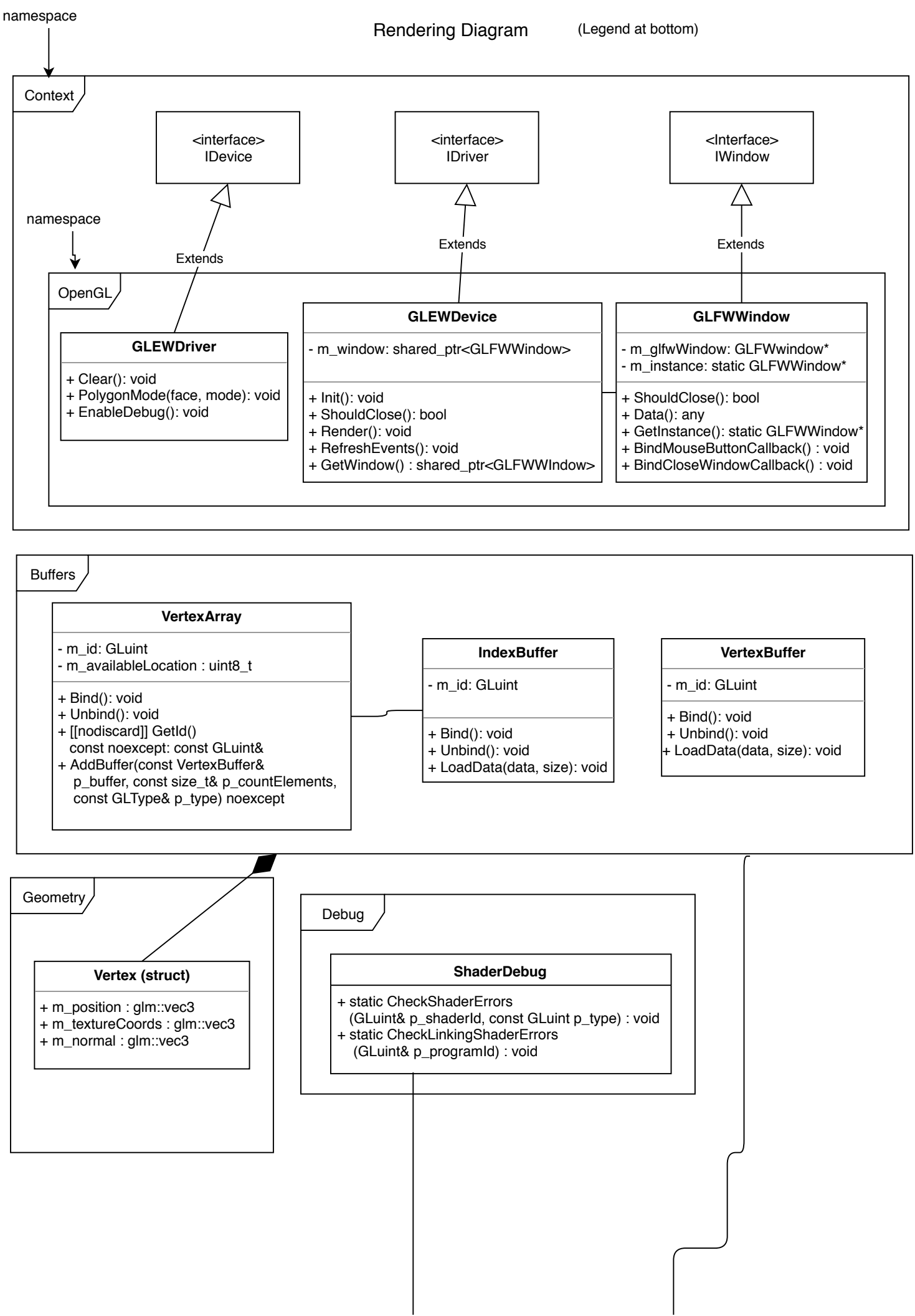
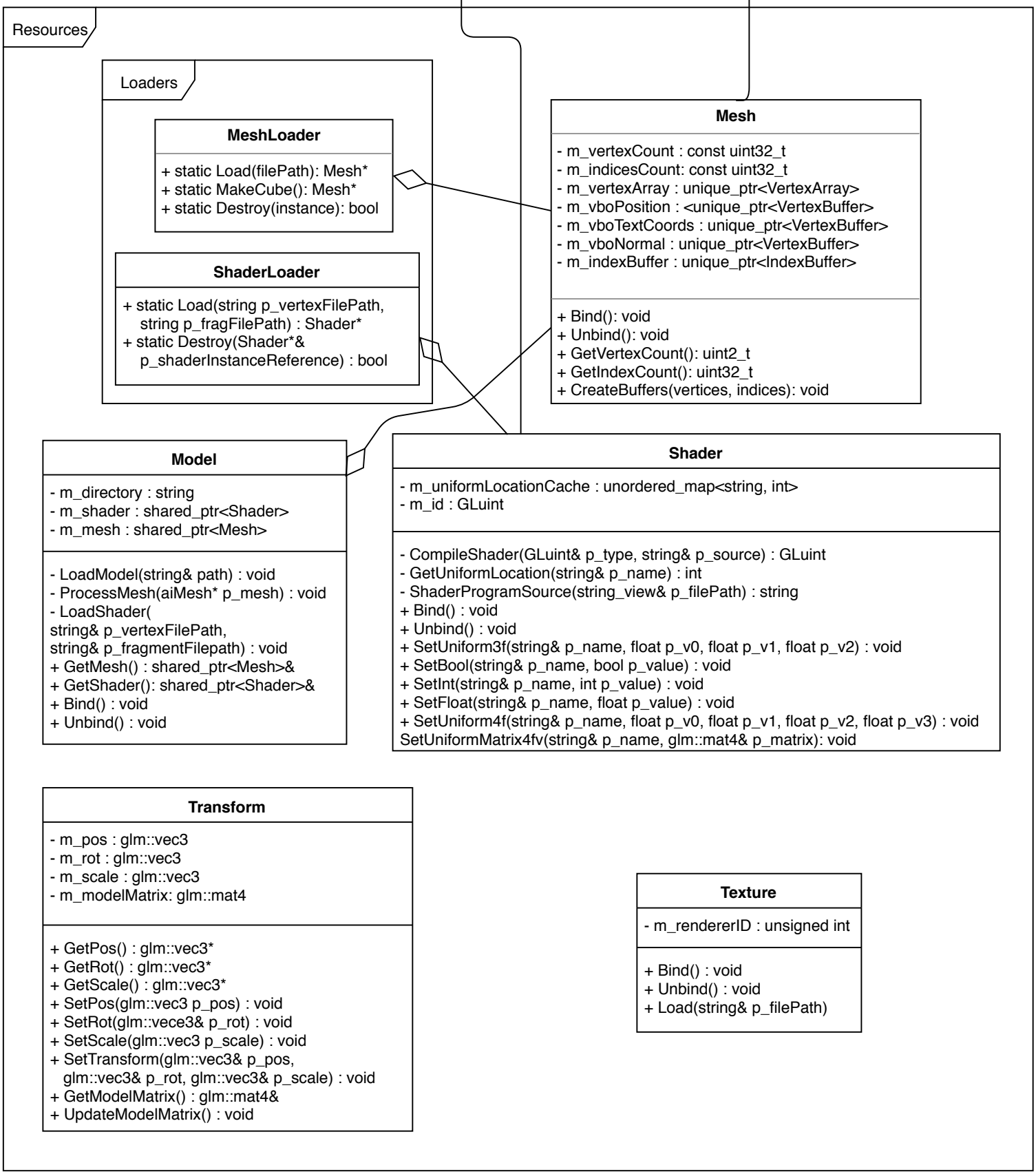

namespace
Rendering Diagram
(Legend at bottom)
Context
<interface>
IDevice
<interface>
IDriver
<Interface>
IWindow
namespace
Extends
Extends
Extends
OpenGL
GLEWDriver
+ Clear(): void
+ PolygonMode(face, mode): void
+ EnableDebug(): void
GLEWDevice
- m_window: shared_ptr<GLFWWindow>
+ Init(): void
+ ShouldClose(): bool
+ Render(): void
+ RefreshEvents(): void
+ GetWindow() : shared_ptr<GLFWWIndow>
GLFWWindow
- m_glfwWindow: GLFWwindow*
- m_instance: static GLFWWindow*
+ ShouldClose(): bool
+ Data(): any
+ GetInstance(): static GLFWWindow*
+ BindMouseButtonCallback() : void
+ BindCloseWindowCallback() : void
Buffers
VertexArray
- m_id: GLuint
- m_availableLocation : uint8_t
+ Bind(): void
+ Unbind(): void
+ [[nodiscard]] GetId()
const noexcept: const GLuint&
+ AddBuffer(const VertexBuffer&
p_buffer, const size_t& p_countElements,
const GLType& p_type) noexcept
IndexBuffer
- m_id: GLuint
+ Bind(): void
+ Unbind(): void
+ LoadData(data, size): void
VertexBuffer
- m_id: GLuint
+ Bind(): void
+ Unbind(): void
+ LoadData(data, size): void
Geometry
Vertex (struct)
+ m_position : glm::vec3
+ m_textureCoords : glm::vec3
+ m_normal : glm::vec3
Debug
ShaderDebug
+ static CheckShaderErrors
(GLuint& p_shaderId, const GLuint p_type) : void
+ static CheckLinkingShaderErrors
(GLuint& p_programId) : void


Resources
Loaders
MeshLoader
+ static Load(filePath): Mesh*
+ static MakeCube(): Mesh*
+ static Destroy(instance): bool
ShaderLoader
+ static Load(string p_vertexFilePath,
string p_fragFilePath) : Shader*
+ static Destroy(Shader*&
p_shaderInstanceReference) : bool
Mesh
- m_vertexCount : const uint32_t
- m_indicesCount: const uint32_t
- m_vertexArray : unique_ptr<VertexArray>
- m_vboPosition : <unique_ptr<VertexBuffer>
- m_vboTextCoords : unique_ptr<VertexBuffer>
- m_vboNormal : unique_ptr<VertexBuffer>
- m_indexBuffer : unique_ptr<IndexBuffer>
+ Bind(): void
+ Unbind(): void
+ GetVertexCount(): uint2_t
+ GetIndexCount(): uint32_t
+ CreateBuffers(vertices, indices): void
Model
- m_directory : string
- m_shader : shared_ptr<Shader>
- m_mesh : shared_ptr<Mesh>
- LoadModel(string& path) : void
- ProcessMesh(aiMesh* p_mesh) : void
- LoadShader(
string& p_vertexFilePath,
string& p_fragmentFilepath) : void
+ GetMesh() : shared_ptr<Mesh>&
+ GetShader(): shared_ptr<Shader>&
+ Bind() : void
+ Unbind() : void
Shader
- m_uniformLocationCache : unordered_map<string, int>
- m_id : GLuint
- CompileShader(GLuint& p_type, string& p_source) : GLuint
- GetUniformLocation(string& p_name) : int
- ShaderProgramSource(string_view& p_filePath) : string
+ Bind() : void
+ Unbind() : void
+ SetUniform3f(string& p_name, float p_v0, float p_v1, float p_v2) : void
+ SetBool(string& p_name, bool p_value) : void
+ SetInt(string& p_name, int p_value) : void
+ SetFloat(string& p_name, float p_value) : void
+ SetUniform4f(string& p_name, float p_v0, float p_v1, float p_v2, float p_v3) : void
SetUniformMatrix4fv(string& p_name, glm::mat4& p_matrix): void
Transform
- m_pos : glm::vec3
- m_rot : glm::vec3
- m_scale : glm::vec3
- m_modelMatrix: glm::mat4
+ GetPos() : glm::vec3*
+ GetRot() : glm::vec3*
+ GetScale() : glm::vec3*
+ SetPos(glm::vec3 p_pos) : void
+ SetRot(glm::vece3& p_rot) : void
+ SetScale(glm::vec3 p_scale) : void
+ SetTransform(glm::vec3& p_pos,
glm::vec3& p_rot, glm::vec3& p_scale) : void
+ GetModelMatrix() : glm::mat4&
+ UpdateModelMatrix() : void
Texture
- m_rendererID : unsigned int
+ Bind() : void
+ Unbind() : void
+ Load(string& p_filePath)

Utils

Event (template<typename ... Types>

- m_availableListenerId: uint8_t
- m_callbacks: unordered_map<uint8_t, callback function>
- KeyCode : enum

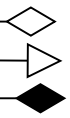
+ Invoke(Types ...p_args) : void
+ AddListener(function p_callback) : uint8_t
+ RemoveListener(uint8_t p_id) : bool
+ RemoveAllListeners() : void

Legend

Aggregation

Heritage

Composition



LowRenderer

Camera

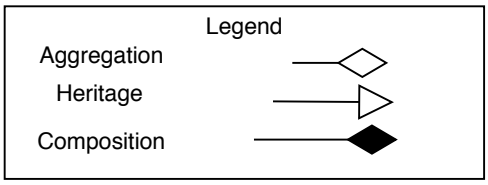
- m_position : glm::vec3
- m_rightAxis: glm::vec3
- m_upAxis: glm::vec3
- m_forward: glm::vec3
- m_target: glm::vec3
- m_back : glm::vec3
- m_movementSpeed : float
- m_yaw : float
- m_pitch : float
- m_firstMouse : bool
- m_lastXPos : double
- m_lastYPos : double
- m_lookSensitivity : float
- m_instance : static unique_ptr<Camera>

- UpdateCameraVector() noexcept : void

+ GetPos() : glm::vec3&
+ GetRight() : glm::vec3&
+ GetUp() : glm::vec3&
+ GetForward() : glm:;vec3&
+ GetTarget(): glm::vec3&
+ GetBack() : glm::vec3&
+ SetPos(glm:;vec3& p_vec) : void

LightSource

+ Position: glm::vec3
+ Ambient: glm::vec3
+ Diffuse: glm::vec3
+ Specular: glm::vec3
+ Direction: glm::vec3



Core Diagram

Core

GameManager

- m_activeScene : int
- m_deltaTime : double
- m_lastFrame : double
- m_device : Rendering::Context::IDevice*
- m_allScenes : vector<Scene>
- m_rendererRef : Rendering::Managers::Renderer*

+ AddGameObjectToScene (const string& p_name, shared_ptr<GameObject> p_gameObject, const int p_scene) : void
+ static AddComponentToGameObject(GameObject& p_object, typename T p_component) : void
+ HandleEvents() : void
+ Update() : void
+ DrawActiveScene(Rendering::Managers::Renderer& p_renderer) : void
+ SetActiveScene(int p_activeScene) : void
+ GetScenes() noexcept : vector<Scene>&
+ GetActiveSceneNumber() const noexcept : int
+ GetGameObjectsInScene(unsigned int p_scene) : unordered_map<string, shared_ptr<Core::GameObject>>&

Scene

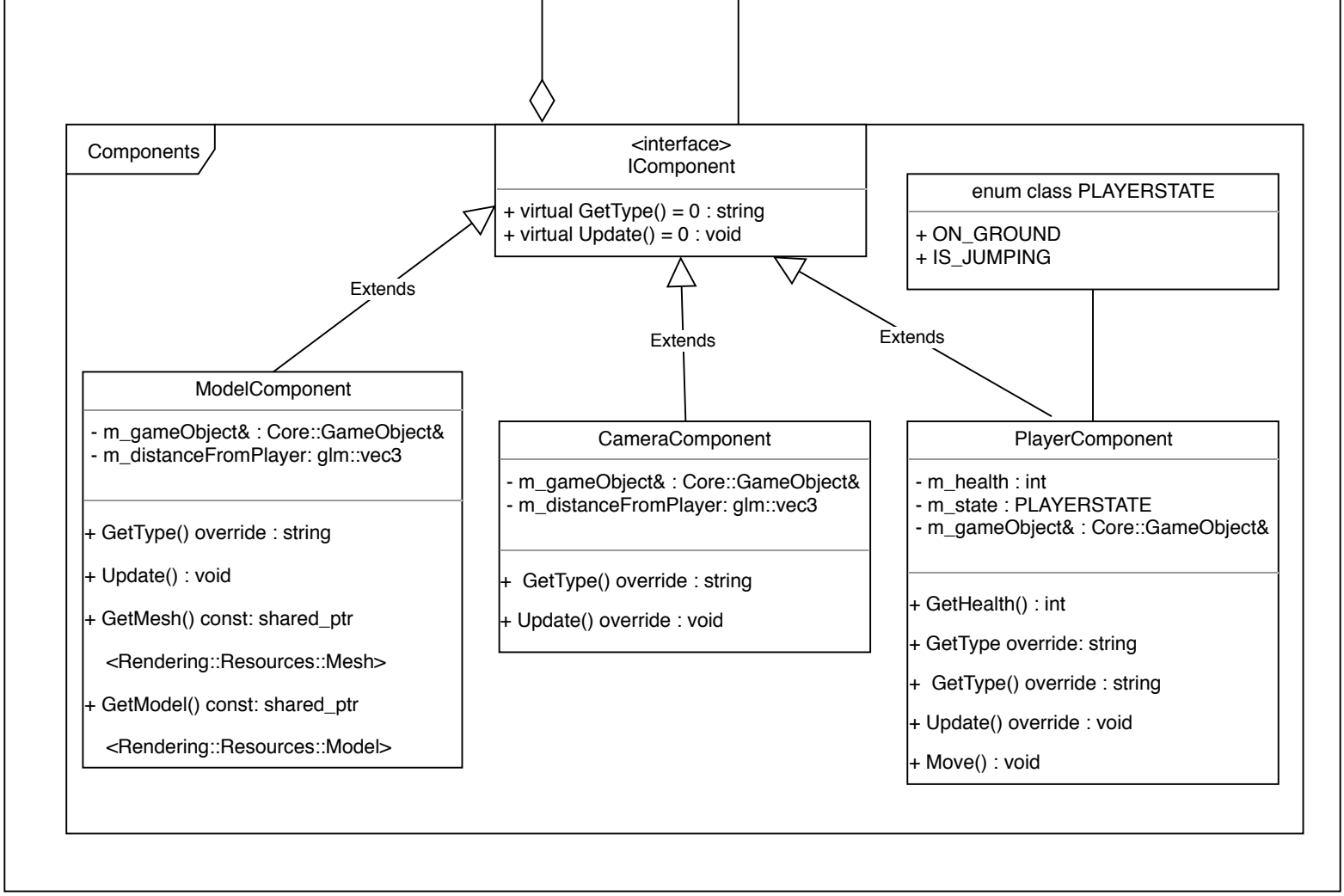
- m_allGameObjects : unordered_map<string, shared_ptr<GameObject>>

+ AddGameObject(shared_ptr<GameObject> p_gameObject, const string p_name) : void
+ FindGameObject(const string& p_name) : shared_ptr<GameObject>
+ DrawAll(Rendering::Managers::Renderer& p_renderer)
+ Update() : void
+ UpdateAllGameObjects() : void
+ UpdateAllComponents() : void
+ UpdateAllShaders() : void
+ GetAllGameObjects(): unordered_map<string, shared_ptr<GameObject>>&

GameObject

+ m_components : vector<shared_ptr<Components::IComponent>> m_components
+ m_transform : Rendering::Resources::Transform

+ MoveTo(glm:: vec3 p_vec) : void
+ TranslateBy(glm::vec3 p_vec): void
+ GetTransform() : Rendering::Resources::Transform&
+ SetTransform(glm::vec3& p_pos, glm::vec3& p_rot, glm::vec3& p_scale) : void
+ SetTransform(Rendering::Resources::Transform& p_transform) : void
+ UpdateAllComponents() : void
+ UpdateShaders() : void
+ AddComponent(typename T p_component) : void
+ AddComponent<typename T>(typename ... args p_args) : void
+ [[nodiscard]] GetComponent<typename T>() const noexcept: shared_ptr<T>
+ GetComponents() : vector<shared_ptr<Components::IComponent>>&


Components
<interface>
IComponent
+ virtual GetType() = 0 : string
+ virtual Update() = 0 : void
enum class PLAYERSTATE
+ ON_GROUND
+ IS_JUMPING
Extends
Extends
Extends
ModelComponent
- m_gameObject& : Core::GameObject&
- m_distanceFromPlayer: glm::vec3
+ GetType() override : string
+ Update() : void
+ GetMesh() const: shared_ptr
<Rendering::Resources::Mesh>
+ GetModel() const: shared_ptr
<Rendering::Resources::Model>
CameraComponent
- m_gameObject& : Core::GameObject&
- m_distanceFromPlayer: glm::vec3
+ GetType() override : string
+ Update() override : void
PlayerComponent
- m_health : int
- m_state : PLAYERSTATE
- m_gameObject& : Core::GameObject&
+ GetHealth() : int
+ GetType override: string
+ GetType() override : string
+ Update() override : void
+ Move() : void