_middle = 0x04;
00009     constexpr int mouse_x1 = 0x05;
00010     constexpr int mouse_x2 = 0x06;
00011
00012     // Keyboard keys
00013     constexpr int key_backspace = 0x08;
00014     constexpr int key_tab = 0x09;
00015     constexpr int key_clear = 0x0C;
00016     constexpr int key_enter = 0x0D;
00017     constexpr int key_shift = 0x10;
00018     constexpr int key_ctrl = 0x11;
00019     constexpr int key_alt = 0x12;
00020     constexpr int key_pause = 0x13;
00021     constexpr int key_caps_lock = 0x14;
00022     constexpr int key_esc = 0x1B;
00023     constexpr int key_space = 0x20;
00024     constexpr int key_page_up = 0x21;
00025     constexpr int key_page_down = 0x22;
00026     constexpr int key_end = 0x23;
00027     constexpr int key_home = 0x24;
00028     constexpr int key_arrow_left = 0x25;
00029     constexpr int key_arrow_up = 0x26;
00030     constexpr int key_arrow_right = 0x27;
00031     constexpr int key_arrow_down = 0x28;
00032     constexpr int key_select = 0x29;
00033     constexpr int key_print = 0x2A;
00034     constexpr int key_execute = 0x2B;
00035     constexpr int key_print_screen = 0x2C;
00036     constexpr int key_insert = 0x2D;
00037     constexpr int key_delete = 0x2E;
00038     constexpr int key_help = 0x2F;
00039
00040     // Number keys 0-9
00041     constexpr int key_0 = 0x30;
00042     constexpr int key_1 = 0x31;
00043     constexpr int key_2 = 0x32;
00044     constexpr int key_3 = 0x33;
00045     constexpr int key_4 = 0x34;
00046     constexpr int key_5 = 0x35;
00047     constexpr int key_6 = 0x36;
00048     constexpr int key_7 = 0x37;
00049     constexpr int key_8 = 0x38;
00050     constexpr int key_9 = 0x39;
```

```
00051
00052     // Alphabet keys A-Z
00053     constexpr int key_a = 0x41;
00054     constexpr int key_b = 0x42;
00055     constexpr int key_c = 0x43;
00056     constexpr int key_d = 0x44;
00057     constexpr int key_e = 0x45;
00058     constexpr int key_f = 0x46;
00059     constexpr int key_g = 0x47;
00060     constexpr int key_h = 0x48;
00061     constexpr int key_i = 0x49;
00062     constexpr int key_j = 0x4A;
00063     constexpr int key_k = 0x4B;
00064     constexpr int key_l = 0x4C;
00065     constexpr int key_m = 0x4D;
00066     constexpr int key_n = 0x4E;
00067     constexpr int key_o = 0x4F;
00068     constexpr int key_p = 0x50;
00069     constexpr int key_q = 0x51;
00070     constexpr int key_r = 0x52;
00071     constexpr int key_s = 0x53;
00072     constexpr int key_t = 0x54;
00073     constexpr int key_u = 0x55;
00074     constexpr int key_v = 0x56;
00075     constexpr int key_w = 0x57;
00076     constexpr int key_x = 0x58;
00077     constexpr int key_y = 0x59;
00078     constexpr int key_z = 0x5A;
00079
00080     // Numpad keys
00081     constexpr int key_numpad_0 = 0x60;
00082     constexpr int key_numpad_1 = 0x61;
00083     constexpr int key_numpad_2 = 0x62;
00084     constexpr int key_numpad_3 = 0x63;
00085     constexpr int key_numpad_4 = 0x64;
00086     constexpr int key_numpad_5 = 0x65;
00087     constexpr int key_numpad_6 = 0x66;
00088     constexpr int key_numpad_7 = 0x67;
00089     constexpr int key_numpad_8 = 0x68;
00090     constexpr int key_numpad_9 = 0x69;
00091     constexpr int key_multiply = 0x6A;
00092     constexpr int key_add = 0x6B;
00093     constexpr int key_separator = 0x6C;
00094     constexpr int key_subtract = 0x6D;
00095     constexpr int key_decimal = 0x6E;
00096     constexpr int key_divide = 0x6F;
00097
00098     // Function keys
00099     constexpr int key_f1 = 0x70;
00100     constexpr int key_f2 = 0x71;
00101     constexpr int key_f3 = 0x72;
00102     constexpr int key_f4 = 0x73;
00103     constexpr int key_f5 = 0x74;
00104     constexpr int key_f6 = 0x75;
00105     constexpr int key_f7 = 0x76;
00106     constexpr int key_f8 = 0x77;
00107     constexpr int key_f9 = 0x78;
00108     constexpr int key_f10 = 0x79;
00109     constexpr int key_f11 = 0x7A;
00110     constexpr int key_f12 = 0x7B;
00111     constexpr int key_f13 = 0x7C;
00112     constexpr int key_f14 = 0x7D;
00113     constexpr int key_f15 = 0x7E;
00114     constexpr int key_f16 = 0x7F;
00115     constexpr int key_f17 = 0x80;
00116     constexpr int key_f18 = 0x81;
00117     constexpr int key_f19 = 0x82;
00118     constexpr int key_f20 = 0x83;
00119     constexpr int key_f21 = 0x84;
00120     constexpr int key_f22 = 0x85;
00121     constexpr int key_f23 = 0x86;
00122     constexpr int key_f24 = 0x87;
00123
00124     // Other keys
00125     constexpr int key_num_lock = 0x90;
00126     constexpr int key_scroll_lock = 0x91;
00127     constexpr int key_left_shift = 0xA0;
00128     constexpr int key_right_shift = 0xA1;
00129     constexpr int key_left_ctrl = 0xA2;
00130     constexpr int key_right_ctrl = 0xA3;
00131     constexpr int key_left_alt = 0xA4;
00132     constexpr int key_right_alt = 0xA5;
00133     constexpr int key_semicolon = 0xBA;   // ;: key
00134     constexpr int key_plus = 0xBB;        // =+ key
00135     constexpr int key_comma = 0xBC;       // ,< key
00136     constexpr int key_minus = 0xBD;       // -_ key
00137     constexpr int key_period = 0xBE;      // .> key
```

```
00138     constexpr int key_slash = 0xBF;       // /? key
00139     constexpr int key_grave = 0xC0;       // `~ key
00140     constexpr int key_bracket_left = 0xDB;  // [{ key
00141     constexpr int key_backslash = 0xDC;   // \| key
00142     constexpr int key_bracket_right = 0xDD; // ]} key
00143     constexpr int key_quote = 0xDE;       // '" key
00144     constexpr int key_oem_8 = 0xDF;
00145     constexpr int key_process = 0xE5;
00146     constexpr int key_packet = 0xE7;
00147     constexpr int key_attn = 0xF6;
00148     constexpr int key_crsel = 0xF7;
00149     constexpr int key_exsel = 0xF8;
00150     constexpr int key_ereof = 0xF9;
00151     constexpr int key_play = 0xFA;
00152     constexpr int key_zoom = 0xFB;
00153     constexpr int key_noname = 0xFC;
00154     constexpr int key_pa1 = 0xFD;
00155     constexpr int key_oem_clear = 0xFE;
00156
00157 } // namespace keyCode
```

4.9 RenderContext.h

```
00001 #pragma once
00002 #include "Window.h"
00003 #include "utils.h"
00004
00005 #include<glad/gl.h>
00006
00007 struct RenderContextCreateInfo
00008 {
00009     Window& window;
00010     int majorVersion = 4;
00011     int minorVersion = 5;
00012     bool debugContext = true;
00013     int colorBits = 32;
00014     int depthBits = 24;
00015     int stencilBits = 8;
00016     int samples = 0; // Number of samples for multisampling (0 = no multisampling)
00017     RenderContextCreateInfo(Window& win,
00018         int maj = 4,
00019         int min = 5,
00020         bool debug = true,
00021         int color = 32,
00022         int depth = 24,
00023         int stencil = 8,
00024         int sampleCount = 0)
00025         : window(win),
00026         majorVersion(maj),
00027         minorVersion(min),
00028         debugContext(debug),
00029         colorBits(color),
00030         depthBits(depth),
00031         stencilBits(stencil),
00032         samples(sampleCount)
00033     {}
00034 };
00035
00036
00037 class RenderContext
00038 {
00039 public:
00040     RenderContext(const RenderContextCreateInfo& createInfo);
00041     RenderContext(Window& window,int majorVersion = 4, int minorVersion = 5, bool debug = true);
00042     ~RenderContext() noexcept;
00043
00044     RenderContext(const RenderContext&) = delete;
00045     RenderContext& operator=(const RenderContext&) = delete;
00046
00047     bool MakeCurrent(Window& window) noexcept;
00048
00049     inline void clearColor(float r, float g, float b, float a) noexcept
00050     {
00051         glClearColor(r, g, b, a);
00052         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);
00053     }
00054
00055     inline void Present() noexcept
00056     {
00057         SwapBuffers(m_hDC);
00058     }
00059
00060     inline bool IsNull() const noexcept { return m_hGLRC == nullptr; }
```

```
00061     inline operator bool() const noexcept { return m_hGLRC != nullptr; }
00062 private:
00063     HDC m_hDC;
00064     HGLRC m_hGLRC;
00065 };
00066
```

4.10 Shader.h

```
00001 #pragma once
00002 #include<string>
00003 #include<unordered_map>
00004
00005 #include<glad/gl.h>
00006 #include<glm/glm.hpp>
00007 #include "utils.h"
00008
00009 enum class ShaderType : GLenum
00010 {
00011     VERTEX = GL_VERTEX_SHADER,
00012     FRAGMENT = GL_FRAGMENT_SHADER,
00013     GEOMETRY = GL_GEOMETRY_SHADER,
00014     TESS_CONTROL = GL_TESS_CONTROL_SHADER,
00015     TESS_EVALUATION = GL_TESS_EVALUATION_SHADER,
00016     COMPUTE = GL_COMPUTE_SHADER
00017 };
00018
00019 inline std::string shaderTypeToString(ShaderType type)
00020 {
00021     switch (type)
00022     {
00023     case ShaderType::VERTEX: return "VERTEX";
00024     case ShaderType::FRAGMENT: return "FRAGMENT";
00025     case ShaderType::GEOMETRY: return "GEOMETRY";
00026     case ShaderType::TESS_CONTROL: return "TESS_CONTROL";
00027     case ShaderType::TESS_EVALUATION: return "TESS_EVALUATION";
00028     case ShaderType::COMPUTE: return "COMPUTE";
00029     default: return "UNKNOWN_SHADER_TYPE";
00030     }
00031 }
00032
00033 constexpr GLenum shaderTypeToGLenum(ShaderType type)
00034 {
00035     return enumValue(type);
00036 }
00037
00038 enum class ShaderLoadOption
00039 {
00040     String,
00041     File
00042 };
00043 class Shader
00044 {
00045 public:
00046     Shader(ShaderType type) noexcept;
00047     Shader(ShaderType type, std::string& data, ShaderLoadOption loadOption);
00048     Shader(Shader&&) noexcept;
00049
00050     ~Shader() noexcept;
00051
00052     // Prevent copying
00053     Shader(const Shader&) = delete;
00054     Shader& operator=(const Shader&) = delete;
00055
00056
00057     Shader& operator=(Shader&&) noexcept;
00058
00059     bool loadFromFile(const std::string& filePath);
00060     bool loadFromString(const std::string& shaderSrc);
00061
00062     inline ShaderType getType() const { return m_Type; }
00063     inline unsigned int getID() const { return m_ID; }
00064 private:
00065     bool checkCompileStatus();
00066 private:
00067     unsigned int m_ID;
00068     ShaderType m_Type;
00069 };
00070
00071
00072 class ShaderProgram
00073 {
00074 public:
```

```
00075     ShaderProgram() noexcept;
00076     ShaderProgram(ShaderProgram&&)noexcept;
00077     ShaderProgram(const std::string& vertexPath, const std::string& fragmentPath);
00078     ShaderProgram(Shader& vertexShader, Shader& fragmentShader);
00079 
00080     ~ShaderProgram() noexcept;
00081 
00082     // Prevent copying
00083     ShaderProgram(const ShaderProgram&) = delete;
00084     ShaderProgram& operator=(const ShaderProgram&) = delete;
00085 
00086 
00087     ShaderProgram& operator=(ShaderProgram&&)noexcept;
00088 
00089     void attachShader(const Shader& shader);
00090     bool link();
00091     void Bind() const;
00092     void Unbind() const;
00093 
00094     inline unsigned int getID() const { return m_ID; }
00095     inline bool isValid() const { return m_ID != 0; }
00096 
00097 
00098     bool checkLinkStatus();
00099 
00100     void SetUniform1i(const std::string& uniformName, int v) noexcept;
00101 
00102 
00103     void SetUniform1f(const std::string& uniformName, float v) noexcept;
00104     void SetUniform2f(const std::string& uniformName, float v0,float v1) noexcept;
00105     void SetUniform3f(const std::string& uniformName, float v0,float v1, float v2) noexcept;
00106     void SetUniform4f(const std::string& uniformName, float v0,float v1, float v2,float v3) noexcept;
00107 
00108     void SetUniformVec2(const std::string& uniformName, const glm::vec2& vec) noexcept;
00109     void SetUniformVec3(const std::string& uniformName, const glm::vec3& vec) noexcept;
00110     void SetUniformVec4(const std::string& uniformName, const glm::vec4& vec) noexcept;
00111     // specify array size
00112     void SetUniformMat2(const std::string& uniformName, const glm::mat2& mat) noexcept;
00113     void SetUniformMat3(const std::string& uniformName, const glm::mat3& mat) noexcept;
00114     void SetUniformMat4(const std::string& uniformName, const glm::mat4& mat) noexcept;
00115 private:
00116     unsigned int GetUniformLocation(const std::string& uniformName) noexcept;
00117 private:
00118     unsigned int m_ID;
00119     std::unordered_map<std::string, int> m_UniformCache;
00120 };
00121 
00122 
```

4.11 Texture.h

```
00001 #pragma once
00002 #include<glad/gl.h>
00003 #include "glTypes.h"
00004 #include "utils.h"
00005 
00006 // Simplified format categories
00007 enum class TextureInternalFormat : GLenum
00008 {
00009     // Basic formats
00010     R8 = GL_R8,
00011     RG8 = GL_RG8,
00012     RGB8 = GL_RGB8,
00013     RGBA8 = GL_RGBA8,
00014 
00015     // Float formats
00016     R16F = GL_R16F,
00017     RG16F = GL_RG16F,
00018     RGB16F = GL_RGB16F,
00019     RGBA16F = GL_RGBA16F,
00020 
00021     // Depth formats
00022     DEPTH24 = GL_DEPTH_COMPONENT24,
00023     DEPTH32F = GL_DEPTH_COMPONENT32F,
00024     DEPTH24_STENCIL8 = GL_DEPTH24_STENCIL8
00025 };
00026 
00027 enum class TextureBaseFormat : GLenum
00028 {
00029     RED = GL_RED,
00030     RG = GL_RG,
00031     RGB = GL_RGB,
00032     RGBA = GL_RGBA,
```

```
00033     DEPTH_COMPONENT = GL_DEPTH_COMPONENT,
00034     DEPTH_STENCIL = GL_DEPTH_STENCIL
00035 };
00036
00037 enum class PixelDataType : GLenum
00038 {
00039     UNSIGNED_BYTE = GL_UNSIGNED_BYTE,
00040     BYTE = GL_BYTE,
00041     UNSIGNED_SHORT = GL_UNSIGNED_SHORT,
00042     SHORT = GL_SHORT,
00043     UNSIGNED_INT = GL_UNSIGNED_INT,
00044     INT = GL_INT,
00045     FLOAT = GL_FLOAT
00046 };
00047
00048 inline constexpr GLenum TextureInternalFormatToGLenum(TextureInternalFormat format) noexcept
00049 {
00050     return enumValue(format);
00051 }
00052
00053 inline constexpr GLenum TextureBaseFormatToGLenum(TextureBaseFormat format) noexcept
00054 {
00055     return enumValue(format);
00056 }
00057
00058 enum class pixelDataType : GLenum
00059 {
00060     UNSIGNEDBYTE = GL_UNSIGNED_BYTE,
00061     BYTE = GL_BYTE,
00062     UNSIGNEDSHORT = GL_UNSIGNED_SHORT,
00063     SHORT = GL_SHORT,
00064     UNSIGNEDINT = GL_UNSIGNED_INT,
00065     INT = GL_INT,
00066     FLOAT = GL_FLOAT,
00067     UNSIGNEDBYTE332 = GL_UNSIGNED_BYTE_3_3_2,
00068     UNSIGNEDBYTE233REV = GL_UNSIGNED_BYTE_2_3_3_REV,
00069     UNSIGNEDSHORT565 = GL_UNSIGNED_SHORT_5_6_5,
00070     UNSIGNEDSHORT565REV = GL_UNSIGNED_SHORT_5_6_5_REV,
00071     UNSIGNEDSHORT4444 = GL_UNSIGNED_SHORT_4_4_4_4,
00072     UNSIGNEDSHORT4444REV = GL_UNSIGNED_SHORT_4_4_4_4_REV,
00073     UNSIGNEDSHORT5551 = GL_UNSIGNED_SHORT_5_5_5_1,
00074     UNSIGNEDSHORT1555REV = GL_UNSIGNED_SHORT_1_5_5_5_REV,
00075     UNSIGNEDINT8888 = GL_UNSIGNED_INT_8_8_8_8,
00076     UNSIGNEDINT8888REV = GL_UNSIGNED_INT_8_8_8_8_REV,
00077     UNSIGNEDINT1010102 = GL_UNSIGNED_INT_10_10_10_2,
00078     UNSIGNEDINT2101010REV = GL_UNSIGNED_INT_2_10_10_10_REV
00079
00080 };
00081
00082 inline constexpr GLenum PixelDataTypeToGLenum(pixelDataType format) noexcept
00083 {
00084     return enumValue(format);
00085 }
00086
00087
00088 struct Texture2DDataCreateInfo
00089 {
00090     int width, height;
00091     void* data;
00092     pixelDataType type;
00093     TextureInternalFormat internalFormat;
00094     int mipLevels = 1;
00095 };
00096 class Texture2D
00097 {
00098 public:
00099     Texture2D(int width, int height, void* data, pixelDataType type, TextureInternalFormat
      internalFormat, int mipLevels = 1) noexcept;
00100     Texture2D(const Texture2DDataCreateInfo& createInfo) noexcept;
00101     ~Texture2D() noexcept;
00102
00103     //delete copy constructor and copy assignment operator
00104     Texture2D(const Texture2D&) = delete;
00105     Texture2D& operator=(const Texture2D&) = delete;
00106     //move constructor
00107     Texture2D(Texture2D&& other) noexcept;
00108     //move assignment operator
00109     Texture2D& operator=(Texture2D&& other) noexcept;
00110
00111     void SubImage(int width, int height, const void* data, pixelDataType type, TextureBaseFormat
      baseFormat, int level = 0, int xOfffset = 0, int yOffset = 0) const noexcept;
00112
00113     void Bind(unsigned int slot = 0) const noexcept;
00114     void Unbind(unsigned int slot = 0) const noexcept;
00115
00116 private:
00117     unsigned int m_ID;
```

```
00118 };
00119
```

4.12 Timer.h

```
00001 #pragma once
00002 #include <Windows.h>
00003
00004 class Timer
00005 {
00006 public:
00007     Timer() noexcept
00008         :
00009         m_Frequency{},
00010         m_Time{}
00011     {
00012         QueryPerformanceFrequency(&m_Frequency);
00013         QueryPerformanceCounter(&m_Time);
00014     }
00015     Timer(const Timer&) noexcept = delete;
00016     Timer& operator=(const Timer&) noexcept = delete;
00017
00018     Timer(Timer&&) noexcept = default;
00019     Timer& operator=(Timer&&) noexcept = default;
00020
00021     inline void reset()
00022     {
00023         QueryPerformanceCounter(&m_Time);
00024     }
00025
00026     inline void resetFrequency() noexcept
00027     {
00028         QueryPerformanceFrequency(&m_Frequency);
00029     }
00030
00031
00032     inline double elapsed() const noexcept
00033     {
00034         LARGE_INTEGER currentTime;
00035         QueryPerformanceCounter(&currentTime);
00036         return static_cast<double>(currentTime.QuadPart - m_Time.QuadPart) /
      static_cast<double>(m_Frequency.QuadPart);
00037     }
00038     inline double elapsedAndReset() noexcept
00039     {
00040         LARGE_INTEGER currentTime;
00041         QueryPerformanceCounter(&currentTime);
00042         double dt = static_cast<double>(currentTime.QuadPart - m_Time.QuadPart) /
      static_cast<double>(m_Frequency.QuadPart);
00043         m_Time = currentTime; // reset after computing elapsed
00044         return dt;
00045     }
00046
00047
00048
00049
00050 private:
00051     LARGE_INTEGER m_Frequency;
00052     LARGE_INTEGER m_Time;
00053 };
00054
```

4.13 utils.h

```
00001 #pragma once
00002 #include <type_traits>
00003 template <typename E>
00004 constexpr auto enumValue(E e) noexcept {
00005     return static_cast<std::underlying_type_t<E>>(e);
00006 }
```

4.14 VertexArray.h

```
00001 #pragma once
```

```
00002 #include "glTypes.h"
00003 class VertexArray
00004 {
00005 public:
00006     VertexArray() noexcept;
00007     //move constructor
00008     VertexArray(VertexArray&& other) noexcept;
00009 
00010     ~VertexArray();
00011 
00012     //move assignment operator
00013     VertexArray& operator=(VertexArray&& other) noexcept;
00014 
00015     // delete copy  constructor and copy assignment operator
00016     VertexArray(const VertexArray&) = delete;
00017     VertexArray& operator=(const VertexArray&) = delete;
00018 
00019     void Bind() const noexcept;
00020     void Unbind() const noexcept;
00021 
00022     void addAttribute(unsigned int index, int size, GLType type, bool normalized, int stride, const
      void* pointer) const noexcept;
00023 
00024     inline unsigned int getID() const noexcept { return m_ID; }
00025 private:
00026     unsigned int m_ID;
00027 };
00028 
```

4.15 Window.h

```
00001 // Window.h
00002 #pragma once
00003 #include <Windows.h>
00004 #include <string>
00005 
00006 class Window
00007 {
00008 public:
00009     Window(HINSTANCE hInstance, const std::string& windowClassName, const std::string& title, int
      width, int height, DWORD windowStyle);
00010     ~Window() noexcept;
00011 
00012     void ProcessMessages() noexcept;
00013     bool ShouldClose() const noexcept { return m_ShouldClose; }
00014 
00015     HDC GetDeviceContext() const noexcept  { return GetDC(m_hWnd); }
00016     int GetWidth() const noexcept  { return m_Width; }
00017     int GetHeight() const noexcept  { return mHeight; }
00018     float GetAspectRatio() const noexcept { return m_AspectRatio; }
00019 
00020     void Close() noexcept { m_ShouldClose = true; }
00021 
00022     bool isCreated() const noexcept { return m_IsCreated; }
00023     explicit operator bool() const noexcept { return m_IsCreated; }
00024     static LRESULT CALLBACK StaticWndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) noexcept;
00025 private:
00026     LRESULT HandleMsg(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) noexcept;
00027 
00028     static bool DoesWindowClassExistsAndHavePrivateDCAndUsesCorrectWndProc(const std::string&
      className, HINSTANCE hInstance ) noexcept;
00029 private:
00030 
00031 
00032     HINSTANCE m_hInstance;
00033     HWND m_hWnd;
00034     std::string windowTitle;
00035     int m_Width, mHeight;
00036     float m_AspectRatio;
00037     bool m_ShouldClose;
00038     bool m_IsCreated;
00039 };
```

Index

Application, 5
    Application, 5
    Init, 6

CameraFPS, 6

engineOne/src/Core/Application.h, 13
engineOne/src/Core/Buffer.h, 14
engineOne/src/Core/Camera.h, 15
engineOne/src/Core/GLLoader.h, 15
engineOne/src/Core/glTypes.h, 16
engineOne/src/Core/glUtils.h, 16
engineOne/src/Core/Input.h, 17
engineOne/src/Core/keyCodes.h, 17
engineOne/src/Core/RenderContext.h, 19
engineOne/src/Core/Shader.h, 20
engineOne/src/Core/Texture.h, 21
engineOne/src/Core/Timer.h, 23
engineOne/src/Core/utils.h, 23
engineOne/src/Core/VertexArray.h, 23
engineOne/src/Core/Window.h, 24

GLLoader, 7

IndexBuffer, 7
Init
    Application, 6

Mesh, 7

RenderContext, 8
RenderContextCreateInfo, 8

Shader, 9
ShaderProgram, 9

Texture2D, 10
Texture2DDataCreateInfo, 10
Timer, 11

Vertex, 11
VertexArray, 11
VertexBuffer, 12

Window, 12