



ugr

Universidad
de Granada



Trabajo de fin de Grado
Grado en ingeniería informática(2016-2017)

Anexo 1: Diagrama de clases

Alumno

Antonio David López Machado

Tutor

Carlos Ureña Almagro

Índice

1. Partida	2
2. Objetos 3D	2
3. Avatar	2
4. Enemigo	3
5. Npc	3
6. Menú	3
7. Matriz	4
8. Animación	4
9. Contexto	4
10. Estado de juego y camara	5
11. Objetos en la escena	5
12. Escena	5
13. Sistema de puertas	6
14. Sistema de partículas/proyectiles y objetos	6
15. Malla	7
16. Material	7
17. Iluminación	8
18. Sombras	8
19. Región	8
20. Sonido	9
21. Colección	9
22. Gestor de ficheros	9
23. Controlador	10

1. Partida

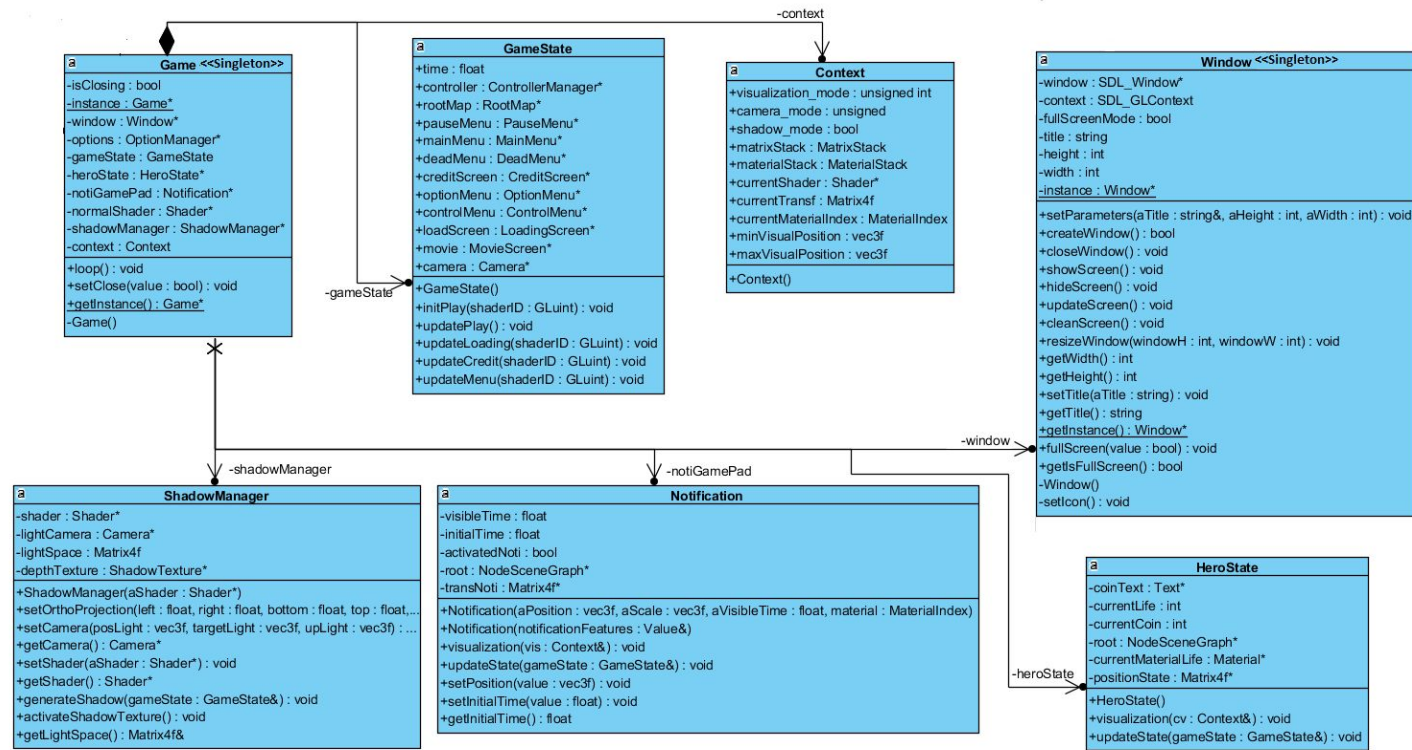


Figura 01 Diagrama de clases de partida

2. Objetos 3D

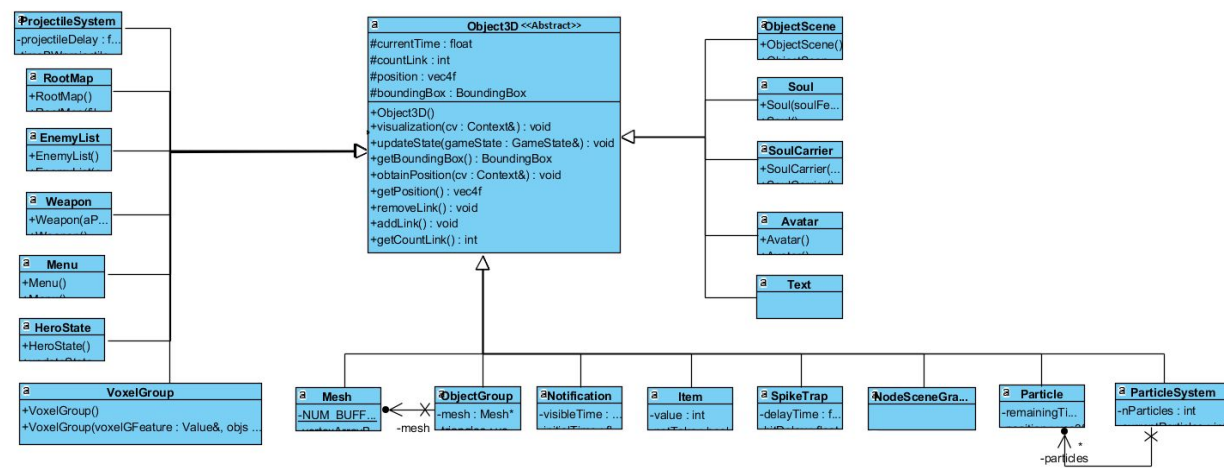


Figura 02 Diagrama de clases de objetos 3D

3. Avatar

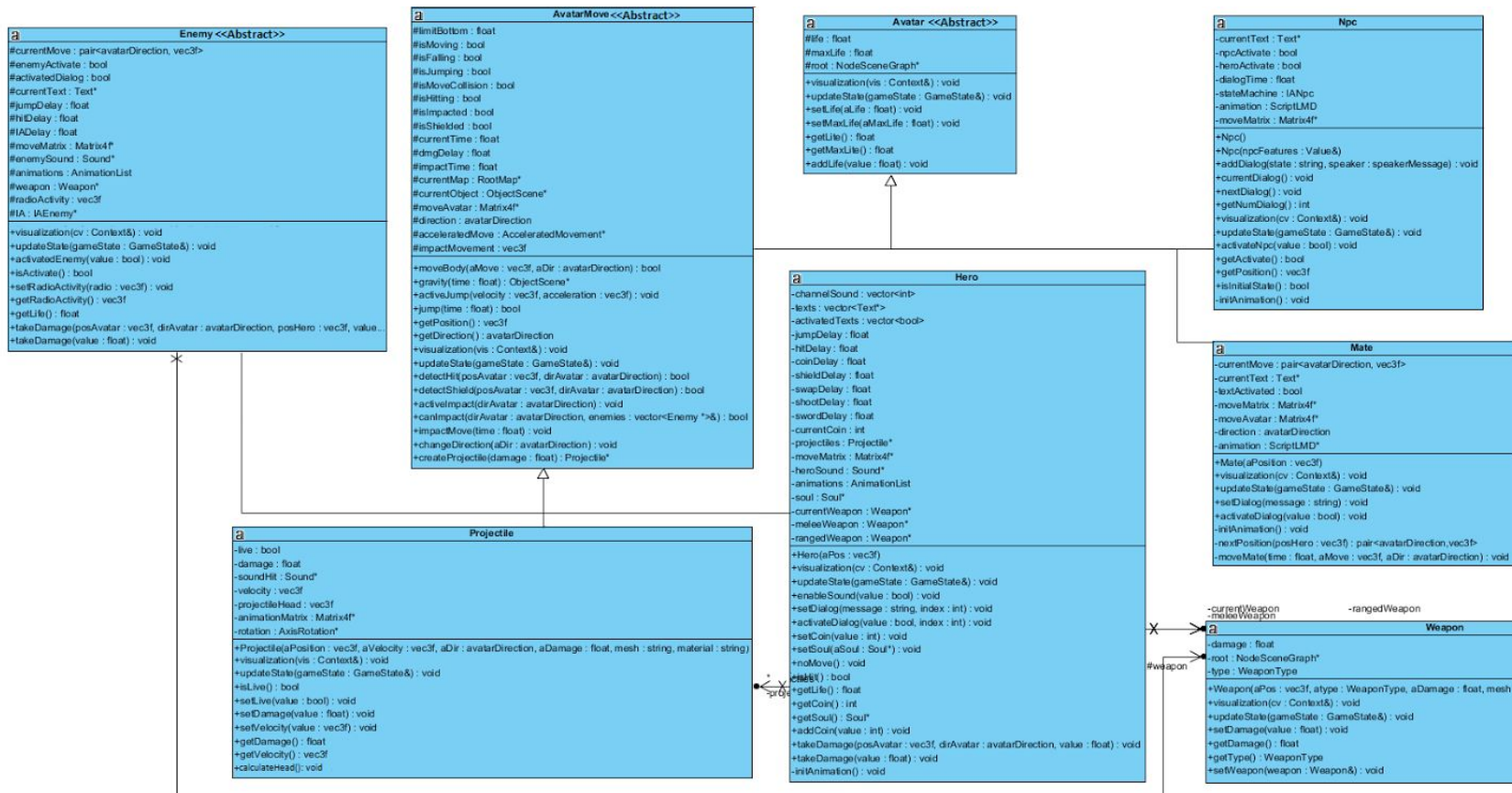


Figura 03 Diagrama de clases de avatar

4. Enemigo

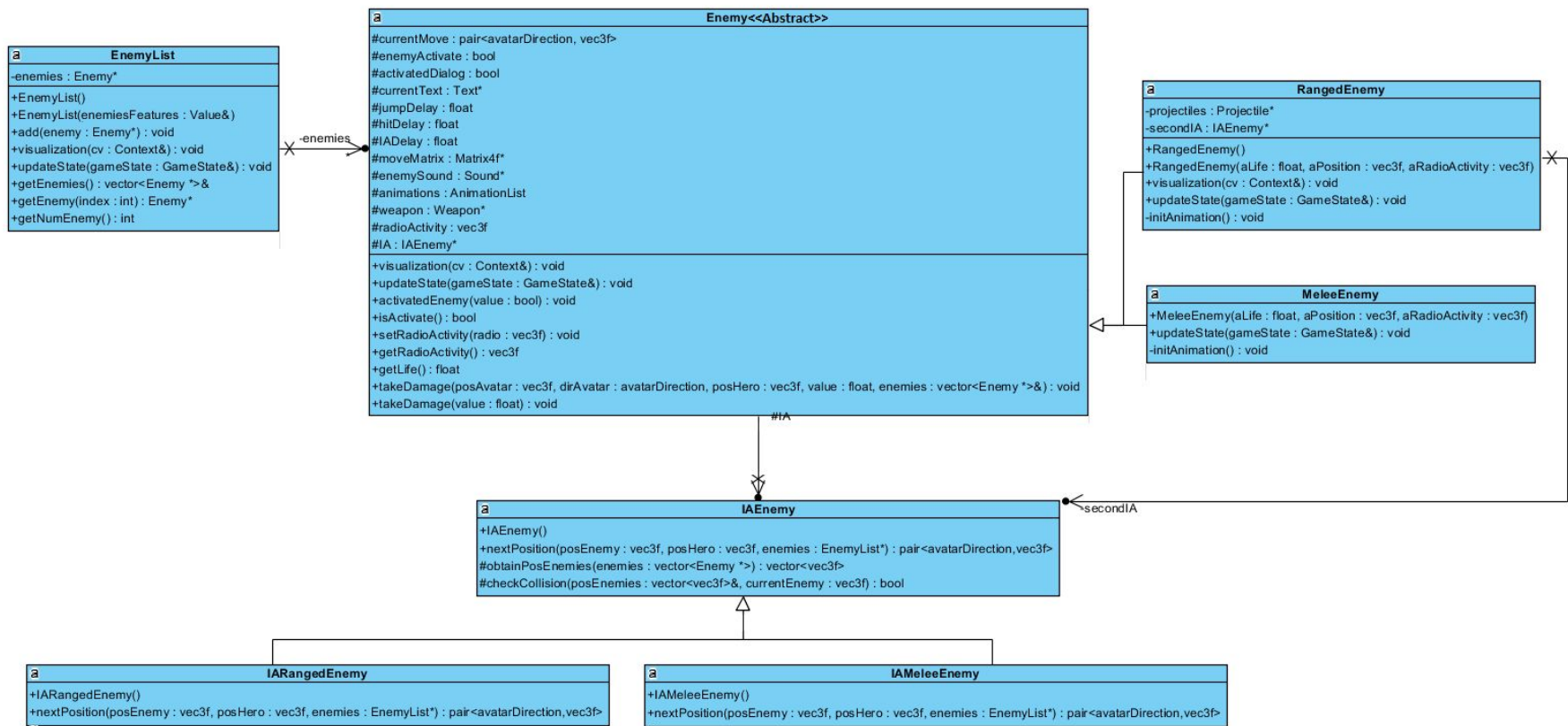


Figura 04 Diagrama de clases de enemigo

5. Npc

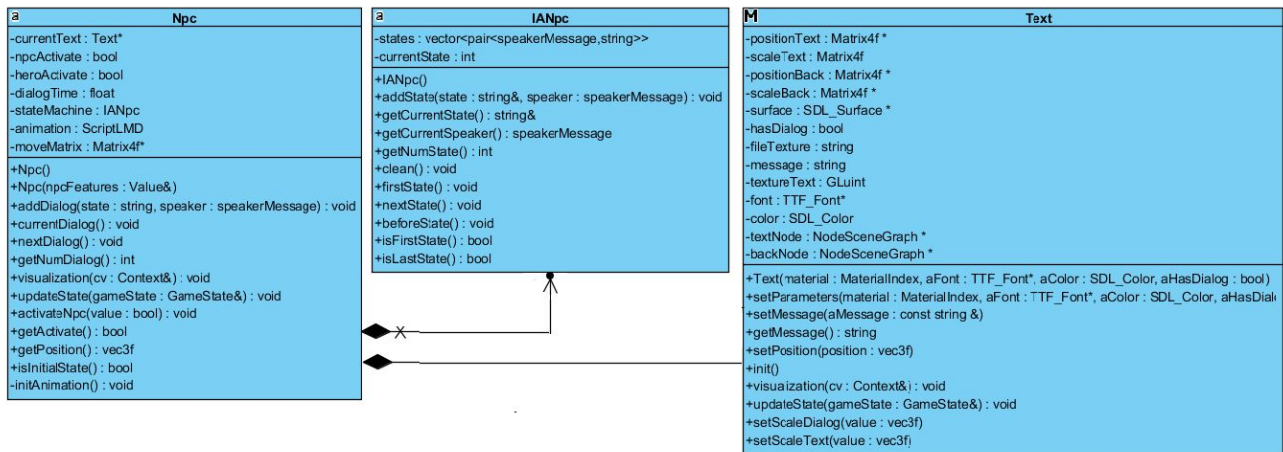


Figura 05 Diagrama de clases de npc

6. Menú

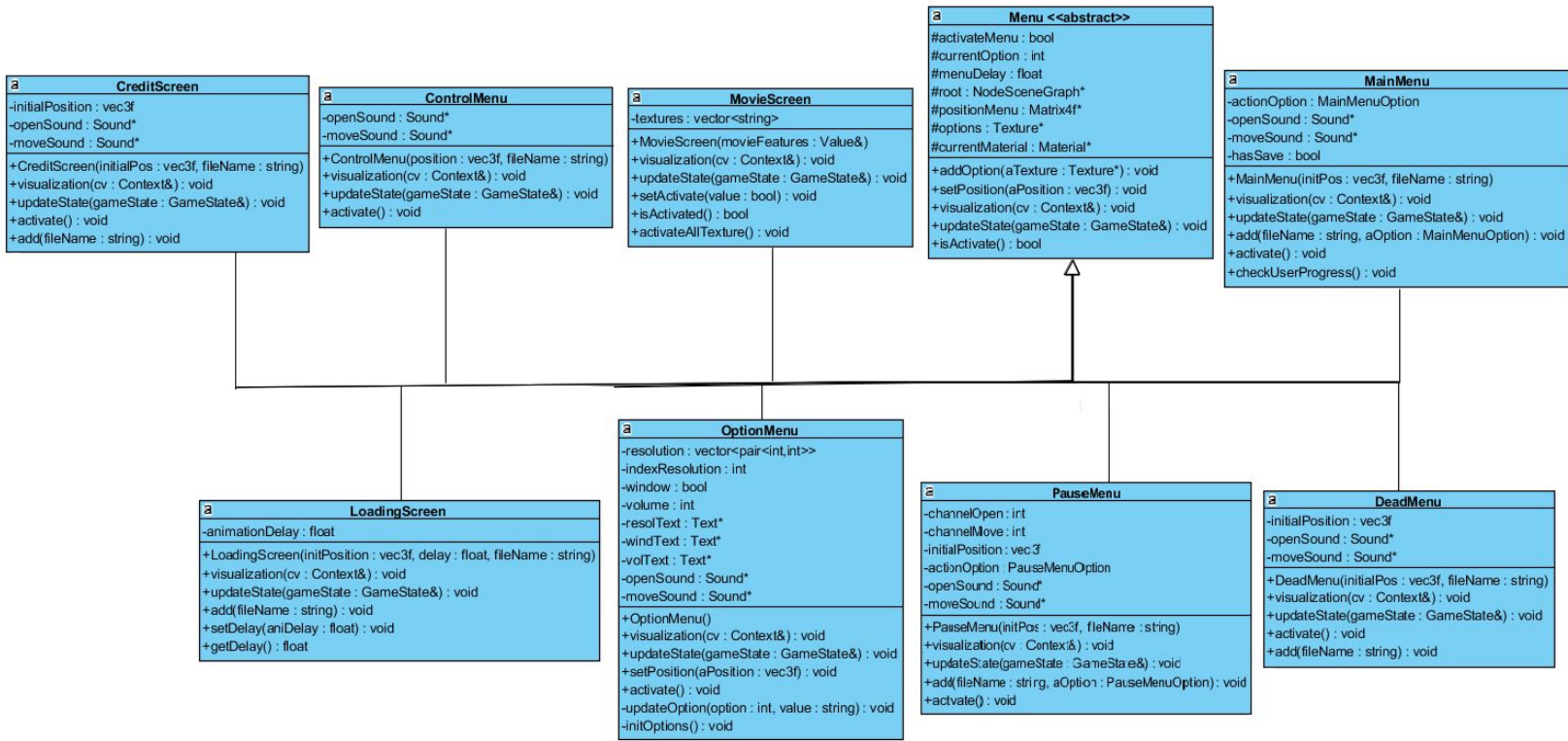


Figura 06 Diagrama de clases Menú

```

classDiagram
    class Matrix4f {
        -matrix: GLfloat*
        +Matrix4f()
        +Matrix4f(aMatrix: Matrix4f&)
        +translation(x: float, y: float, z: float): void
        +scale(x: float, y: float, z: float): void
        +rotation(grade: float, x: float, y: float, z: float): void
        +identity(): void
        +product(aMatrix: GLfloat*): void
        +product(aVector: vec4f): vec4f
        +setMatrix(aMatrix: GLfloat*): void
        +getMatrix(): GLfloat*
        +getMatrix(c): GLfloat*
    }
    class Matrix4fDynamic {
        <<Abstract>>
        #currentMatrix: Matrix4f
        +updateState(time: float): Matrix4f&
        +resetState(): void
        +getMatrix(): Matrix4f&
    }
    class LinearMovement {
        -currentTime: float
        -velocity: vec3f
        +LinearMovement()
        +LinearMovement(x: float, y: float, z: float)
        +LinearMovement(aVelocity: vec3f&)
        +setParameters(x: float, y: float, z: float): void
        +setParameters(aVelocity: vec3f&): void
        +updateState(time: float): Matrix4f&
        +resetState(): void
    }
    class MatrixStatic {
        +MatrixStatic()
        +MatrixStatic(matrix: Matrix4f&)
        +updateState(time: float): Matrix4f&
        +resetState(): void
    }
    class AxisRotation {
        -angularVelocity: float
        -currentGrade: float
        -axis: vec3f
        +AxisRotation()
        +AxisRotation(anAVelocity: float, x: float, y: float, z: float)
        +setParameters(anAVelocity: float, x: float, y: float, z: float): void
        +updateState(time: float): Matrix4f&
        +resetState(): void
        +getAngularVelocity(): float
        +getAxis(): vec3f
        +getCurrentGrade(): float
    }
    class MatrixScript {
        -currentTime: float
        -currentMove: int
        -script: vector<pair<float, Matrix4fDynamic*>>
        +MatrixScript()
        +add(time: float, matrix: Matrix4fDynamic*<<Abstract>>)
        +getNumElement(): int
        +updateState(time: float): Matrix4f&
        +resetState(): void
        +getMatrix(): Matrix4f&
        +getState(): int
    }
    class OscillateRotation {
        -maxGrade: float
        -minGrade: float
        -initialGrade: float
        -currentGrade: float
        -angularVelocity: float
        -currentTime: float
        -increment: bool
        -loop: int
        -currentLoop: int
        -direction: vec3f
        +OscillateRotation()
        +OscillateRotation(incree: bool, maxG: float, minG: float, initG: float, velocity: float, dir: vec3f, aLoop: int)
        +setParameters(incree: bool, maxG: float, minG: float, initG: float, velocity: float, dir: vec3f, aLoop: int): void
        +updateState(time: float): Matrix4f&
        +resetState(): void
        +getMaxGrade(): float
        +getMinGrade(): float
        +getInitialGrade(): float
        +getCurrentGrade(): float
        +getAngularVelocity(): float
        +getIncrement(): bool
        +getLoop(): int
    }
    class AcceleratedMovement {
        -currentTime: float
        -velocity: vec3f
        -acceleration: vec3f
        +AcceleratedMovement()
        +AcceleratedMovement(xVelo: float, yVelo: float, zVelo: float, xAccel: float, yAccel: float, zAccel: float)
        +AcceleratedMovement(aVelocity: vec3f&, aAcceleration: vec3f&)
        +setParameters(xVelo: float, yVelo: float, zVelo: float, xAccel: float, yAccel: float, zAccel: float): void
        +setParameters(aVelocity: vec3f&, aAcceleration: vec3f&): void
        +updateState(time: float): Matrix4f&
        +resetState(): void
        +getVelocity(): vec3f
        +getAcceleration(): vec3f
    }
    Matrix4fDynamic <|-- Matrix4f
    Matrix4fDynamic <|-- LinearMovement
    Matrix4fDynamic <|-- MatrixStatic
    Matrix4fDynamic <|-- AxisRotation
    Matrix4fDynamic <|-- MatrixScript
    Matrix4fDynamic <|-- OscillateRotation
    Matrix4fDynamic <|-- AcceleratedMovement
    Matrix4fDynamic --> Matrix4f : #currentMatrix
    
```

Figura 07 Diagrama de clases del sistema de matrices

```
classDiagram
    class AnimationList {
        -currentAnimation : int
        -animations : ScriptLMD*
        +AnimationList()
        +add(animation : ScriptLMD*) : void
        +activate(index : int) : void
        +update(time : float) : void
        +resetAnimation(index : int) : void
        +getState() : int
        +getAnimation() : ScriptLMD*
        +getNumAnimation() : int
    }
    class ScriptLMD {
        -currentTime : float
        -script : MatrixScript*
        +ScriptLMD()
        +add(aMatrix : MatrixScript*) : void
        +updateState(time : float) : void
        +readMatrix(index : int) : Matrix4f
        +getScriptState(index : int) : int
        +resetState() : void
        +getNumState() : int
    }
    class MatrixScript {
        -currentTime : float
        -currentMove : int
        -script : vector<pair<float, Matrix4fDynamic*>>
        +MatrixScript()
        +add(time : float, matrix : Matrix4fDynamic*) : void
        +getNumElement() : int
        +updateState(time : float) : Matrix4f
        +resetState() : void
        +getMatrix() : Matrix4f
        +getState() : int
    }
    class ScriptAnimation {
        <<Abstract>>
        -initialTime : float
        +updateState(time : float) : void
        +readMatrix(index : int) : Matrix4f
    }
    class Matrix4fDynamic {
        <<Abstract>>
        #currentMatrix : Matrix4f
        +updateState(time : float) : Matrix4f
        +resetState() : void
        +getMatrix() : Matrix4f
    }
    AnimationList "1" -- "*" ScriptLMD : -animations
    ScriptLMD "1" -- "1" MatrixScript : -script
    ScriptAnimation <|-- ScriptLMD
    Matrix4fDynamic <|-- MatrixScript
```

Figura 08 Diagrama de clases del sistema de animación

```

classDiagram
    class Context {
        +visualization_mode : unsigned int
        +camera_mode : unsigned
        +shadow_mode : bool
        +matrixStack : MatrixStack
        +materialStack : MaterialStack
        +currentShader : Shader*
        +currentTransf : Matrix4f
        +currentMaterialIndex : MaterialIndex
        +minVisualPosition : vec3f
        +maxVisualPosition : vec3f
        +Context()
    }
    class Shader {
        -vertexfile : string
        -fragmentfile : string
        -programID : GLuint
        -fragmentID : GLuint
        -vertexID : GLuint
        +Shader()
        +Shader(aVertexfile : string&, aFragmentfile : string&)
        +setFiles(aVertexfile : string&, aFragmentfile : string&) : void
        +compileFragmentShaders(aFragmentfile : string& = "") : bool
        +compileVertexShaders(aVertexfile : string& = "") : bool
        +linkShaders() : bool
        +createProgram(aVertexfile : string& = "", aFragmentfile : string& = "") : bool
        +getProgram() : GLuint
        -LoadFileTxt(file : string&) : string
    }
    class MatrixStack {
        -currentMatrix : Matrix4f
        -mainStack : Matrix4f
        +MatrixStack()
        +push() : void
        +pop(cont : int) : void
        +activate(shaderID : GLuint) : void
        +assignIdentity() : void
        +cMatrix(mat : Matrix4f&) : void
        +cTraslation(dx : float, dy : float, dz : float) : void
        +cScale(sx : float, sy : float, sz : float) : void
        +cRotation(ang_gra : float, ex : float, ey : float, ez : float) : void
        +getMatrix() : Matrix4f&
        +clean() : void
    }
    class MaterialStack {
        -mainStack : Material*
        +MaterialStack()
        +push(aMaterial : Material*) : void
        +pop(cont : int) : void
        +getMaterial() : Material*
    }
    Context "1" -- "*" MatrixStack : +matrixStack
    Context "1" -- "*" MaterialStack : +materialStack
    Context "1" -- "*" Shader : +currentShader
    
```

Figura 09 Diagrama de clases del contexto

10. Estado de juego y camara

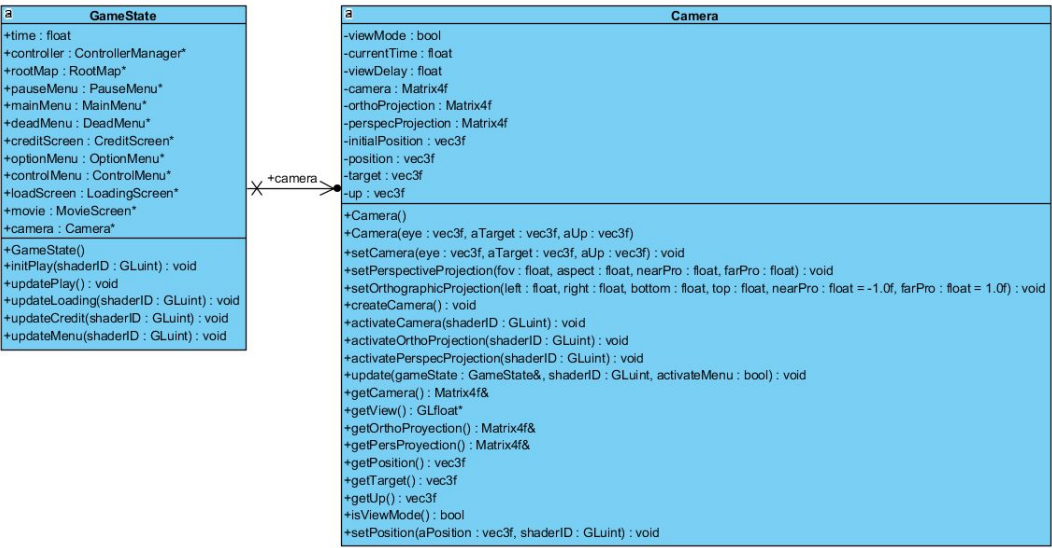


Figura 10 Diagrama de clases de estado de juego y camara.

11. Objetos en la escena

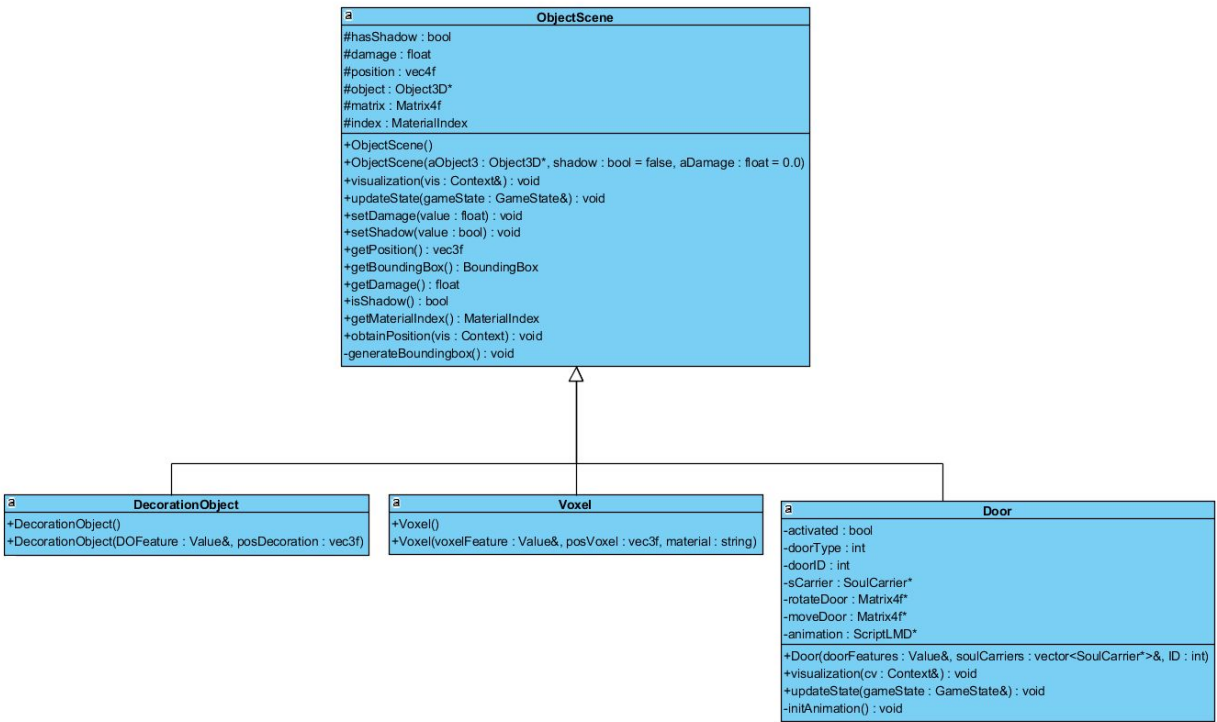


Figura 11 Diagrama de clases de objeto en la escena

12. Escena

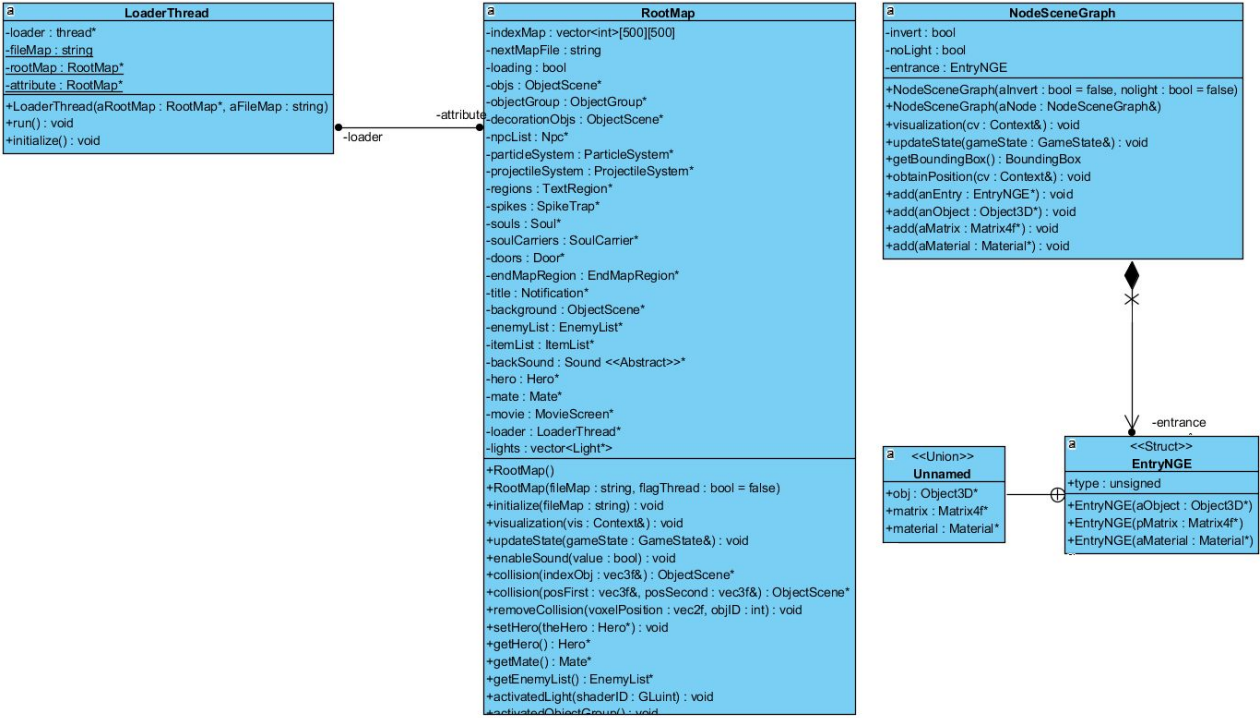


Figura 12 Diagrama de clase de escena

13. Sistema de puertas

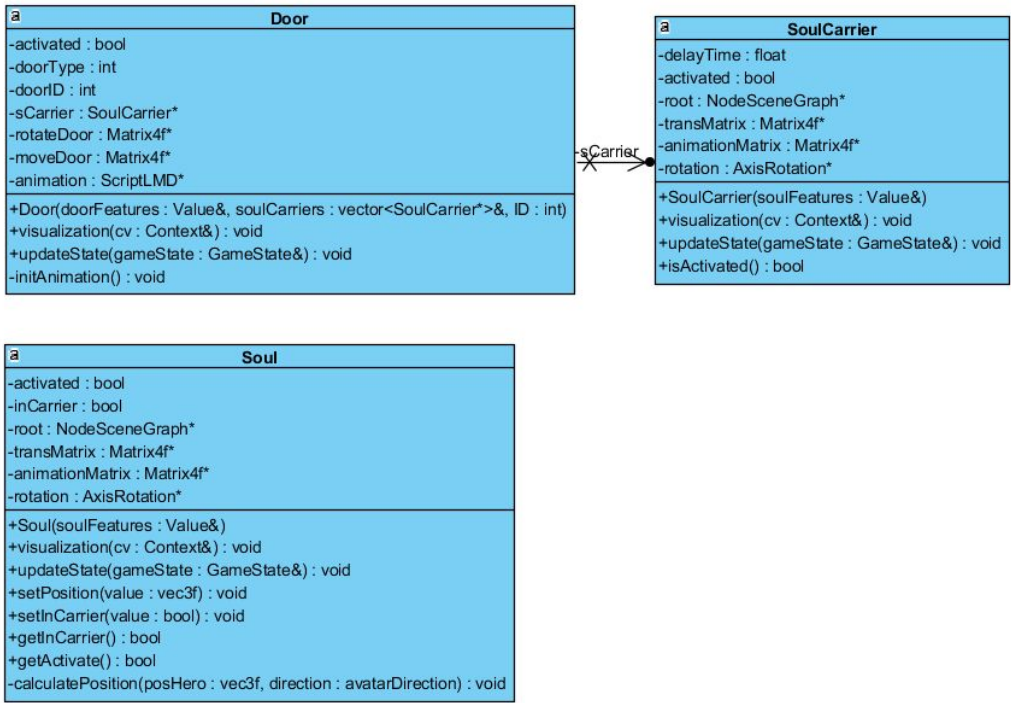


Figura 13 Diagrama de clases del sistema de puertas

14. Sistema de partículas/proyectiles y objetos

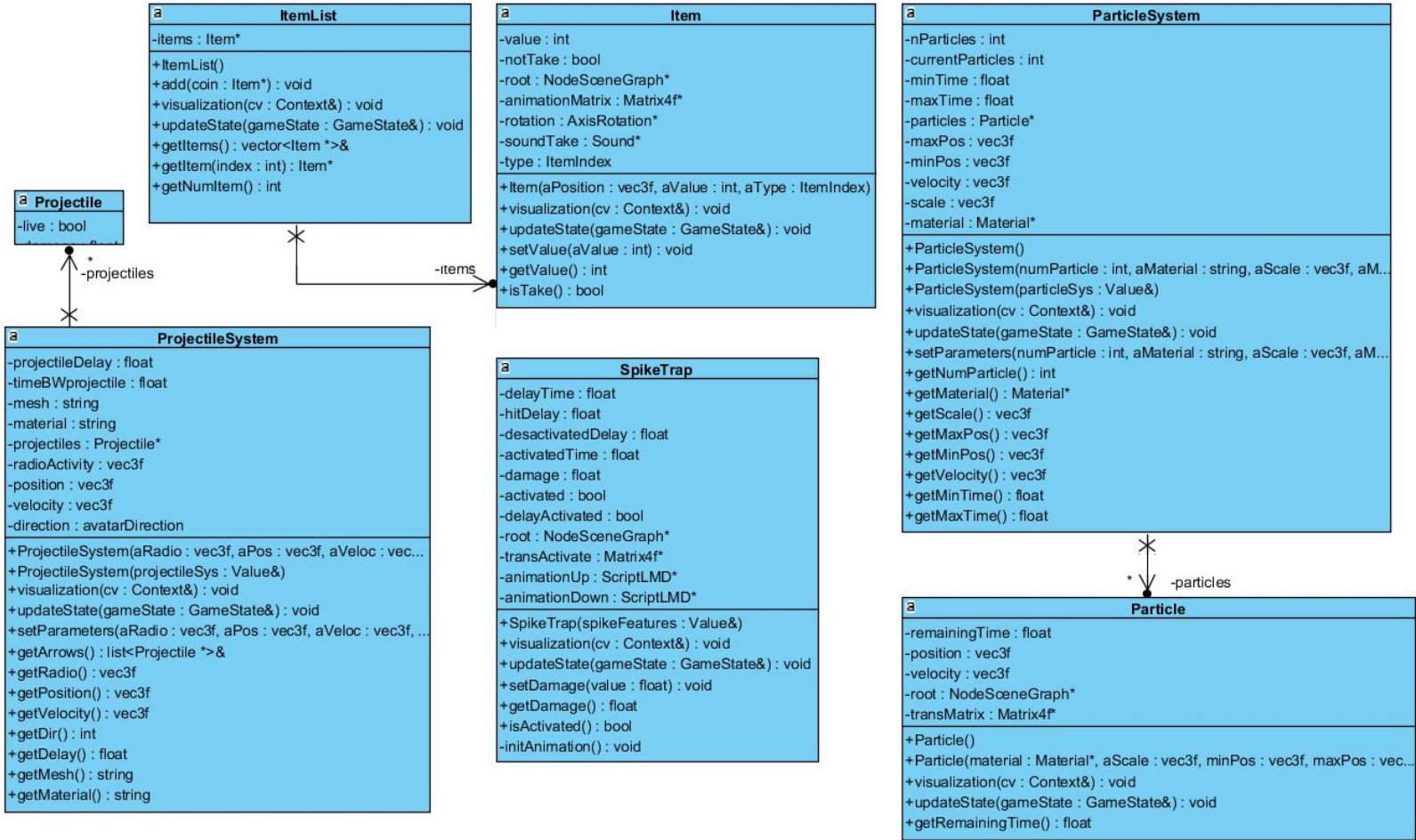


Figura 14 Diagrama de clases de sistema de partículas/proyectiles y sistema de objetos

15. Malla

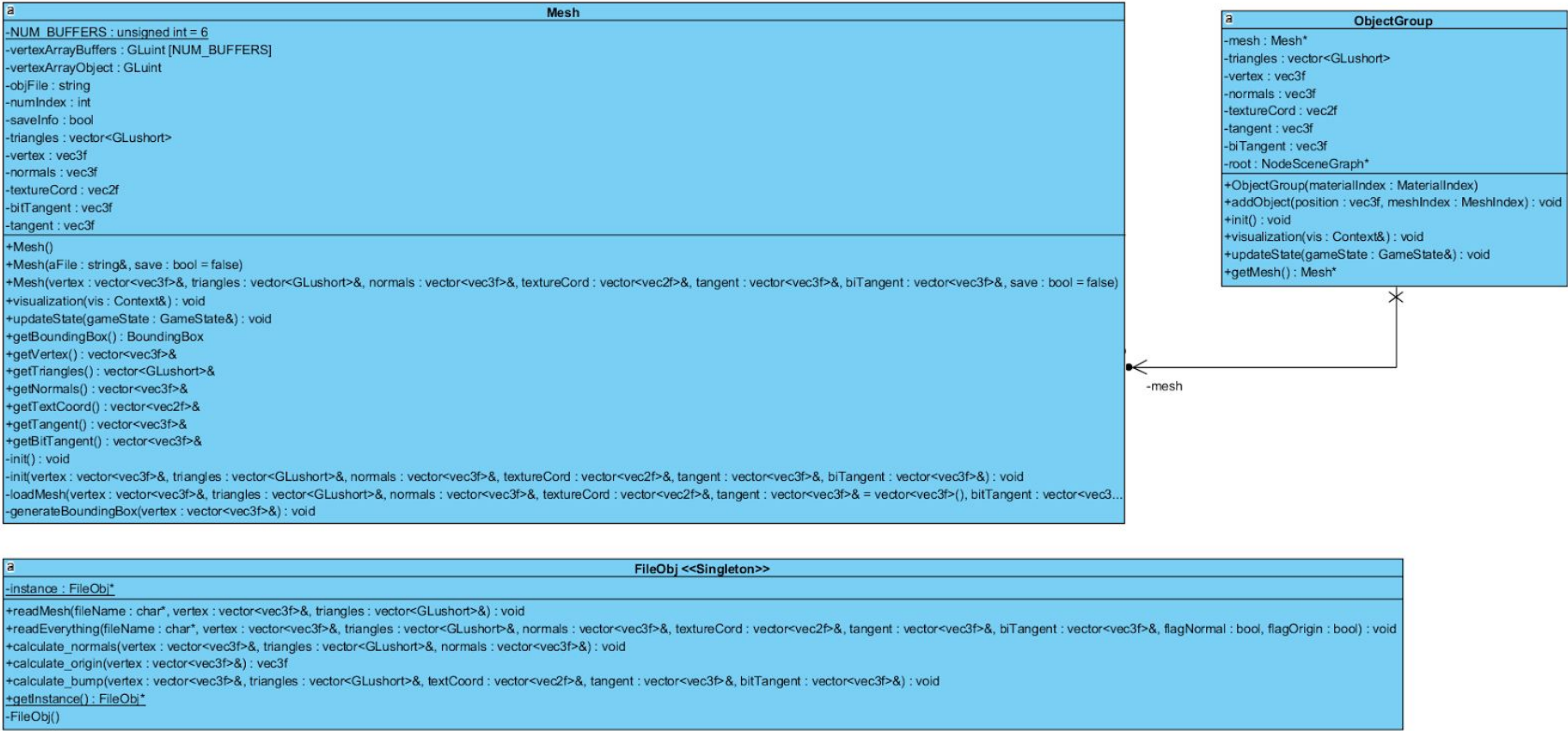


Figura 15 Diagrama de clases de sistema Mesh

16. Material

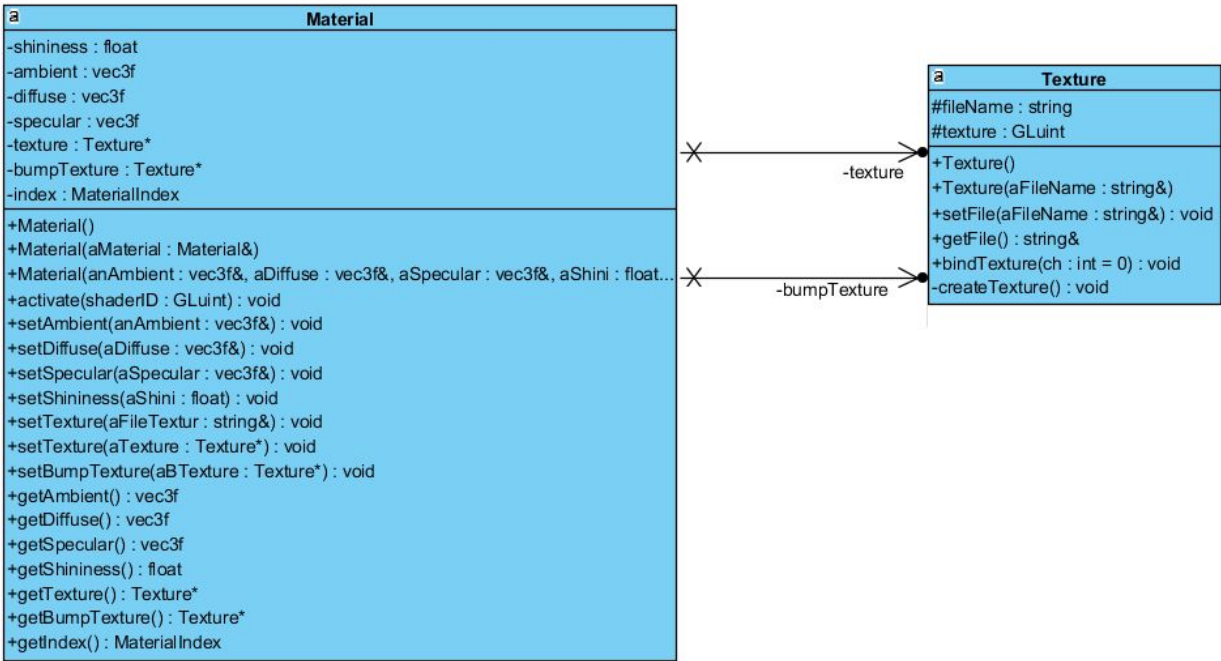


Figura 16 Diagrama de clases de Material

17. Iluminación

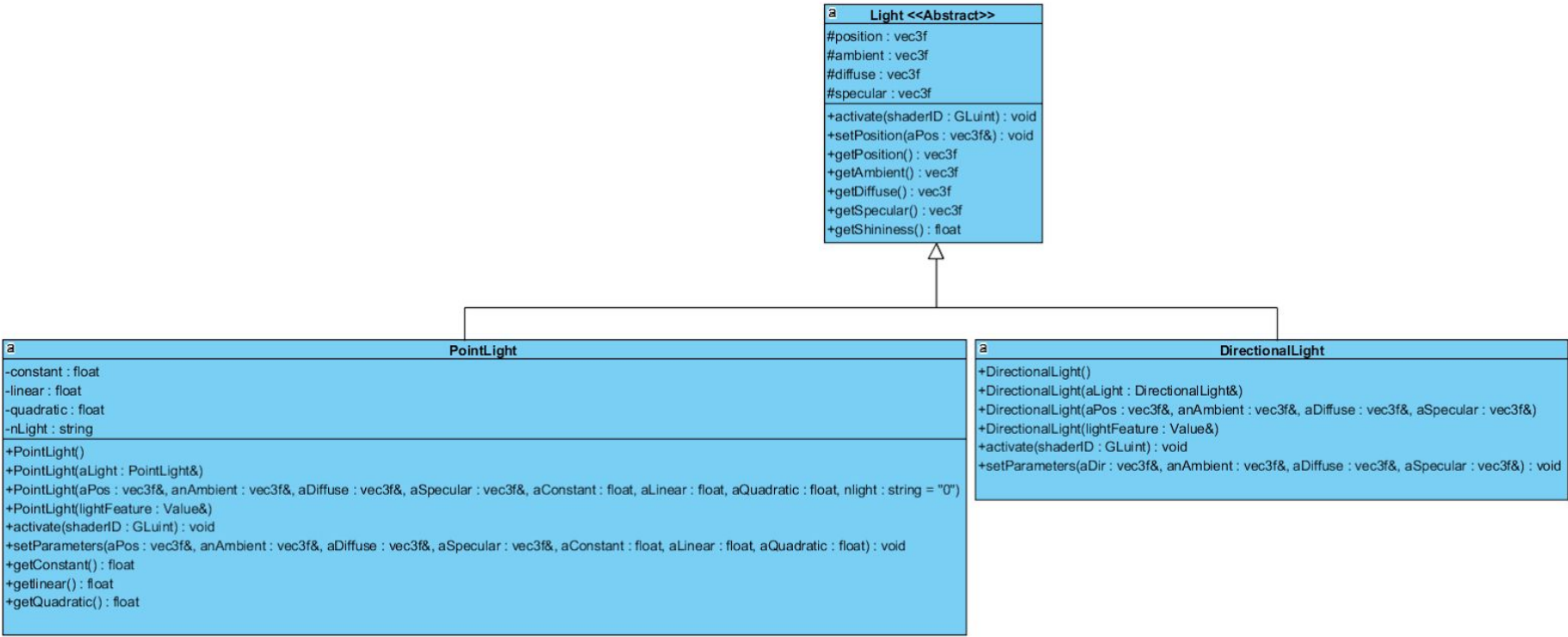


Figura 17 Diagrama de clases de iluminación

18. Sombras

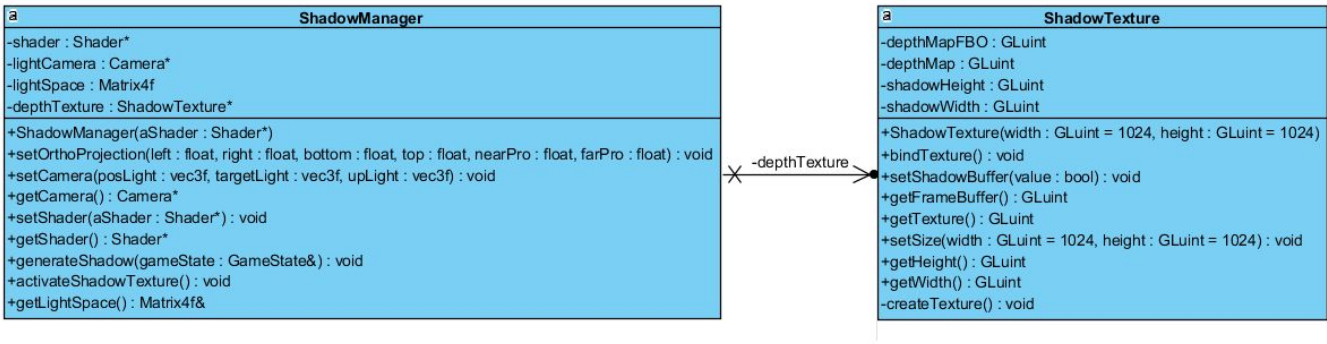


Figura 18 Diagrama de clases de sombras

19. Región

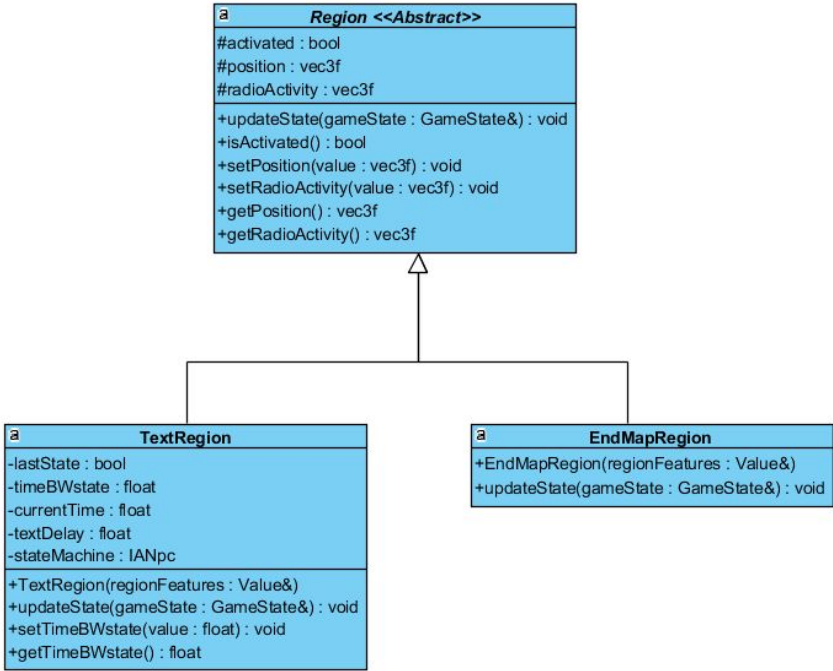


Figura 19 Diagrama de clases de región

20. Sonido

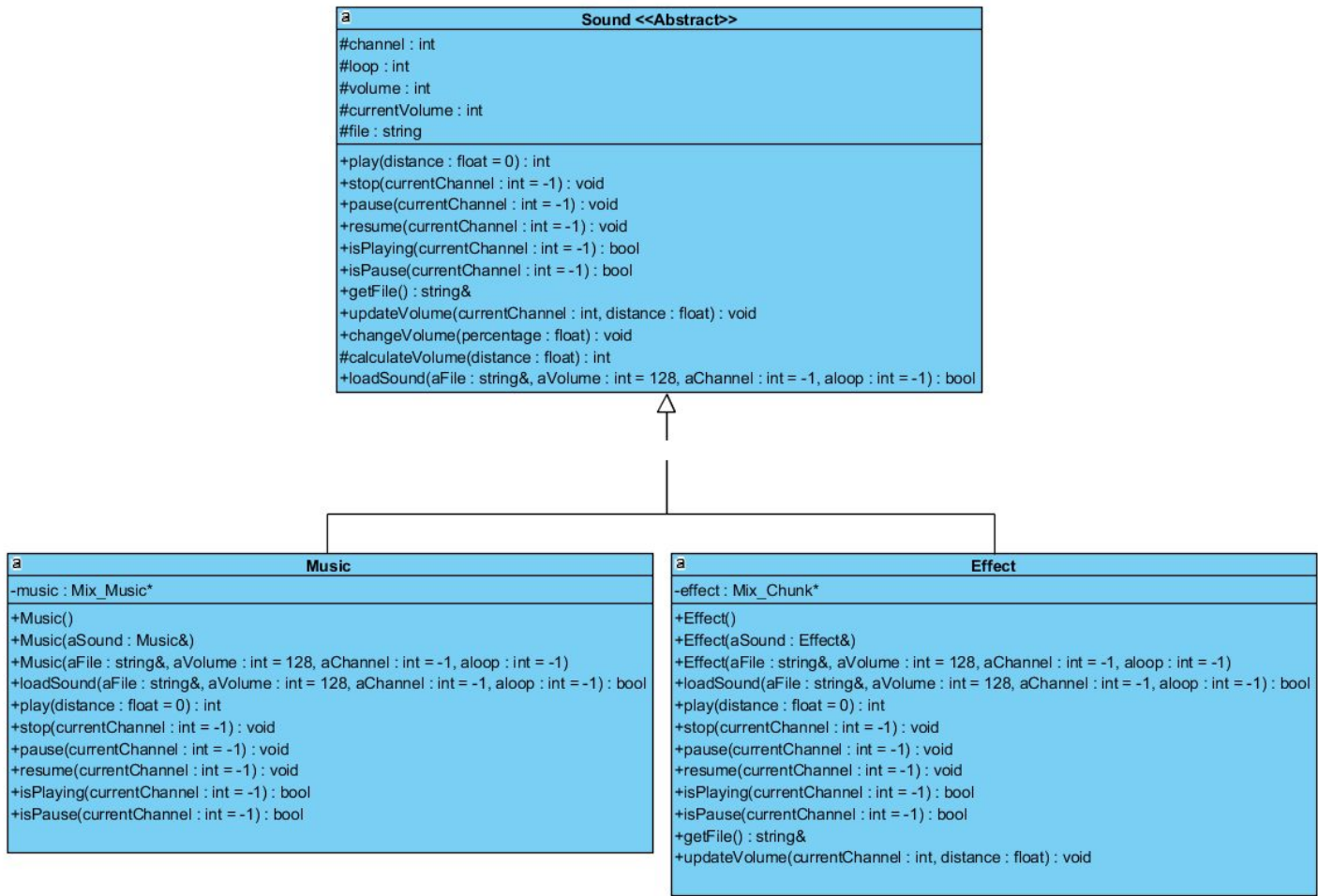


Figura 20 Diagrama de clases de sonido

21. Colección

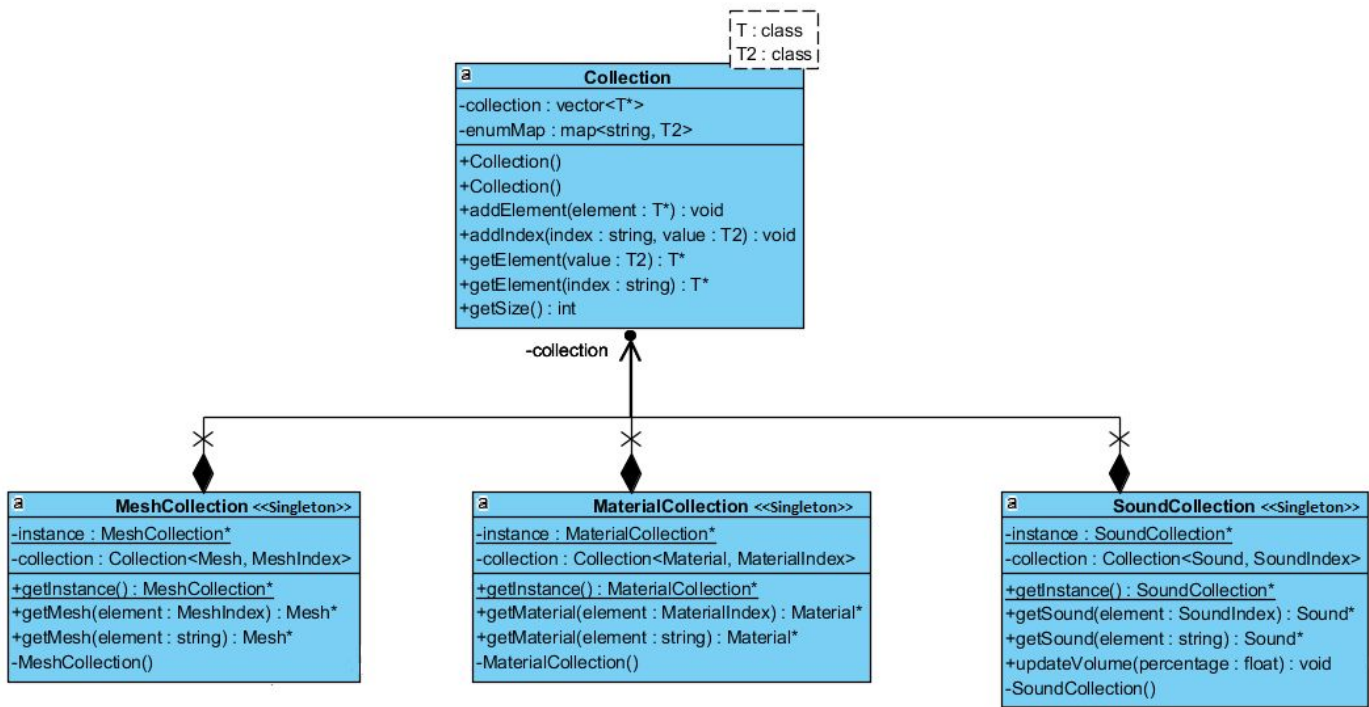


Figura 21 Diagrama de clases de colección

22. Gestor de ficheros

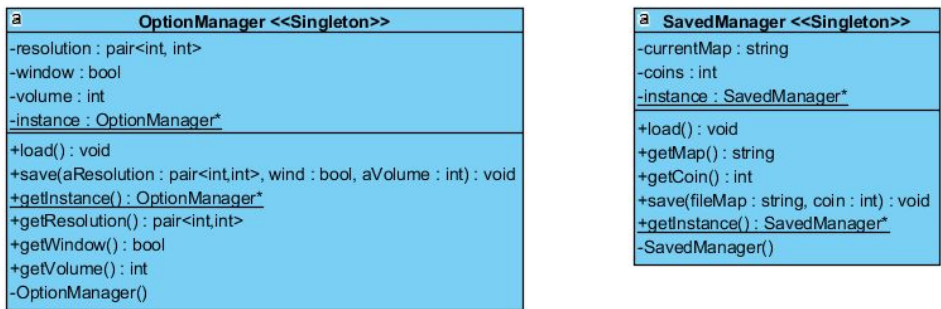


Figura 22 Diagrama de clases de Gestor de ficheros

23. Controlador

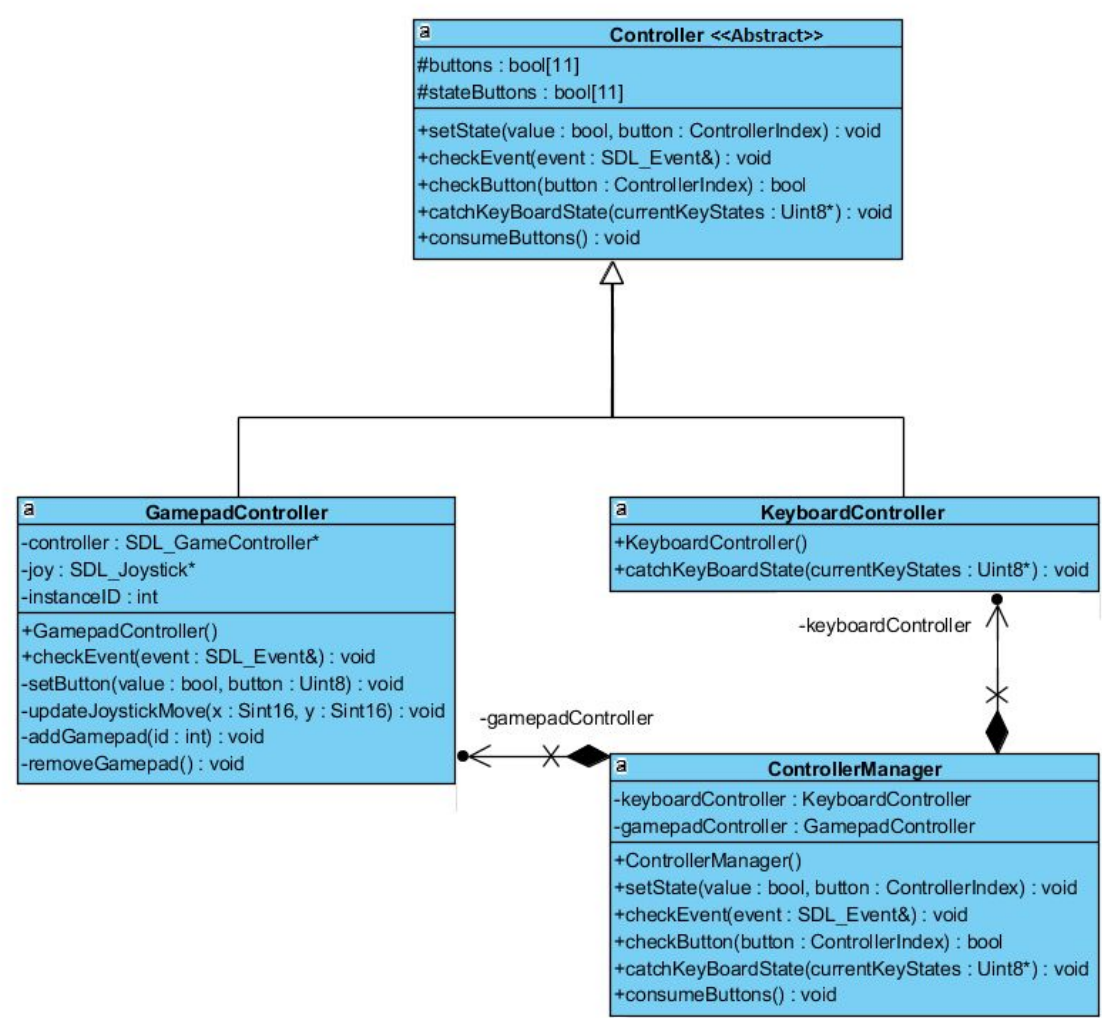


Figura 23 Diagrama de clases de controlador