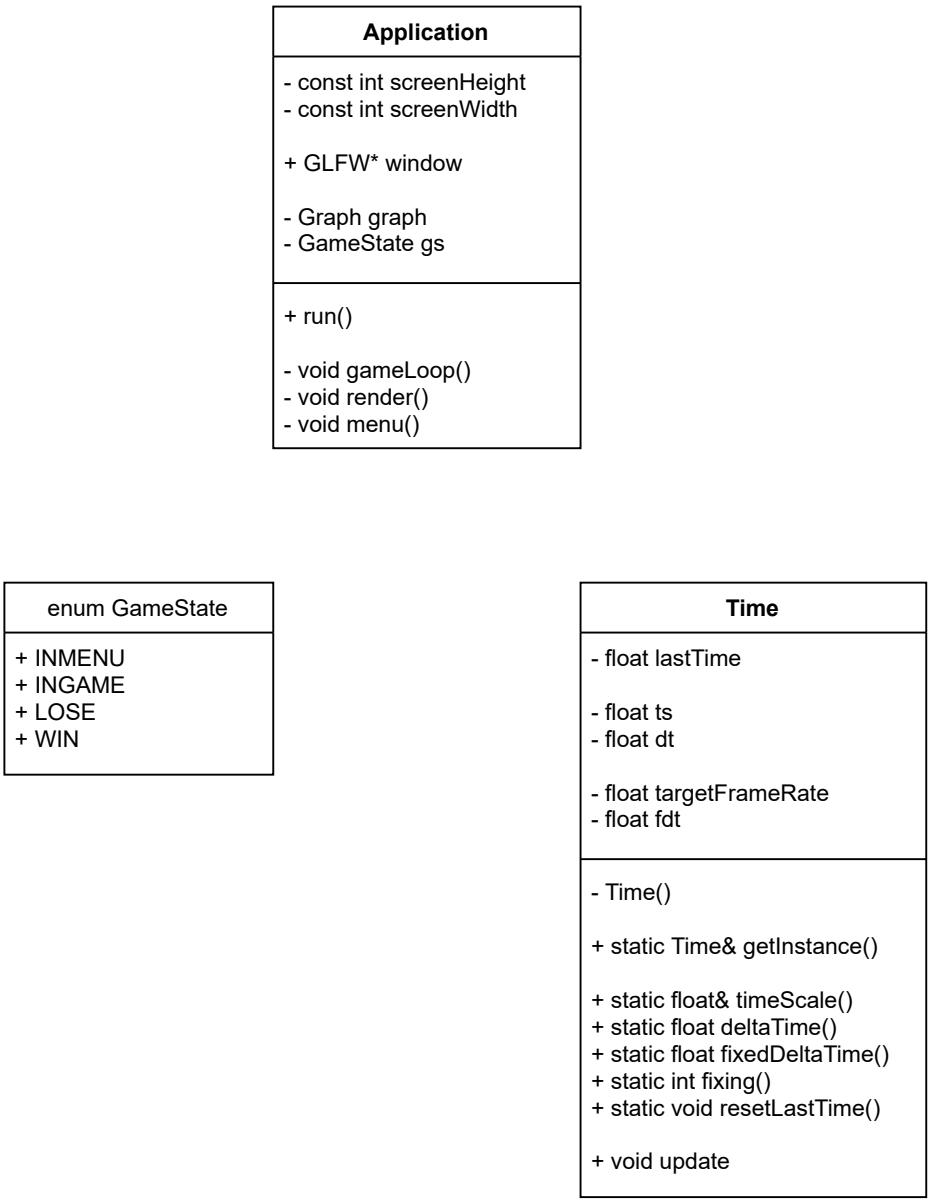
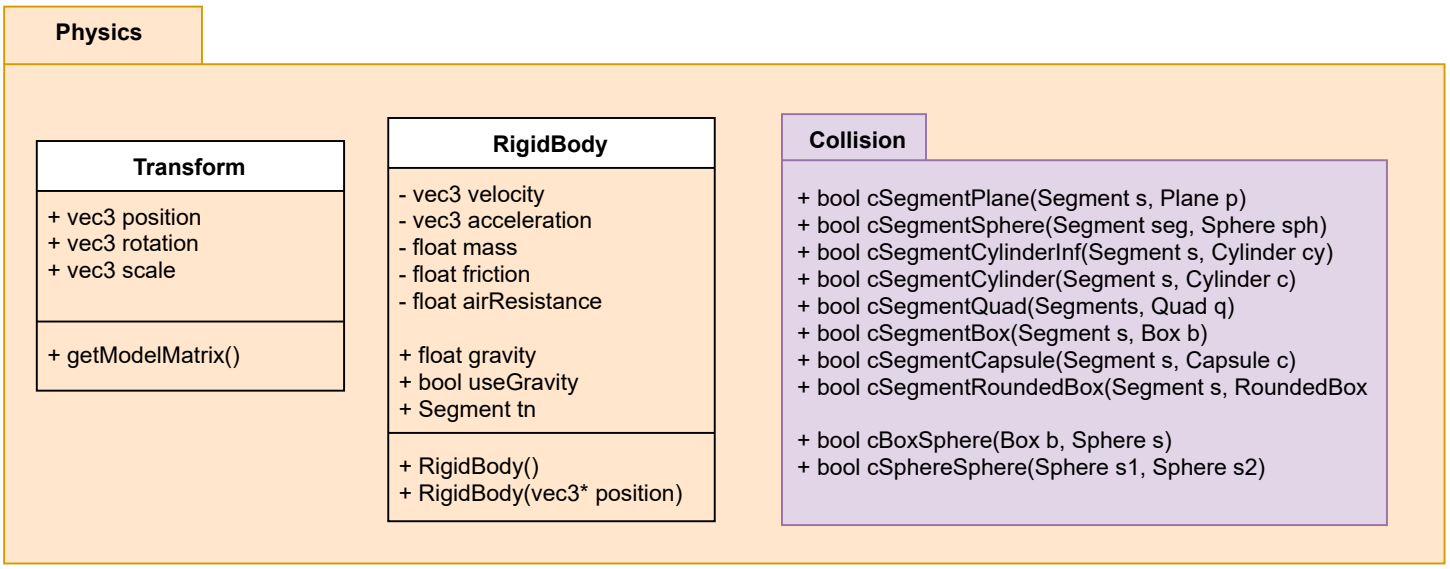
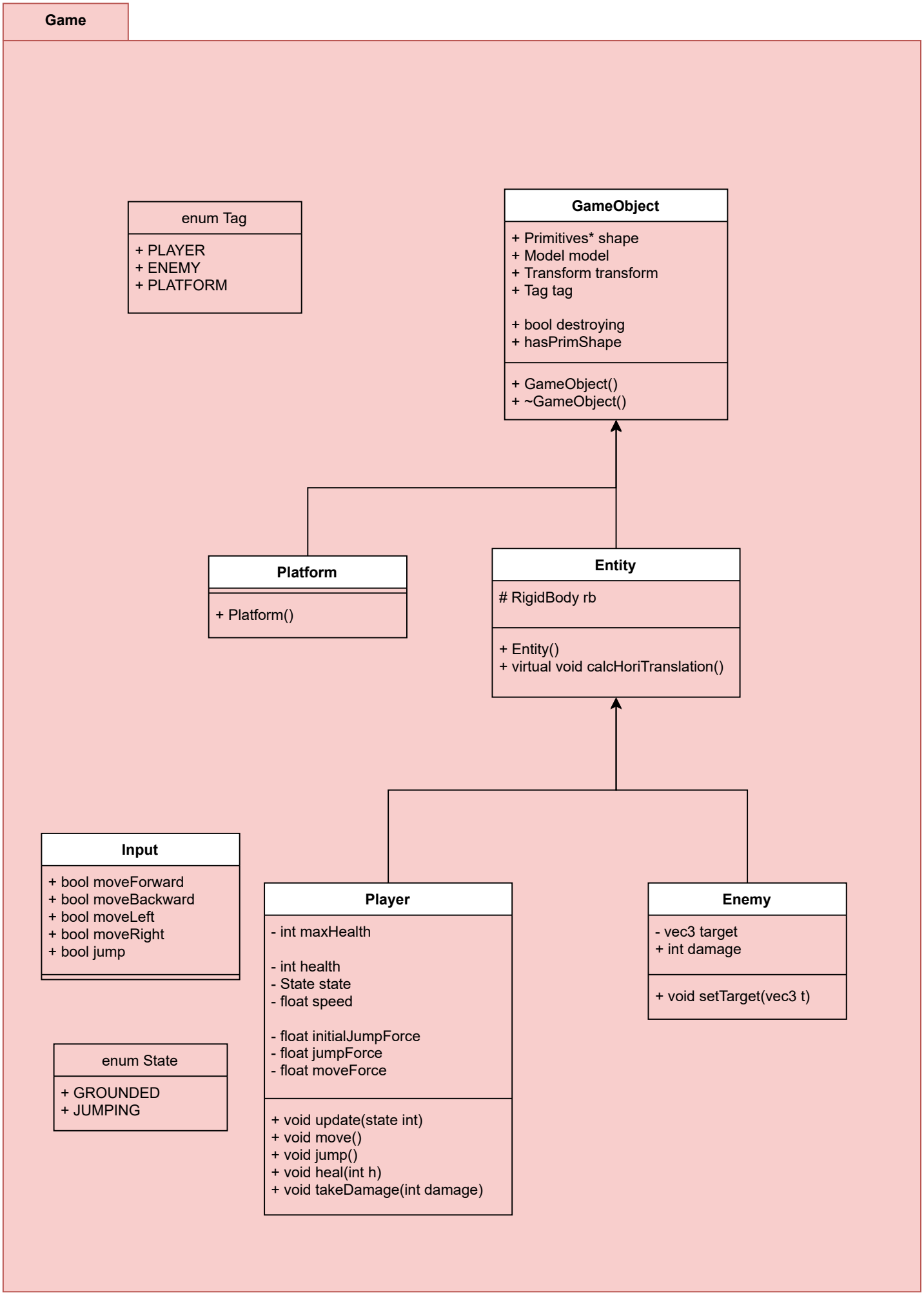
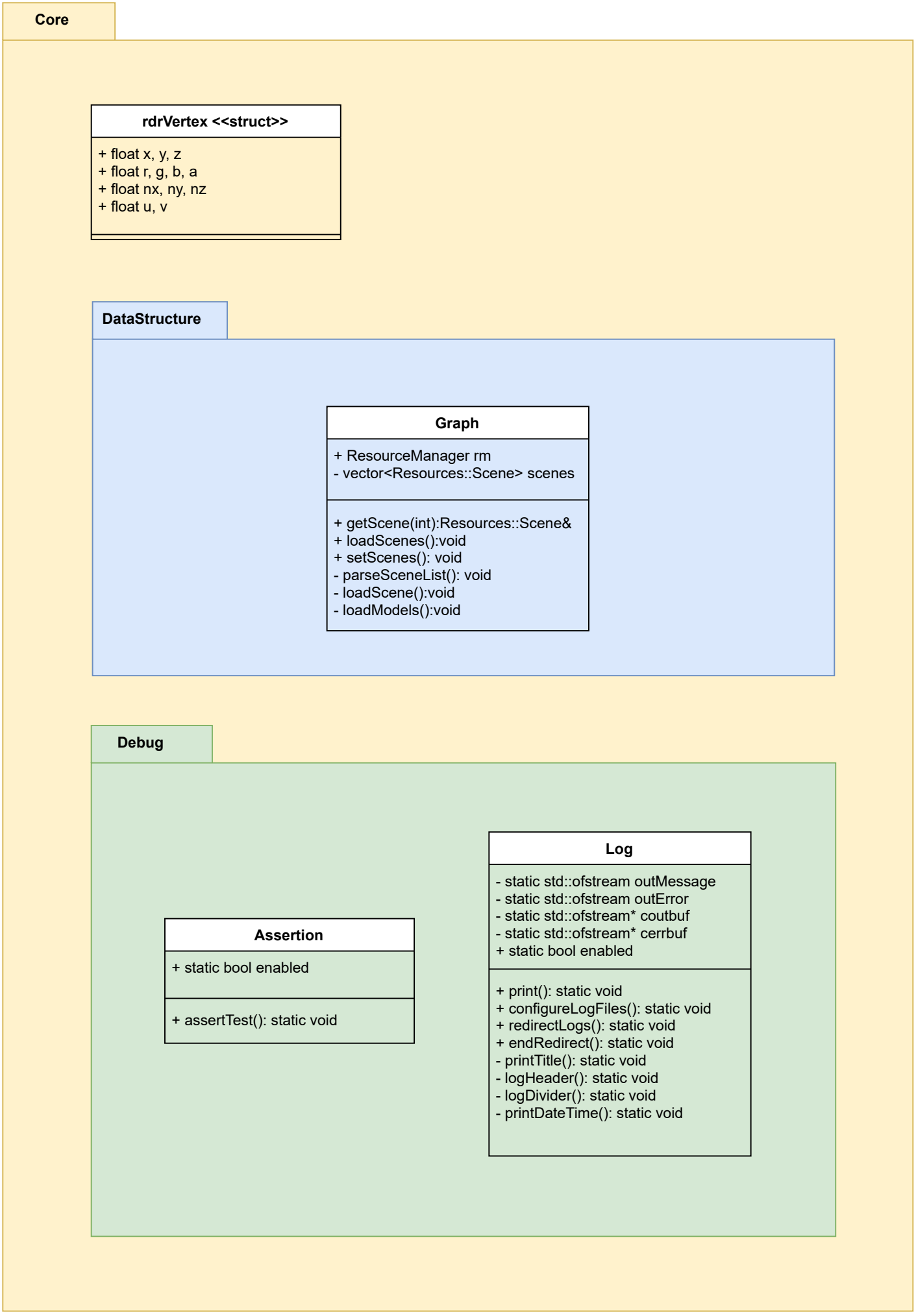
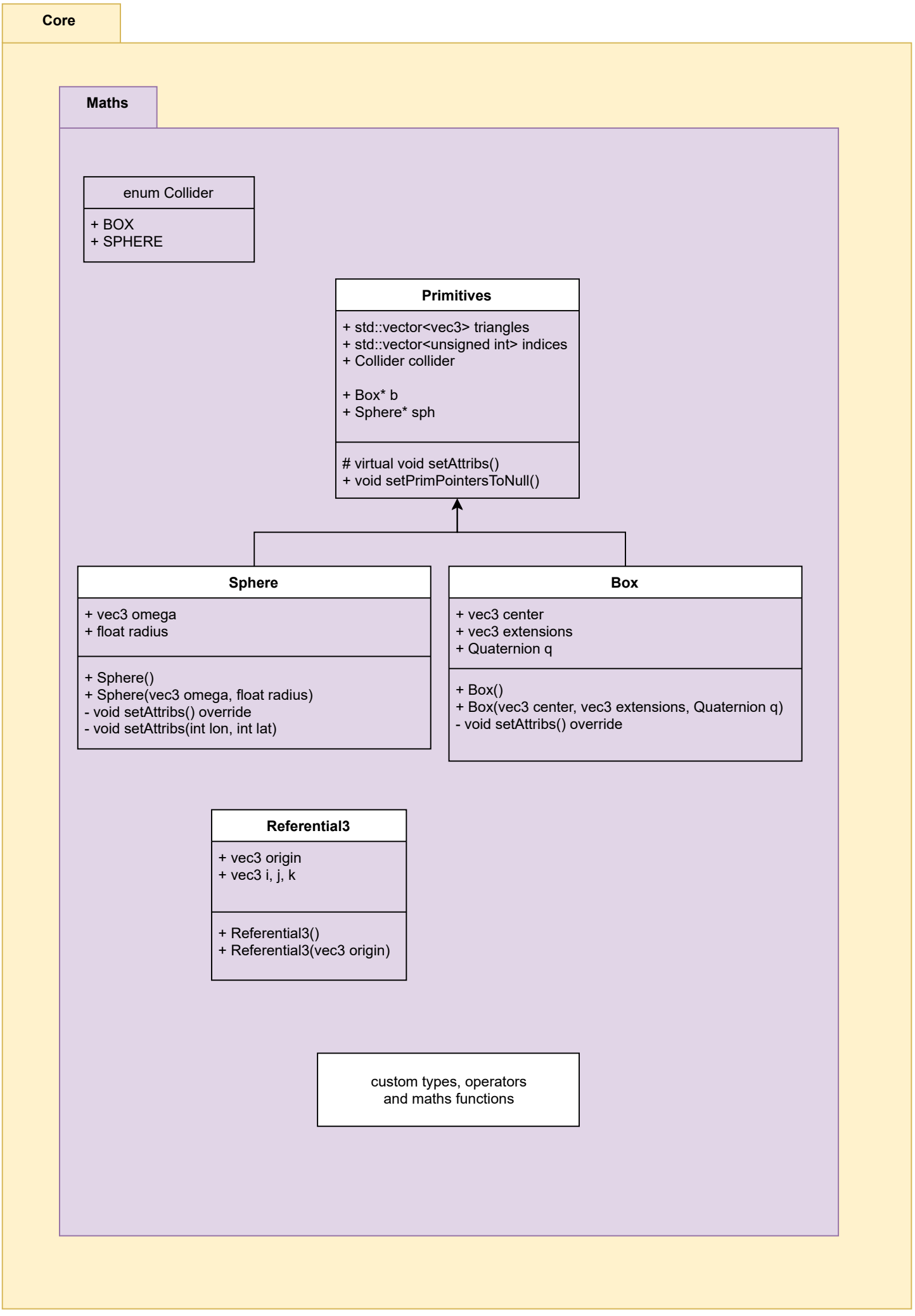
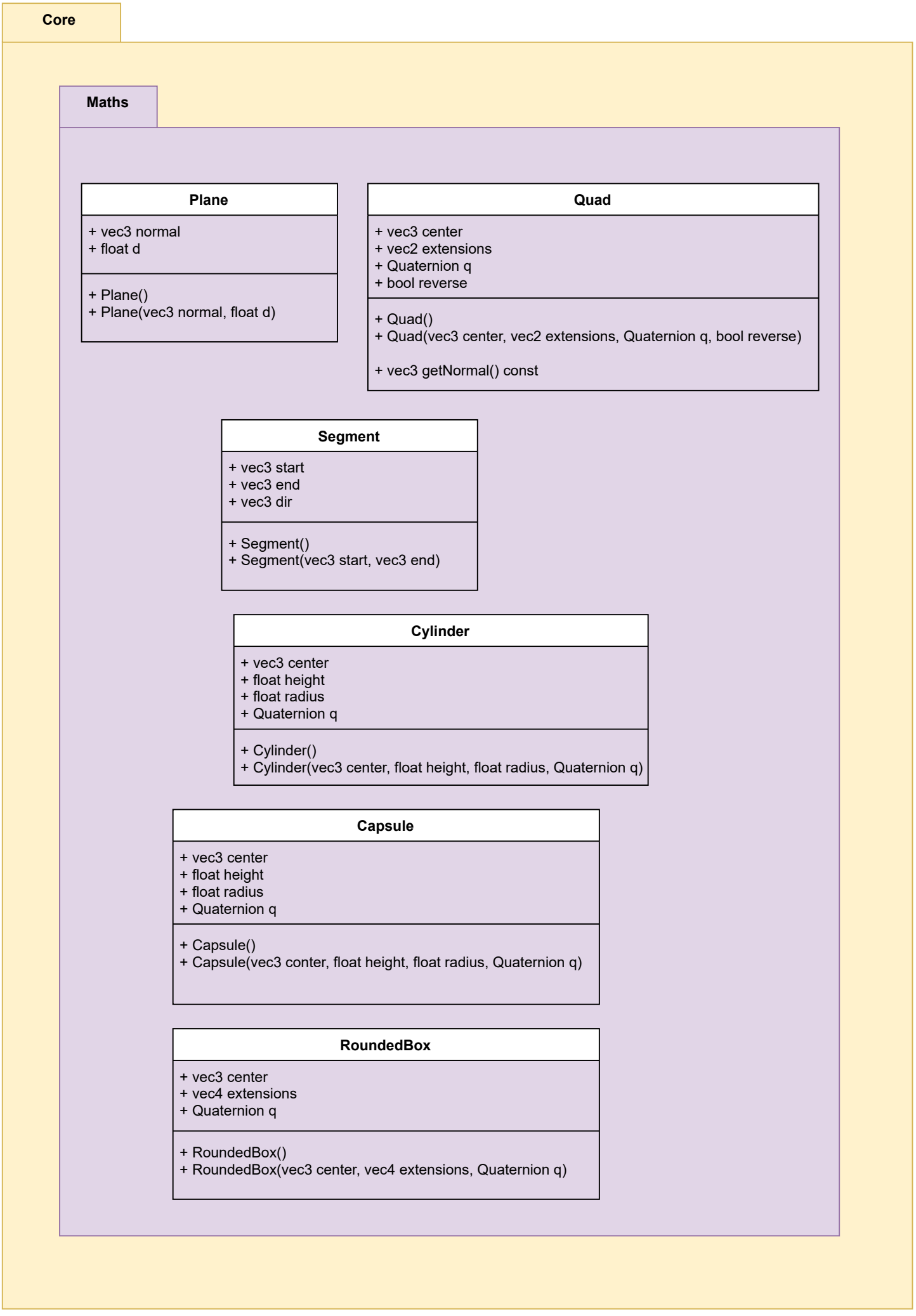

Application
- const int screenHeight
- const int screenWidth
+ GLFW* window
- Graph graph
- GameState gs
+ run()
- void gameLoop()
- void render()
- void menu()
enum GameState
+ INMENU
+ INGAME
+ LOSE
+ WIN
Time
- float lastTime
- float ts
- float dt
- float targetFrameRate
- float fdt
- Time()
+ static Time& getInstance()
+ static float& timeScale()
+ static float deltaTime()
+ static float fixedDeltaTime()
+ static int fixing()
+ static void resetLastTime()
+ void update

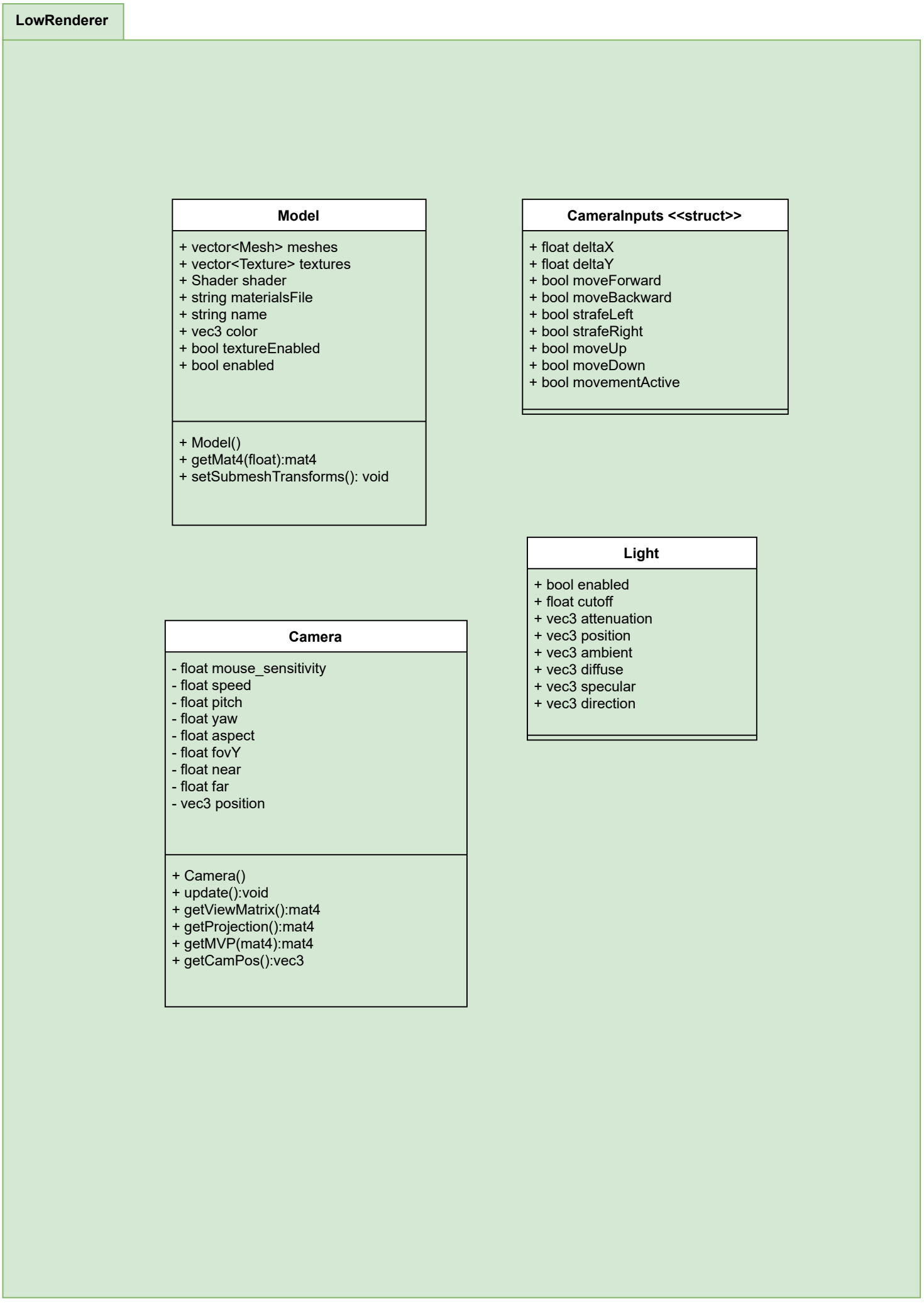

Physics
Transform
+ vec3 position
+ vec3 rotation
+ vec3 scale
+ getModelMatrix()
RigidBody
- vec3 velocity
- vec3 acceleration
- float mass
- float friction
- float airResistance
+ float gravity
+ bool useGravity
+ Segment tn
+ RigidBody()
+ RigidBody(vec3* position)
Collision
+ bool cSegmentPlane(Segment s, Plane p)
+ bool cSegmentSphere(Segment seg, Sphere sph)
+ bool cSegmentCylinderInf(Segment s, Cylinder cy)
+ bool cSegmentCylinder(Segment s, Cylinder c)
+ bool cSegmentQuad(Segments, Quad q)
+ bool cSegmentBox(Segment s, Box b)
+ bool cSegmentCapsule(Segment s, Capsule c)
+ bool cSegmentRoundedBox(Segment s, RoundedBox
+ bool cBoxSphere(Box b, Sphere s)
+ bool cSphereSphere(Sphere s1, Sphere s2)

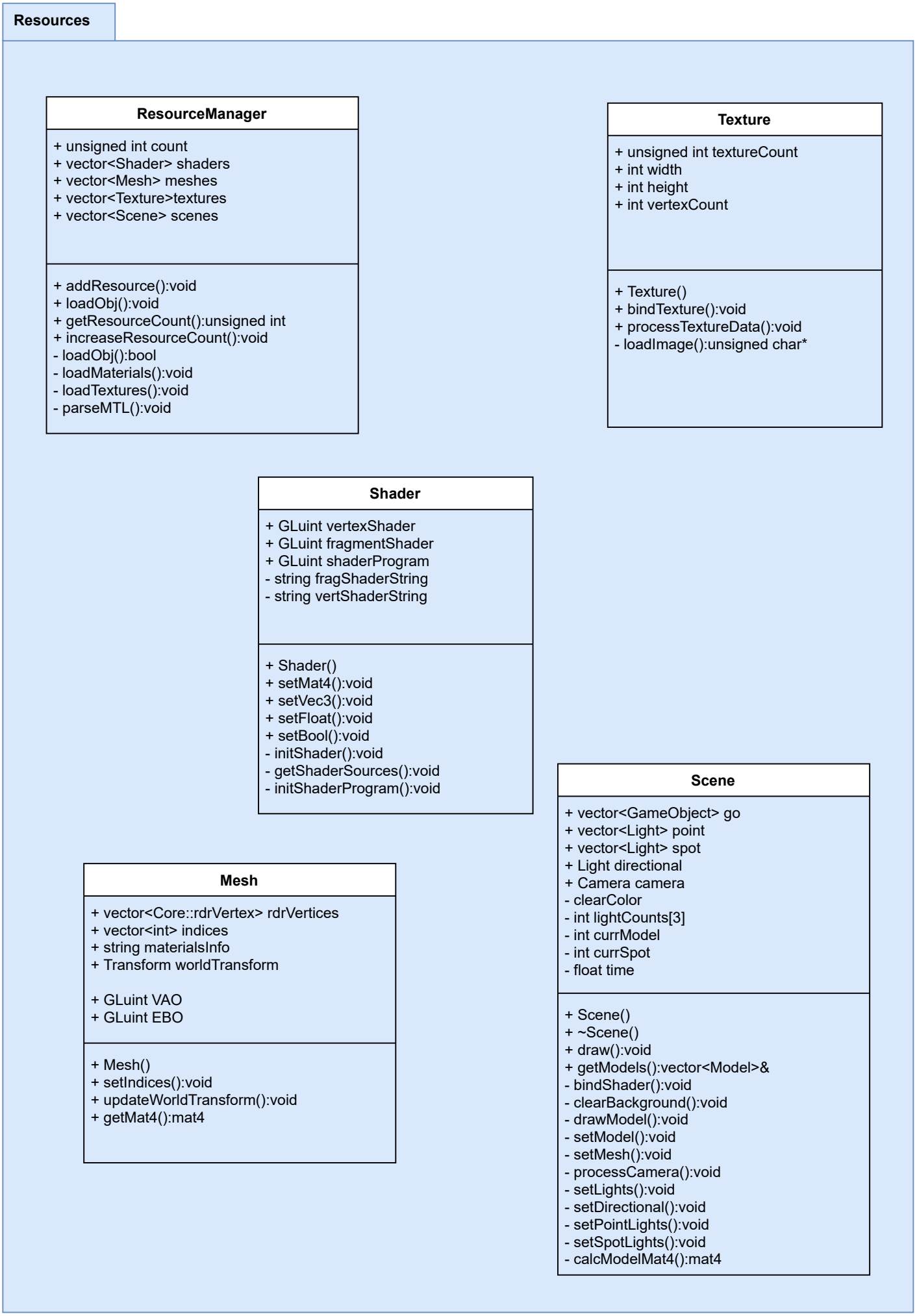

Game
enum Tag
+ PLAYER
+ ENEMY
+ PLATFORM
GameObject
+ Primitives* shape
+ Model model
+ Transform transform
+ Tag tag
+ bool destroying
+ hasPrimShape
+ GameObject()
+ ~GameObject()
Platform
+ Platform()
Entity
RigidBody rb
+ Entity()
+ virtual void calcHoriTranslation()
Input
+ bool moveForward
+ bool moveBackward
+ bool moveLeft
+ bool moveRight
+ bool jump
Player
- int maxHealth
- int health
- State state
- float speed
- float initialJumpForce
- float jumpForce
- float moveForce
+ void update(state int)
+ void move()
+ void jump()
+ void heal(int h)
+ void takeDamage(int damage)
Enemy
- vec3 target
+ int damage
+ void setTarget(vec3 t)
enum State
+ GROUNDED
+ JUMPING


Core
rdrVertex <<struct>>
+ float x, y, z
+ float r, g, b, a
+ float nx, ny, nz
+ float u, v
DataStructure
Graph
+ ResourceManager rm
- vector<Resources::Scene> scenes
+ getScene(int):Resources::Scene&
+ loadScenes():void
+ setScenes(): void
- parseSceneList(): void
- loadScene():void
- loadModels():void
Debug
Assertion
+ static bool enabled
+ assertTest(): static void
Log
- static std::ofstream outMessage
- static std::ofstream outError
- static std::ofstream* coutbuf
- static std::ofstream* cerrbuf
+ static bool enabled
+ print(): static void
+ configureLogFiles(): static void
+ redirectLogs(): static void
+ endRedirect(): static void
- printTitle(): static void
- logHeader(): static void
- logDivider(): static void
- printDateTime(): static void


Core
Maths
enum Collider
+ BOX
+ SPHERE
Primitives
+ std::vector<vec3> triangles
+ std::vector<unsigned int> indices
+ Collider collider
+ Box* b
+ Sphere* sph
# virtual void setAttribs()
+ void setPrimPointersToNull()
Sphere
+ vec3 omega
+ float radius
+ Sphere()
+ Sphere(vec3 omega, float radius)
- void setAttribs() override
- void setAttribs(int lon, int lat)
Box
+ vec3 center
+ vec3 extensions
+ Quaternion q
+ Box()
+ Box(vec3 center, vec3 extensions, Quaternion q)
- void setAttribs() override
Referential3
+ vec3 origin
+ vec3 i, j, k
+ Referential3()
+ Referential3(vec3 origin)
custom types, operators
and maths functions


Core
Maths
Plane
+ vec3 normal
+ float d
+ Plane()
+ Plane(vec3 normal, float d)
Quad
+ vec3 center
+ vec2 extensions
+ Quaternion q
+ bool reverse
+ Quad()
+ Quad(vec3 center, vec2 extensions, Quaternion q, bool reverse)
+ vec3 getNormal() const
Segment
+ vec3 start
+ vec3 end
+ vec3 dir
+ Segment()
+ Segment(vec3 start, vec3 end)
Cylinder
+ vec3 center
+ float height
+ float radius
+ Quaternion q
+ Cylinder()
+ Cylinder(vec3 center, float height, float radius, Quaternion q)
Capsule
+ vec3 center
+ float height
+ float radius
+ Quaternion q
+ Capsule()
+ Capsule(vec3 conter, float height, float radius, Quaternion q)
RoundedBox
+ vec3 center
+ vec4 extensions
+ Quaternion q
+ RoundedBox()
+ RoundedBox(vec3 center, vec4 extensions, Quaternion q)


LowRenderer
Model
+ vector<Mesh> meshes
+ vector<Texture> textures
+ Shader shader
+ string materialsFile
+ string name
+ vec3 color
+ bool textureEnabled
+ bool enabled
+ Model()
+ getMat4(float):mat4
+ setSubmeshTransforms(): void
CameraInputs <<struct>>
+ float deltaX
+ float deltaY
+ bool moveForward
+ bool moveBackward
+ bool strafeLeft
+ bool strafeRight
+ bool moveUp
+ bool moveDown
+ bool movementActive
Light
+ bool enabled
+ float cutoff
+ vec3 attenuation
+ vec3 position
+ vec3 ambient
+ vec3 diffuse
+ vec3 specular
+ vec3 direction
Camera
- float mouse_sensitivity
- float speed
- float pitch
- float yaw
- float aspect
- float fovY
- float near
- float far
- vec3 position
+ Camera()
+ update():void
+ getViewMatrix():mat4
+ getProjection():mat4
+ getMVP(mat4):mat4
+ getCamPos():vec3


Resources
ResourceManager
+ unsigned int count
+ vector<Shader> shaders
+ vector<Mesh> meshes
+ vector<Texture>textures
+ vector<Scene> scenes
+ addResource():void
+ loadObj():void
+ getResourceCount():unsigned int
+ increaseResourceCount():void
- loadObj():bool
- loadMaterials():void
- loadTextures():void
- parseMTL():void
Texture
+ unsigned int textureCount
+ int width
+ int height
+ int vertexCount
+ Texture()
+ bindTexture():void
+ processTextureData():void
- loadImage():unsigned char*
Shader
+ GLuint vertexShader
+ GLuint fragmentShader
+ GLuint shaderProgram
- string fragShaderString
- string vertShaderString
+ Shader()
+ setMat4():void
+ setVec3():void
+ setFloat():void
+ setBool():void
- initShader():void
- getShaderSources():void
- initShaderProgram():void
Mesh
+ vector<Core::rdrVertex> rdrVertices
+ vector<int> indices
+ string materialsInfo
+ Transform worldTransform
+ GLuint VAO
+ GLuint EBO
+ Mesh()
+ setIndices():void
+ updateWorldTransform():void
+ getMat4():mat4
Scene
+ vector<GameObject> go
+ vector<Light> point
+ vector<Light> spot
+ Light directional
+ Camera camera
- clearColor
- int lightCounts[3]
- int currModel
- int currSpot
- float time
+ Scene()
+ ~Scene()
+ draw():void
+ getModels():vector<Model>&
- bindShader():void
- clearBackground():void
- drawModel():void
- setModel():void
- setMesh():void
- processCamera():void
- setLights():void
- setDirectional():void
- setPointLights():void
- setSpotLights():void
- calcModelMat4():mat4