

UNIVERSIDADE DA CORUÑA

The Last of Shendralar

Miguel Mouriño García <miguel.mourino.garcia@udc.es>

Gilberto Plaza González <gilberto.plaza@udc.es>

Bruno Cabado Lousa <bruno.cabado@udc.es>

David Maseda Neira <david.maseda@udc.es>

Julián Silveira Carnota <j.silveira@udc.es>

Índice:

| | |
|--|----------|
| 1 - Desarrollo artístico | 4 |
| 1.1 - Antecedentes | 4 |
| 1.1.1 - Ambientación | 4 |
| 1.1.2 - Historia | 5 |
| 1.2 - Personajes | 5 |
| 1.3 - Otras características de la ambientación | 6 |
| 1.3.1 - Objetos destacados | 6 |
| 1.3.2 - Lugares | 6 |
| 1.4 - Guión | 7 |
| 1.4.1 Inicio, juego 2D | 7 |
| 1.4.2 Continuación, juego 3D | 7 |
| 2 - Desarrollo técnico | 8 |
| 2.1 - Videojuego en 2D | 8 |
| 2.1.1 - Descripción | 8 |
| 2.1.1.1 - Personajes | 8 |
| 2.1.1.1.1 - Protagonista | 8 |
| 2.1.1.2 - Enemigos | 9 |
| 2.1.1.2.1 - Warmond | 9 |
| 2.1.1.2.2 - Disas | 9 |
| 2.1.1.2.3 - Ludwig | 10 |
| 2.1.1.3 - Diseño | 10 |
| 2.1.2 - Escenas | 10 |
| 2.1.2.1 - Escena 1: Menu de Inicio | 11 |
| 2.1.2.1.1 - Descripción | 11 |
| 2.1.2.2 - Escena 2: Farm | 11 |
| 2.1.2.2.1 - Descripción | 11 |
| 2.1.2.2.2 - Modelo | 11 |
| 2.1.2.3 - Escena 3: Lindisfarne Castle | 13 |
| 2.1.2.3.1 - Descripción | 13 |
| 2.1.2.3.2 - Modelo | 13 |
| 2.1.2.4 - Escena 4: Trisquel Forest | 13 |
| 2.1.2.4.1 - Descripción | 13 |
| 2.1.2.4.2 - Modelo | 13 |
| 2.1.2.5 - Escena 5: Menú de Pausa | 13 |
| 2.1.2.5.1 - Descripción | 13 |
| 2.1.2.6 - Escena 6: Menú de Muerte | 13 |
| 2.1.2.6.1 - Descripción | 13 |
| 2.1.2.7 - Escena 7: Menú Final | 14 |
| 2.1.2.7.1 - Descripción | 14 |
| 2.1.3 - Aspectos destacables | 14 |

| | |
|---|----|
| 2.1.3.1 - Sistema de Diálogos | 14 |
| 2.1.3.2 - Cámara | 14 |
| 2.1.3.3 - Sistema de creación de niveles | 14 |
| 2.1.3.4 - Gestor de sonido | 14 |
| 2.1.3.5 - Sistema debug de niveles | 15 |
| 2.1.4 - Manual de usuario | 15 |
| 2.2 - Videojuego en 3D | 17 |
| 2.2.1 - Descripción | 17 |
| 2.2.1.1 - Personajes | 17 |
| 2.2.1.1.1 - Protagonista | 17 |
| 2.2.1.2 - Enemigos | 17 |
| 2.2.1.2.1 - Esqueletos | 17 |
| 2.2.1.3 - Diseño | 18 |
| 2.2.2 - Escenas | 18 |
| 2.2.2.1 - Escena 1: Menú de Inicio | 19 |
| 2.2.2.1.1 - Descripción | 19 |
| 2.2.2.1.2 - Modelo | 19 |
| 2.2.2.2 - Escena 2: Coliseo 1 | 19 |
| 2.2.2.2.1 - Descripción | 19 |
| 2.2.2.2.2 - Modelo | 20 |
| 2.2.2.2.2.1 - Inteligencia Artificial | 23 |
| 2.2.2.2.2.2 - Máquina de estados | 25 |
| 2.2.2.2.2.3 - Sistema de colisiones | 25 |
| 2.2.2.2.2.4 - Sistema de sonido | 26 |
| 2.2.2.2.2.5 - Sistema de cámara | 27 |
| 2.2.2.2.2.6 - Sistema de animación | 28 |
| 2.2.2.2.2.7 - Sistema de carga/descarga de escena | 29 |
| 2.2.2.2.2.8 - Jugador y enemigos | 29 |
| 2.2.2.3 - Escena 3: Menú de Pausa | 30 |
| 2.2.2.3.1 - Descripción | 30 |
| 2.2.2.4 - Escena 4: Menú de Muerte | 30 |
| 2.2.2.4.1 - Descripción | 30 |
| 2.2.2.5 - Escena 5: Créditos | 30 |
| 2.2.2.5.1 - Descripción | 30 |
| 2.2.3 - Aspectos destacables | 30 |
| 2.2.3.1 - Sistema de Menús | 30 |
| 2.2.3.2 - Cámara | 31 |
| 2.2.3.3 - Bugs | 31 |
| 2.2.3.4 - Gestor de sonido, efectos y música | 31 |
| 2.2.3.5 - Modelado de objetos y animación | 31 |
| 2.2.4 - Manual de usuario | 32 |

1 - Desarrollo artístico

1.1 - Antecedentes

1.1.1 - Ambientación

Fantasía en la Alta Edad Media. Siglo VIII. El mundo ha quedado dividido en reinos gobernados por distintos tiranos. La única esperanza es Doan, el último reino libre de la época, reino vecino de Shendralar.

La península de Shendralar se encuentra totalmente rodeada por el océano. No se ve más tierra observando el horizonte, el único lugar conocido distinto a Shendralar es Doan, el último reino libre que se encuentra al otro lado del Desierto de Kildare. Muchos han intentado escapar de la península por el desierto pero nadie ha vuelto para contarlo, nadie sabe si alguien lo habrá conseguido, pero lo seguro es que de haberlo hecho, no habría querido regresar.



Mapa de Shendralar

1.1.2 - Historia

En la actualidad el reino de Shendralar se encuentra gobernado por “Tux” un tirano esclavista que reprime a la población del reino mediante el control de la misma. Para ello cada 10 años envía a un grupo de mercenarios llamados “Youhei” los cuales van por los distintos poblados del reino diezmado a la población, y apresando a los ciudadanos más hábiles para utilizarlos como carnaza en la arena de la Capital.

Se acaban de cumplir los 10 años desde la última purga y todas las gentes de los distintos pueblos permanecen atemorizadas, pendientes por si en algún momento necesitan escapar. La tensión en el aire se puede notar cuando se habla con cualquiera, todos saben su probable futuro pero nadie quiere asumirlo.

Algunos dicen estar dispuestos a luchar contra el tirano pero todos los que lo han intentado han acabado muertos. Otros fantasean con la posibilidad de que no haya más purgas ya que está aún no ha llegado y esperan que no llegue. Un futuro incierto azota Shendralar, actualmente sólo el tiempo puede escribir la historia de nuestro reino.

1.2 - Personajes

Protagonista: Campesino que intentará escapar de una purga. La necesidad de escapar le coje por sorpresa ya que era uno de los pocos habitantes que confiaba en que no hubiese una nueva purga. Necesita escapar rápidamente para salvar su vida y por ello sus vestimentas son las que llevaba en ese momento, su camiseta de cuero y pantalones de tela roja de trabajo que solía utilizar para recolectar el trigo y las remolachas de sus plantaciones.

Disas: Maga enemiga, que se unió al bando de nuestro tirano cuando en su aldea quisieron matarla por bruja, y no iban muy desencaminados. Ser pelirroja era un claro síntoma de brujería y algunos de los aldeanos aseguraban haberla visto por las noches con su túnica marrón creando fuego de la nada. Ahora sirve a nuestro tirano y se rumorea que ha jurado acabar con todos aquellos que la hicieron sufrir cuando sólo era una niña.

Warmond: Nigromante enemigo, nadie sabe su procedencia. Un día llegó a servir al tirano y nada más se sabe, parece haber venido del mundo de los muertos con su tez grisácea. Los pocos que le han visto y sobrevivido para contarlo aseguran que sus ojos rojos no son de este mundo. Parece que la sangre lo llama, quizás servir al tirano sólo sea un método para poder seguir matando sin represalias.

Ludwig: Guerrero enemigo nacido en Shandalar. Se rumorea que para unirse al tirano asesinó él mismo a todo su pueblo antes de que llegaran a purgarlo como ofrenda al tirano. Tux le dio una armadura de oro por su hazaña. Nadie que haya visto su rostro sigue vivo, excepto los de su bando.

Tux: Tirano. Desde su nacimiento su padre le inculcó los valores de la violencia y el desprecio de los de la clase inferior. Por ello, cuando heredó la corona, pocos tenían la esperanza de que hubiese un cambio en el modo de vida de las gentes, que ya estaban cansadas tras años de purgas y violencia. Delgado y alto, Tux siempre tiene una mirada astuta y fría.

Mano derecha: Personaje desconocido, siempre al lado de Tux. Nadie sabe nada de él, únicamente que fue el primero en unirse a Tux para llevar a cabo las plagas, lo hizo sin dudarlo. Los pocos que dicen haberlo visto, aseguran que siempre lleva su túnica verde, junto con una capa negra que le llega hasta los pies. Siempre lleva la cabeza tapada con lo que podría ser un gorro de mago marrón por lo que nadie ha podido describir su rostro aún. Uno de estos hombres que jura haberlo visto dice que mató a todo su pueblo usando su lanza azul.

1.3 - Otras características de la ambientación

1.3.1 - Objetos destacados

Horca: Good 'ol reliable. Encontrada en la granja. Efectiva, pero no muy elegante. Pesada y lenta pero con un poder destructivo alto.

Espada corta: Espada bastarda roma, muy usada. Agil y rapida pero no muy efectiva a la hora de hacer daño.

Hojas Gemelas: Dagas ornamentadas blandidas por Warmond el nigromante. Muy afiladas y rápidas, y con un poder destructivo aceptable.

Espada Luz de Luna: Legendario gran espadón empuñado por Ludwig. Lento, pero con una potencia devastadora.

1.3.2 - Lugares

Granja: Granja sencilla y antigua, ahora ya vacía tras el ataque de los mercenarios. Cerca de una costa, y con unas construcciones sencillas, en su día albergaba vida y actividad gracias a la gran cantidad de animales que en ella habitaban.

Talaan: Capital de Shendralar, donde reside la clase noble. Llena de grandes edificios y casas señoriales, también cuenta con barrios bajos en las afueras, donde los sirvientes de los nobles residen. Es una ciudad de contrastes, en donde el lujo es vecino del hambre.

Doan: Reino vecino de destino. Un pueblo libre donde no existe la purga. O al menos eso cuentan los rumores. Estos también hablan sobre las bondades del lugar, en el cual la vivienda, la comida y el agua están garantizados y el trabajo y la salud son abundantes. Limpio y sin violencia, se podría decir que es el cielo en la tierra.

Bosque Trisquel: Bosque antiguo del reino de Shendralar. Grandes y altos árboles lo pueblan, más antiguos que el tiempo según algunos. En su frondosidad muchos se han perdido y pocos han regresado para contar las maravillas que en él habitan, como plantas curativas y aguas que dan un poder sobrehumano.

Lago de Kells: Lago más grande del reino, fuente de alimento y riqueza. De una profundidad desconocida, nadie sabe con certeza todo lo que oculta esta masa de agua. Algunos dicen que en ella habitan criaturas antiquísimas como krakens, las cuales se encuentran un un profundo letargo.

Desierto de Kildare: Desierto de la frontera con el reino de Doan, pocos consiguen cruzarlo. Y el motivo no es otro que las altas, altísimas temperaturas que se alcanzan de hasta 60°. Por ello pocos son los seres que aquí habitan, siendo en su mayoría animales e insectos pequeños nada amigables.

Castillo de Lindisfarne: Castillo ya abandonado, usado en guerras pasadas. Se comenta que si se está el tiempo suficiente en sus alrededores, se pueden escuchar los gritos de los que antaño dieron sus vidas en guerras acaecidas en este lugar. Tétrico y en ruinas, pocos lo visitan.

El Coliseo de la Capital: Coliseo donde el tirano Tux, hace luchar a sus esclavos. Es una gran extensión en forma de círculo en donde los combatientes se sitúan en el centro, en un suelo terroso, para pelear. Mientras, en las gradas, altas y confortables, los nobles y Tux se divierten con el espectáculo.

1.4 - Guión

1.4.1 Inicio, juego 2D

El personaje principal es un granjero de un pueblo remoto. Cuando llegan las noticias de una próxima purga, y el rumor de la llegada de los mercenarios se extiende. Warmond llega al pueblo para acabar con la gente por órdenes del tirano Tux, nuestro protagonista lo busca, lo encuentra, se llena de valentía y empieza una batalla contra él para salvar su vida. Tras acabar con él, nuestro protagonista inicia su huida hacia el Este. El protagonista empieza su camino hacia las tierras libres de Doan.

En el camino de nuestro protagonista, al llegar al castillo de Lindisfarne, se encuentra con otro secuaz del Tirano que viene a taponar el camino y a acabar con su vida, esta vez se trata de la maga Disas. Nuestro personaje se enfrenta a ella en una ardua batalla de la que terminará saliendo victorioso. Escapa del castillo ya que ya no es un lugar seguro y huye hacia el oeste. Nuestro protagonista prosigue su camino hacia Doan.

Finalmente nuestro protagonista se encontrará a Ludwig, este es demasiado fuerte como para intentar luchar contra él, por lo que nuestro personaje empieza a escapar buscando algún sitio donde poder dejarlo atrás, al final nuestro personaje se queda sin alternativas para escapar, donde lo encontrará Ludwig y le asestará un espadazo.

1.4.2 Continuación, juego 3D

Nuestro protagonista se encuentra esclavizado por Tux en Talaan, este al ver que ha sido capaz de derrotar a sus mercenarios, decide proponerle un trato, luchará en el coliseo de la capital contra oleadas de monstruos (en este caso esqueletos) en un macabro juego en el que le promete la libertad si vence. Lo que no sabe nuestro protagonista es que esas oleadas nunca acaban, y nunca acabarán, condenando a luchar hasta el fin de sus energías.

2 - Desarrollo técnico

2.1 - Videojuego en 2D

2.1.1 - Descripción

Juego de combate en tiempo real con cámara proyección $\frac{3}{4}$, ambientado en la edad media. El jugador controla a el personaje protagonista e irá avanzando por diferentes fases derrotando a diferentes enemigos.

Estos enemigos tienen distintas mecánicas en cuanto a la lucha, hacen daño cuando colisionan, crean otros enemigos, usan espada o castean llamas.

Las distintas batallas se desarrollan en entornos diferentes, cada batalla en uno diferente, con su ambientación propia, estos entornos disponen de elementos como puede ser habitáculos, flora o terreno.

2.1.1.1 - Personajes

Todos los personajes con animaciones, tienen un sprite interno de la clase BodyHitbox (src/characters.py), que es el encargado de detectar las colisiones. Tomamos esta decisión debido a que los sprites con los que contábamos en ciertas animaciones no estaban alineados y eran de tamaños dispares.

2.1.1.1.1 - Protagonista



- Modo de combate: Espada
- Puntos de vida: 100
- Daño: 20
- Cuenta con 4 hitboxes que se activan y registran cuando golpea con la espada (ocultos ingame)
 - Hitbox superior e inferior:



- Hitbox superior e inferior:



2.1.1.2 - Enemigos

2.1.1.2.1 - Warmond



- Modo de combate: Colisión con el jugador
- Puntos de vida: 200
- Daño: 20
- Habilidad: Spawn de Esqueletos
 - Esqueletos:



- Modo de combate: Colisión con el jugador
- Puntos de vida: 50

2.1.1.2.2 - Disas



- Modo de combate: Colisión con el jugador
- Puntos de vida: 200
- Daño: 20
- Habilidad: Casteo de llamas



2.1.1.2.3 - Ludwig



- Modo de combate: colisión con el jugador
- Puntos de vida: 200
- Daño: 20
- Usa el mismo esquema de hitbox que el Protagonista

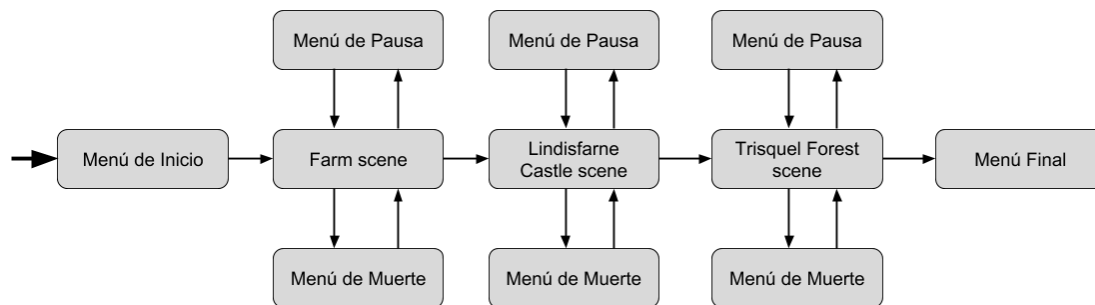
2.1.1.3 - Diseño

El jugador puede moverse en cuatro ejes, arriba, abajo, izquierda y derecha, esto se controlará mediante las teclas de dirección (flechas) del teclado. También se puede atacar con el personaje, esto se efectuará mediante la pulsación de la tecla espacio durante el juego. En caso de estar dentro de un diálogo, la tecla espacio se usará para ir avanzando por el mismo. El personaje dispone de vida la cual se mostrará en la parte inferior izquierda de la pantalla. Los enemigos también disponen de su vida propia, la cual se mostrará en la parte superior de los mismos. El objetivo de las fases es hacer que la vida del enemigo principal llegue a 0, para que se nos abra el camino hacia la nueva zona, la cual se encontrará inaccesible hasta este suceso. En el caso de que nuestra vida llegue a 0, perderemos la partida. Al pasar esto se nos abrirá un menú desde el cual podremos salir del juego o reintentar la fase en la que nos encontramos. El juego dispone de un menú de pausa, accesible con la tecla Escape (ESC) durante el juego. Desde esta pantalla, podremos tanto reanudar la partida como salir del juego.

2.1.2 - Escenas

Dentro de las escenas hay dos tipos de las mismas: escenas de menú y escenas de acción. En las primeras es jugador interactúa con una serie de opciones, y elige una de ellas de acuerdo con lo que desee. En las de acción se desarrolla el juego en sí: diálogos, combate, movimiento, etc... Las escenas de menú tienen una serie de elementos comunes mientras que las de acción tienen otros, a mayores de elementos comunes a ambas.

Diagrama de Flujo de Escenas:



2.1.2.1 - Escena 1: Menu de Inicio

2.1.2.1.1 - Descripción

Nos encontramos ante un menú con dos alternativas, empezar a jugar al juego o salir del mismo, podemos movernos entre ellas con las teclas de dirección(flechas), arriba, abajo, del teclado o mediante el ratón, para seleccionar una de las alternativas basta con pulsar intro (ENTER) sobre la opción seleccionada o con el click izquierdo del ratón.

2.1.2.2 - Escena 2: Farm

2.1.2.2.1 - Descripción

La acción comienza en un pueblo costero formado por un par de casas y tiendas.

Se abre un diálogo que nos indica la situación en la que se encuentra el reino y el lugar donde se encuentra el primer enemigo, Warmond.

Al finalizar el diálogo el jugador podrá moverse, y buscar a Warmond al encontrarse con él, este abrirá un diálogo y al finalizarlo comenzará la batalla.

Al derrotar al enemigo, al jugador se le abrirá el camino hacia la siguiente zona, huyendo hacia el Este.

2.1.2.2.2 - Modelo

Patrón Estrategia: usado en las cámaras (camera_simple, camera_complex) se menciona posteriormente en aspectos destacables.

Patrón Singleton: usado en el resource manager.

Patrón Pool de objetos: se utiliza en algunos jefes del juego que crean otros enemigos, véase Warmond el Nigromante que invoca esqueletos o la Maga Disas que invoca bolas de fuego. Puede encontrarse en las clases de estos jefes dentro del archivo src/character.py del proyecto.

Diagrama de clases sprite (para más detalle, los diagramas están incluidos en la carpeta doc del juego)

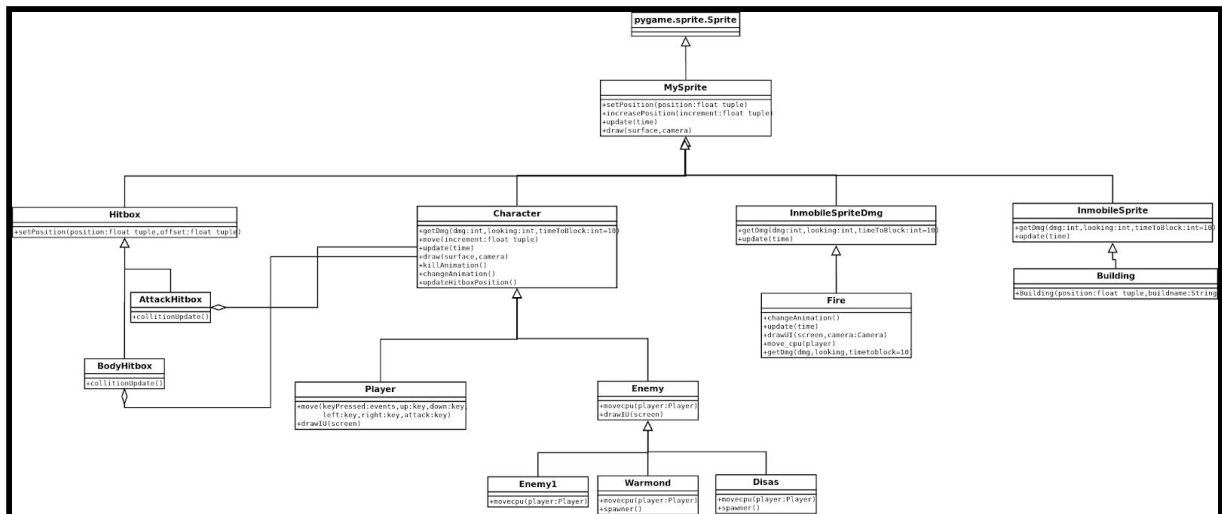
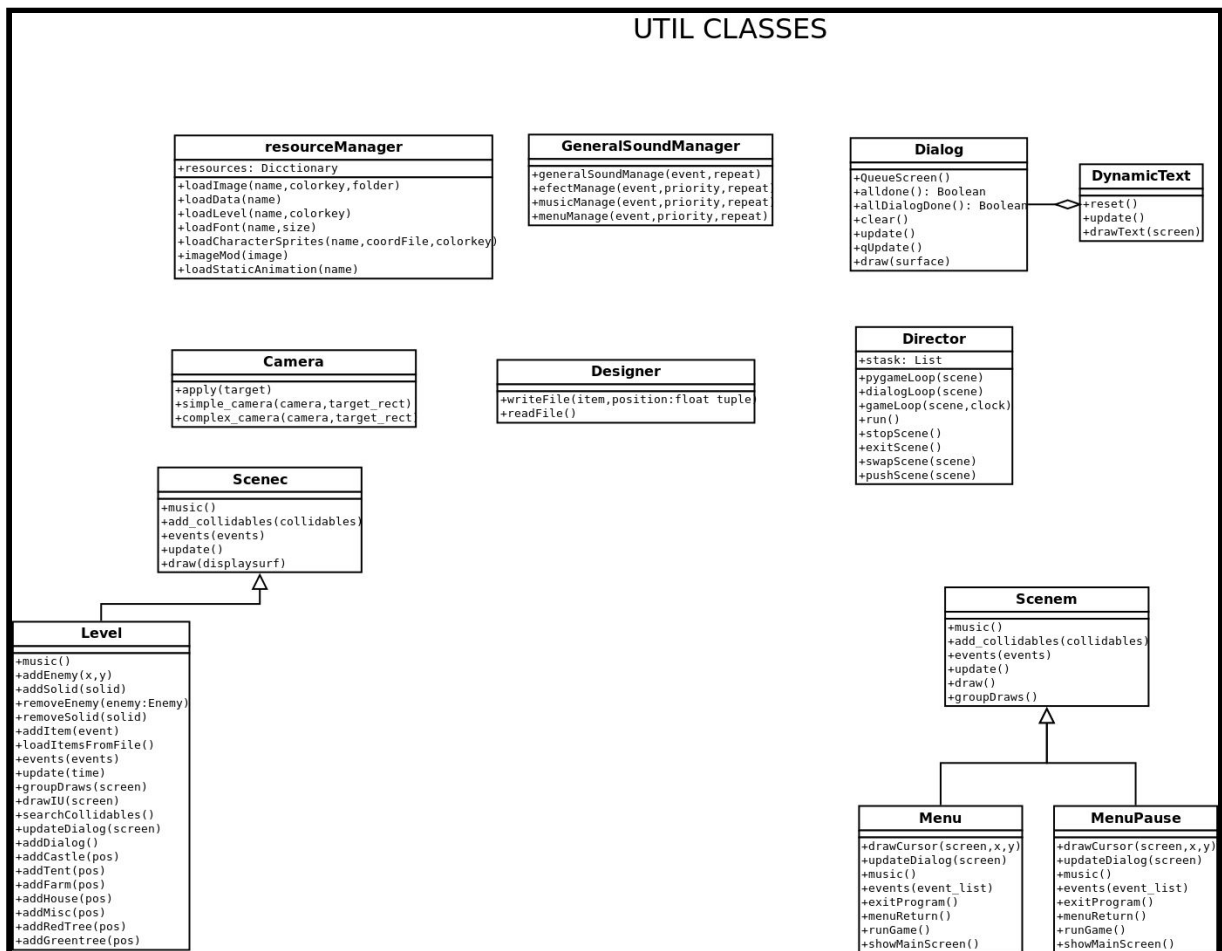


Diagrama de clases generales (para más detalle, los diagramas están incluidos en la carpeta doc del juego)



2.1.2.3 - Escena 3: Lindisfarne Castle

2.1.2.3.1 - Descripción

La acción comienza en el castillo de Lindisfarne, nuestro protagonista caminará hacia el castillo donde se encontrará con la maga Disas está abriendo un diálogo donde le dice a nuestro personaje que le ha cortado el paso y que viene a acabar con él por órdenes del Tirano, al finalizar este diálogo comenzará la batalla contra Disas y sus bolas de fuego. Al derrotar al enemigo, al jugador se le abrirá el camino hacia la siguiente zona, huyendo hacia el Oeste.

2.1.2.3.2 - Modelo

Para esta escena se aplican los mismos modelos y clases que la escena 2

2.1.2.4 - Escena 4: Trisquel Forest

2.1.2.4.1 - Descripción

La acción comienza en el bosque Trisquel, al llegar a este nivel se nos abre un diálogo informándonos de que el guerrero Ludwig viene a por nosotros pero que no peleemos porque es demasiado fuerte como para enfrentarse a él, el personaje debe escapar de Ludwig, al final nuestro personaje acaba en un callejón sin salida, al que llega Ludwig y le asesta un espadazo.

2.1.2.4.2 - Modelo

Para esta escena se aplican los mismos modelos y clases que la escena 2

2.1.2.5 - Escena 5: Menú de Pausa

2.1.2.5.1 - Descripción

Nos encontramos ante un menú con dos alternativas, reanudar la partida o salir del juego, podemos movernos entre ellas con las teclas de dirección(flechas), arriba, abajo, del teclado o mediante el ratón, para seleccionar una de las alternativas basta con pulsar intro (ENTER) sobre la opción seleccionada o con el click izquierdo del ratón.

2.1.2.6 - Escena 6: Menú de Muerte

2.1.2.6.1 - Descripción

Nos encontramos ante un menú con dos alternativas, reintentar el nivel en el que nos encontramos o salir del juego, podemos movernos entre ellas con las teclas de dirección (flechas), arriba, abajo, del teclado o mediante el ratón, para seleccionar una de las alternativas basta con pulsar intro (ENTER) sobre la opción seleccionada o con el click izquierdo del ratón.

2.1.2.7 - Escena 7: Menú Final

2.1.2.7.1 - Descripción

Nos encontramos ante un menú con una única alternativa, The End, al pulsar sobre ella ya sea con el click izquierdo del ratón como con la tecla intro (ENTER) del teclado se cerrará el juego, es la pantalla final.

2.1.3 - Aspectos destacables

2.1.3.1 - Sistema de Diálogos

Implementación de un sistema de diálogos con carga progresiva de texto.

El juego cuenta con un sistema de diálogos de estilo RPG clásico.

Los diálogos avanzan por pantallas y están animados, pudiéndose configurar número de líneas por pantalla, número de pantallas y fuente tipográfica del diálogo. Estos diálogos pueden ser activados en un “modo diálogo” que pausa todas las interacciones del juego hasta que el diálogo concluya.

Puede encontrarse en el directorio src/dialog del proyecto.

2.1.3.2 - Cámara

El sistema de cámara consiste en una cámara simple con dos estrategias implementadas, ambas siguiendo al personaje, pero una de ellas teniendo en cuenta los límites del nivel. El sistema de cámara es extensible y provee la posibilidad de crear nuevas estrategias complejas.

Una vez determinada la estrategia de cámara, el objeto cámara aplica sus coordenadas mediante el método “apply” que modifica la posición del objeto que se quiera desplazar.

Puede encontrarse en el archivo src/game/camera.py del proyecto.

2.1.3.3 - Sistema de creación de niveles

El sistema de creación de niveles es simple pero efectivo. Mediante el uso de imágenes PNG que representan por colores la estructura del nivel, el diseño de niveles se puede hacer con un programa de edición gráfica. Para los props y otros assets, los niveles tienen métodos específicos que permiten añadir elementos de manera sencilla teniendo unas coordenadas.

Puede encontrarse en el archivo src/scene/serializer.py del proyecto.

2.1.3.4 - Gestor de sonido

El juego cuenta con un gestor de sonido independiente del mismo, esto es, el control de sonido se hace en un módulo independiente que se carga en el juego al inicializarlo. De esta manera se pueden cambiar los sonidos del juego cambiando simplemente de módulo

de sonido, o lo que es lo mismo, cambiar un archivo .py por otro. Además el gestor incorpora tres submodulos de sonido, cada uno para cada tipo de audio del juego (música, efectos, menu). Así si sólo se quiere cambiar los sonidos de uno de estos tres apartados, bastaría con cambiar el submódulo.

Puede encontrarse en la carpeta /src/sound del proyecto.

2.1.3.5 - Sistema debug de niveles

Cada nivel tiene la posibilidad de usar un modo debug que imprime información útil y elimina las colisiones para facilitar la depuración de errores. Además, facilita la incorporación de nuevos objetos al mapa. Si el jugador se coloca en una posición determinada, se pueden añadir objetos al mapa con las teclas numéricas. El objeto se añadirá en la posición actual del jugador.

El modo debug o de creación facilita enormemente la tarea de crear niveles, ya que no hace falta reiniciar el juego para hacer pruebas, si no que los cambios ocasionados en el modo debug se guardan en el archivo de definición del nivel, que será cargado en el próximo inicio del juego con todos los nuevos objetos incorporados.

Puede encontrarse en el archivo src/util/levelDesigner.py del proyecto.

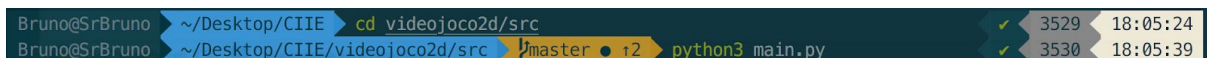
2.1.4 - Manual de usuario

Ejecutar el archivo dependencies.sh para instalar las dependencias necesarias para que el juego funcione correctamente:

```
./dependencies.sh
```

Ejecutar el archivo main.py con python 3, el cual iniciará el juego:

```
python3 main.py
```



```
Bruno@SrBruno ~/Desktop/CIIE$ cd videojoco2d/src
Bruno@SrBruno ~/Desktop/CIIE/videojoco2d/src$ python3 main.py
```

The screenshot shows a terminal window with two lines of text. The first line shows the user running 'cd videojoco2d/src' and the second line shows 'python3 main.py'. To the right of the second line, there is a green checkmark, the number '3530', and the time '18:05:39'. Above the first line, there is a green checkmark, the number '3529', and the time '18:05:24'. The terminal background is dark blue with light blue text.

Detalle de línea de comandos



Detalle de pantalla de inicio

2.2 - Videojuego en 3D

2.2.1 - Descripción

Juego de combate en tiempo real 3d con cámara en tercera persona, ambientado en la edad media. El jugador controla a el personaje protagonista e irá derrotando a enemigos dentro de un coliseo. Las distintas batallas se desarrollan en el mismo lugar pero con diferentes ambientaciones para aportar variedad a la experiencia.

2.2.1.1 - Personajes

2.1.1.1.1 - Protagonista



- Puntos de salud: 10
- Velocidad (andando): 2
- Velocidad (corriendo): 4

2.2.1.2 - Enemigos

2.2.1.2.1 - Esqueletos



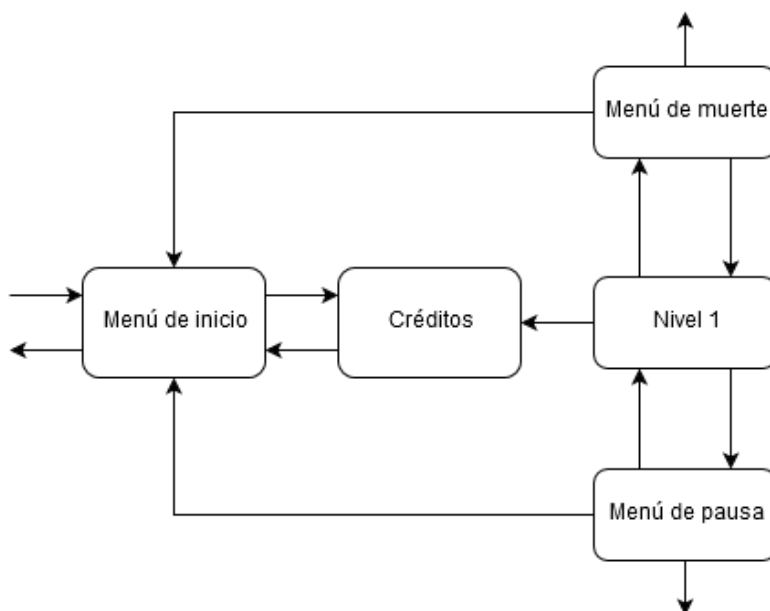
- Puntos de salud: 10
- Velocidad (andando): 1

2.2.1.3 - Diseño

Al iniciar el juego, lo primero que se muestra al jugador es un menú de inicio en el que puede elegir entre varias opciones: jugar a alguno de los niveles disponibles, ver los créditos o salir del juego. Si selecciona la opción de créditos, se mostrará un nuevo menú con los nombres de los autores, sin más opción que volver al menú de inicio. Si se selecciona jugar, se podrá elegir una de las fases disponibles. Una seleccionada, nos encontraremos ya en una escena de acción, en la que el jugador puede moverse en todas direcciones dentro del plano (no existen niveles de altura, siempre es el mismo). Esto se controla mediante las teclas WASD (W hacia adelante, S hacia atrás, A giro a la izquierda y D giro a la derecha) del teclado. En caso de que este movimiento quiera hacerse corriendo, se deberá mantener pulsada la tecla SHIFT (Mayus). Además, con el movimiento del ratón, se puede variar la dirección de la cámara y en consecuencia del movimiento. También se puede atacar con el personaje, esto se efectuará mediante la pulsación del botón izquierdo del ratón. También se podrá hacer “zoom” dentro de la escena, usando para ello la rueda del ratón. El personaje dispone de vida la cual se mostrará en la parte inferior izquierda de la pantalla. Los enemigos también disponen de su vida propia. El objetivo de las fases es hacer que la vida del/de los enemigo/s llegue a 0, para que se supere el nivel. En el caso de que nuestra vida llegue a 0, perderemos la partida. Al pasar esto se nos abrirá un menú desde el cual podremos salir del juego o reintentar la fase en la que nos encontramos, o volver al menú principal. El juego dispone de un menú de pausa, accesible con la tecla Escape (ESC) durante el juego. Desde esta pantalla, podremos tanto reanudar la partida como salir del juego o volver al menú principal.

2.2.2 - Escenas

Dentro de las escenas hay dos tipos de las mismas: escenas de menú y escenas de acción. En las primeras es jugador interactúa con una serie de opciones, y elige una de ellas de acuerdo con lo que desee. En las de acción se desarrolla el juego en sí: combate, movimiento, etc...



2.2.2.1 - Escena 1: Menú de Inicio

2.2.2.1.1 - Descripción

En esta escena se muestra el menú principal desde el cual se puede acceder al menú de selección de nivel ("Play"), a los créditos ("Credits") y por último, salir del juego ("Quit").

En el caso de seleccionar "Play", nos aparecerá el menú de selección de nivel, desde el cual podremos seleccionar el nivel que queremos jugar ("Level x"), o volver atrás al menú principal ("Back").

2.2.2.1.2 - Modelo

Todos los menús que aparecen en el videojuego siguen un patrón plantilla, ya que todos disponen de los mismos elementos:

- Fondo
- Cámara
- Sistema de Partículas para ambientación
- Navegación Menú

El fondo ("Background") se trata de un Panel ajustado a lo que verá la cámara en resolución 16:9, al cual se le puede añadir una imagen para que la añada como fondo de la escena.

La cámara es una cámara estática que apunta hacia el Fondo colocada para ver el mismo.

El sistema de partículas es un sistema direccional, es decir, dirigido en un sentido concreto para que aparezcan pequeñas motas blancas desde la izquierda de la pantalla hacia la derecha, añadido a la escena con el fin de darle un poco de movimiento y que no fuera totalmente estática.

En la Navegación del Menú se trata de un paneles, en los cuales se insertan los botones y todo el contenido que da información en la escena, es decir tanto los textos, como los botones con los que podrá interactuar el usuario.

Lo único que variará en los distintos menús son las características de estos elementos, pero siempre manteniendo la estructura o plantilla de Menú.

2.2.2.2 - Escena 2: Coliseo 1

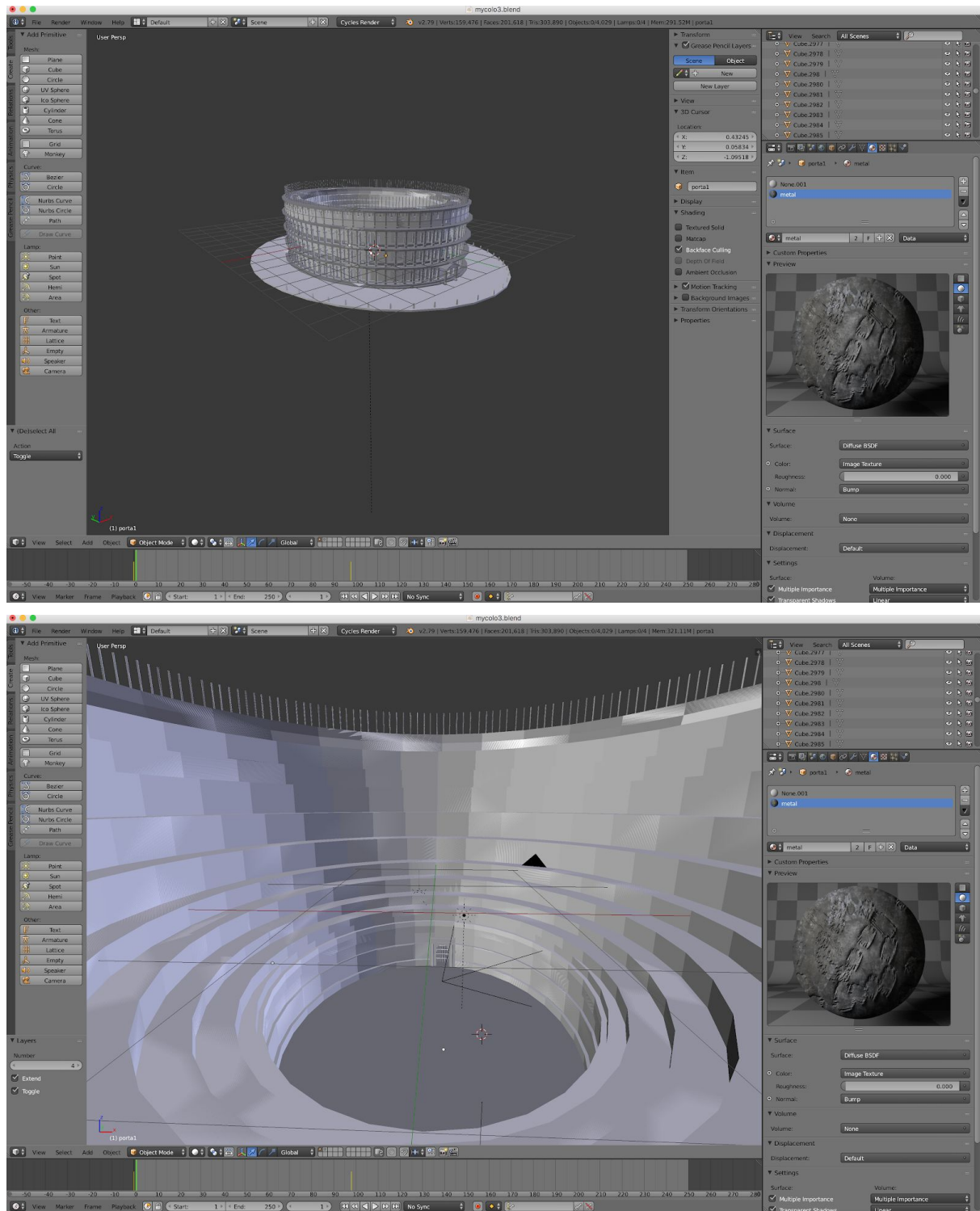
2.2.2.2.1 - Descripción

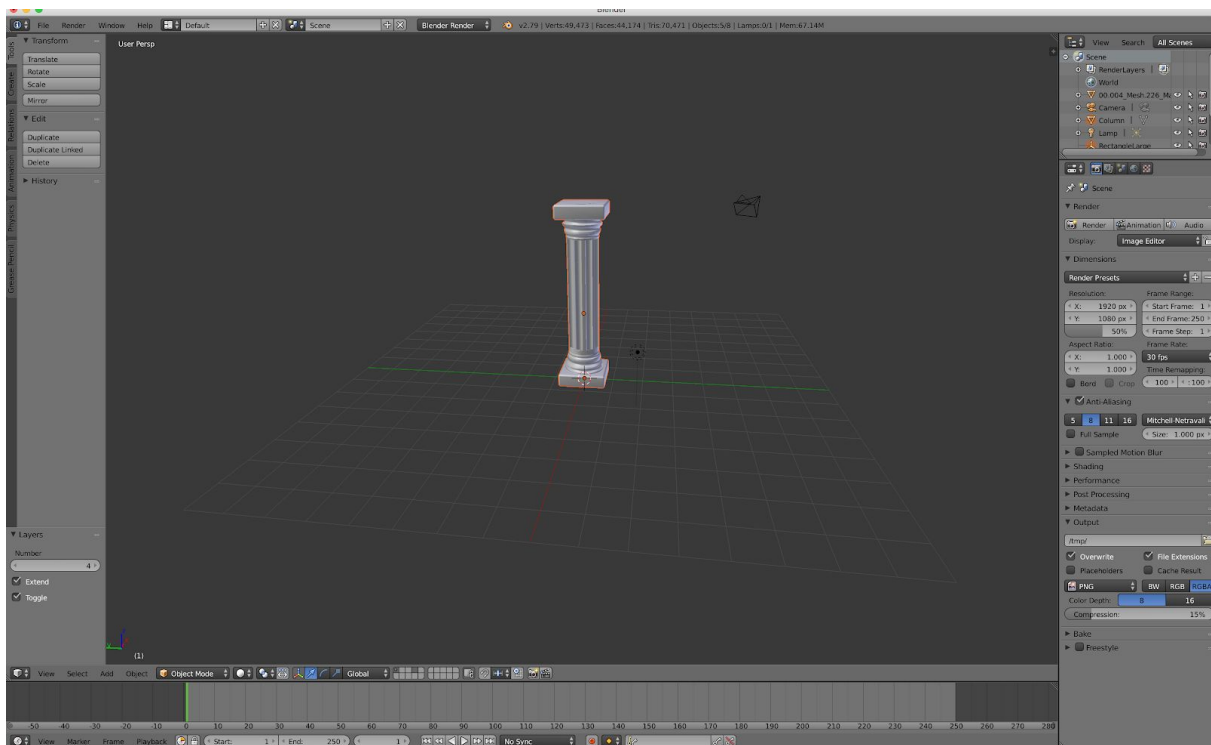
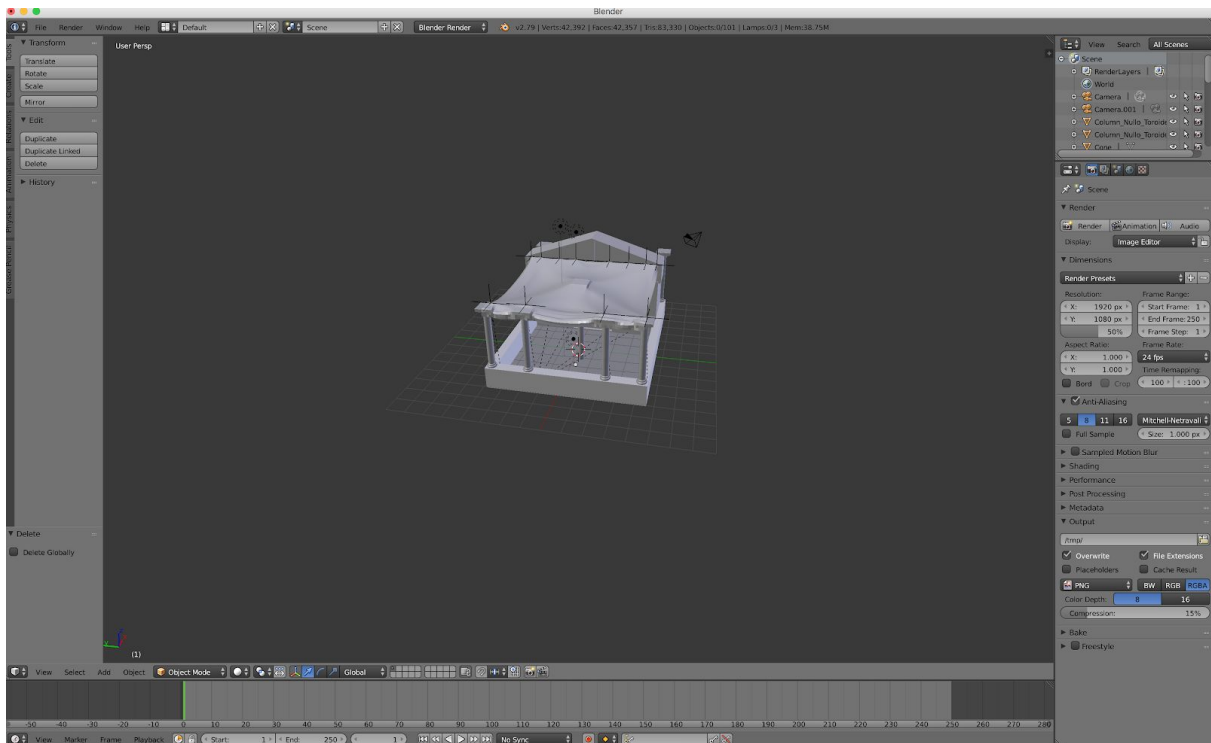
En esta escena de acción el jugador se enfrentará en el coliseo a varios enemigos, esqueletos, hasta derrotarlos a todos.

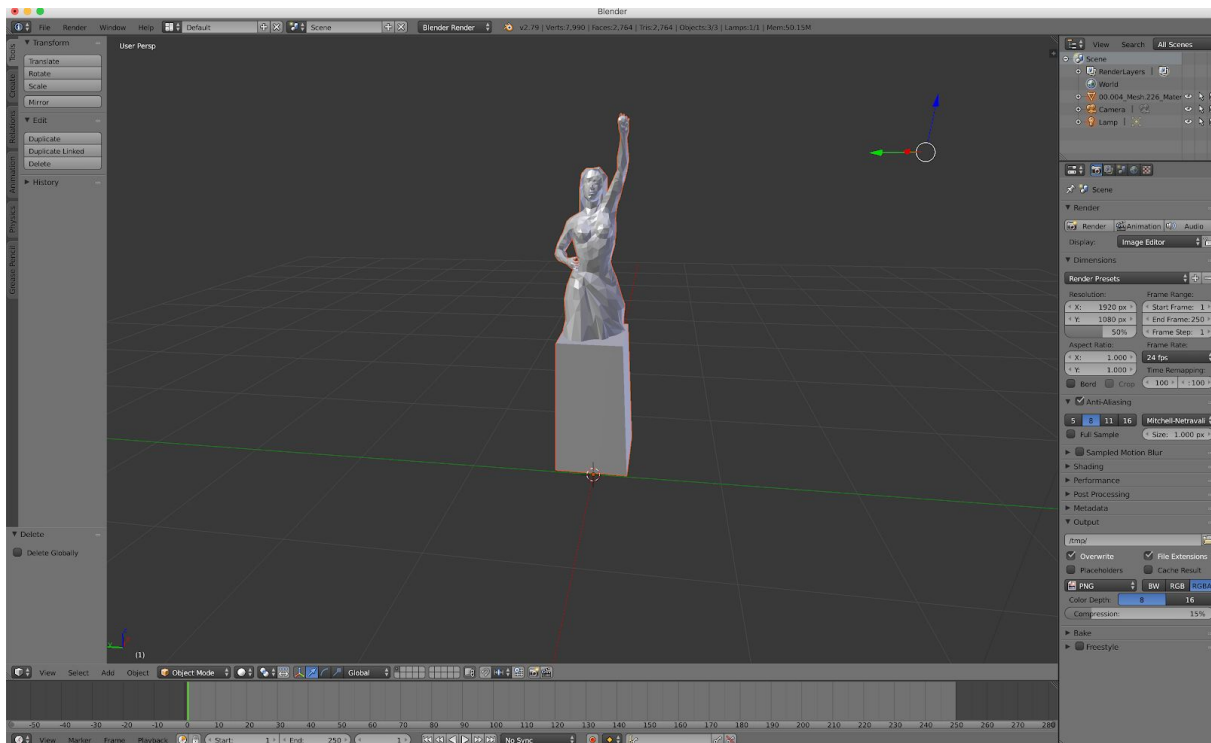
El modelo del coliseo está hecho a medida para el videojuego, utilizando las herramientas que nos proporciona Blender, además, está compuesto por distintos objetos también hechos a mano, como por ejemplo el palco del coliseo donde se hallarían los jefes.

Consta de columnas de la época y de "montones de rocas" para decorar y servidor de obstáculos a la hora de escapar de los esqueletos.

2.2.2.2.2 - Modelo







El modelo se ha realizado de una manera poco conservadora, es decir, no se han optimizado el número de polígonos por falta de tiempo así como tampoco se ha llevado a un nivel de detalle “suficiente” para una versión final del juego, lo que quiere decir que en un futuro debería mejorarse en términos de eficiencia a la hora de jugar y de detallismo.

Todas las texturas utilizadas han sido generadas para el propio coliseo y sus objetos, primeramente se han modelado en blender para conseguir una vista previa y seguidamente se han replicado lo más parecido posible en unity (Por desgracia no se pueden importar las texturas de blender directamente a unity). Se han aplicado mapas de normales a las texturas para conseguir efectos de profundidad y un realismo mínimo.

Como trabajo futuro también se deberían de añadir “personas” al coliseo para que pareciese más real todo, así como un sonido ambiente.

2.2.2.2.1 - Inteligencia Artificial

| Heap |
|---|
| <ul style="list-style-type: none"> - items: T[1..n] - currentItemCount |
| <ul style="list-style-type: none"> + Heap(int): Heap + Add(T): void + RemoveFirst(): T + UpdateItem(T): void + Count(): int + Contains(T): bool + SortDown(T): void + SortUp(T): void + Swap(T, T): void |

| Grid |
|---|
| <ul style="list-style-type: none"> + unwalkableMask: LayerMask + gridWorldSize: Vector2 + nodeRadius: float - grid: Node + path: List<Node> - nodeDiameter: float - gridSizeX: int - gridSizeY: int |
| <ul style="list-style-type: none"> + MaxSize(): int + Awake(): void + CreateGrid(): void + GetNeighbours(Node): List + NodeFromWorldPosition(Vector3): Node + OnDrawGizmos(): void |

| IA |
|--|
| <ul style="list-style-type: none"> + playerDmg: int # target: Transform # character: Character # state: CharacterState |
| <ul style="list-style-type: none"> - Start(): void - Update(): void |

| Node |
|--|
| <ul style="list-style-type: none"> + walkable: bool + worldPosition: Vector3 + gridX: int + gridY: int + gCost: int + hCost: int - heapIndex: int + parent: Node |
| <ul style="list-style-type: none"> + Node(bool, Vector3, int, int): Node + HeapIndex(): int + fCost(): int + CompareTo(Node): int |

| PathFinding |
|--|
| <ul style="list-style-type: none"> - requestManager: PathRequestManager - grid: Grid - mPath: List<Node> |
| <ul style="list-style-type: none"> - Awake(): void + StartFindPath(Vector3, Vector3): void - FindPath(Vector3, Vector3): IEnumerator - RetracePath(Node, Node): Vector3[1..n] - SimplifyPath(List<Node>): Vector3[1..n] - GetDistance(Node, Node): int |

| PathRequestManager |
|---|
| <ul style="list-style-type: none"> - pathRequestQueue: Queue<PathRequest> - currentPathRequest: PathRequest - instance: PathRequestManager - pathFinding: PathFinding - isProcessingPath: bool |
| <ul style="list-style-type: none"> - Awake(): void + RequestPath(Vector3, Vector3, Action): void - TryProcessNext(): void + FinishedProcessingPath(Vector3[1..n], bool): void |

| Unit |
|--|
| <ul style="list-style-type: none"> + target: Transform - path: Vector3[1..n] - character: Character - state: CharacterState - targetIndex: int - lastPos: Vector3 - counter: int + grid: Grid - controller: CharacterController |
| <ul style="list-style-type: none"> - Start(): void - Update(): void + OnPathFound(Vector3[1..n], bool): void - FollowPath(): IEnumerator + OnDrawGizmos(): void |

Para la inteligencia artificial se ha implementado un algoritmo de Pathfinding en A* (A estrella). Se ha dividido en 5 clases: Grid, Unit, Pathfinding, PathRequest y Heap.

Para poder implementar correctamente el algoritmo se ha creado una clase "Grid" que como indica su nombre representa un grid (una rejilla) en el mapa (invisible), este grid tiene una función importante y es que es el encargado de agrupar los distintos puntos del mundo en "cuadrados" de la rejilla para que se puedan delimitar los caminos y encontrar un path óptimo. De esta manera, un punto cualquier del escenario tendrá asociado una región o cuadrado de la rejilla, no obstante, se puede comprobar a qué región pertenece un punto con la función "NodeFromWorldPosition" implementada en dicha clase.

Cada región de la rejilla está representada por un objeto de tipo Node, que viene siendo un nodo donde se almacenan la posición real (en el mundo/escenario) de cada región, además de su posición en el grid (X,Y), además de indicarnos si esa región es *walkable*, es decir, si podemos pasar por ese nodo a la hora de calcular el *path*. A su vez, cada nodo también tiene asociado el valor de su función de coste, que nos permitirá más tarde calcular el *path* óptimo hasta el jugador, este valor de coste estará determinado por el *gCost* y el *hCost* que básicamente son el coste de moverse desde el nodo de salida y el coste de moverse desde el nodo final.

La clase Unit contiene toda la lógica de movimiento del enemigo, en su update se ejecutarán las peticiones para buscar un path hacia el jugador cada 100 ticks, y una vez encontrado un *path* se ejecutará la función para seguirlo.

Las clases PathRequest y PathFinding como su nombre indica se encargan de las peticiones de búsqueda de *path* y de la tarea de buscar un *path* óptimo, y además se ha implementado con montículos para mejorar su eficiencia.

En PathFinding además de la implementación genérica de A* también se ha llevado a cabo una simplificación del *path* de manera que a la hora de ver al enemigo avanzar se aprecia de una manera más suave (*smooth*). Básicamente se han eliminado puntos siguiendo un algoritmo básico que sigue la premisa de que "si el punto i es visible hasta el punto i + 2, entonces el punto i + 1 se elimina".

Pseudocódigo A* seguido.

```
OPEN //the set of nodes to be evaluated
CLOSED //the set of nodes already evaluated
add the start node to OPEN

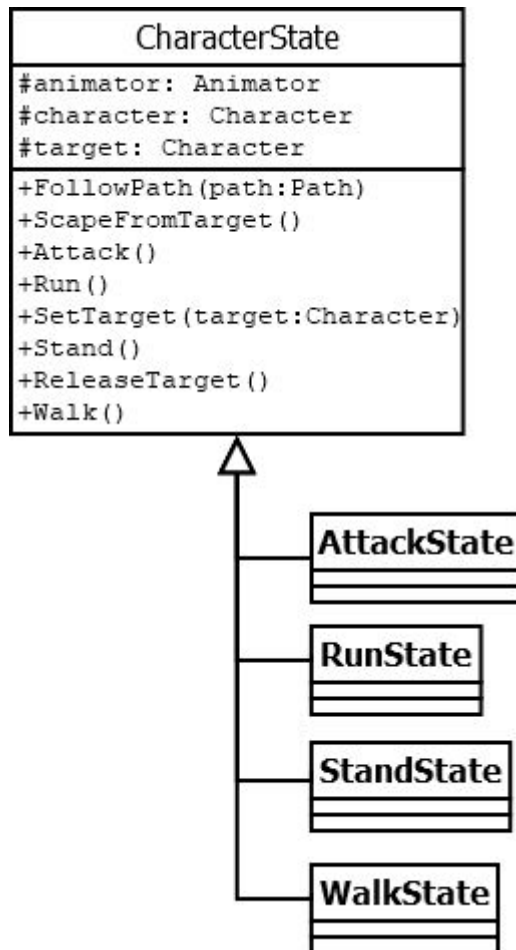
loop
  current = node in OPEN with the lowest f_cost
  remove current from OPEN
  add current to CLOSED

  if current is the target node //path has been found
    return

  foreach neighbour of the current node
    if neighbour is not traversable or neighbour is in CLOSED
      skip to the next neighbour

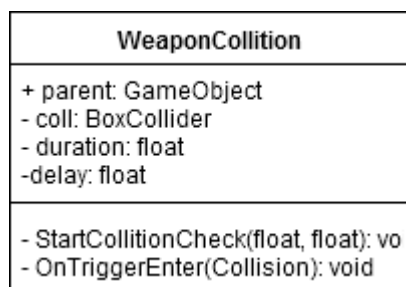
    if new path to neighbour is shorter OR neighbour is not in OPEN
      set f_cost of neighbour
      set parent of neighbour to current
      if neighbour is not in OPEN
        add neighbour to OPEN
```

2.2.2.2.2 - Máquina de estados



Estas clases son las encargadas de gestionar el patrón estado. Donde **CharacterState** es una clase abstracta, donde se definen los comportamientos generales de las acciones. Y los distintos estados hijos reimplementan los comportamientos oportunos.

2.2.2.2.3 - Sistema de colisiones



Las colisiones de ataque se modelaron usando un box collider hijo del **GameObject** que queremos que ataque. Este collider está desactivado en todo momento, excepto cuando se cambia al estado de ataque donde se activa llamando al método `StartCollisionCheck`.

Las colisiones de los personajes con el terreno se modelan usando el componente "Character Controller" y dotando de un collider a los objetos con colisión.

2.2.2.2.4 - Sistema de sonido



El sistema de sonido basa su estrategia en confiar a una clase la gestión del mismo. Esta clase, SoundManager, se encarga de la manipulación de todo tipo de audio, desde la música del juego hasta los efectos de la interfaz de usuario. Para poder realizar esta gestión de manera segura, y entendiéndose segura como una forma en la que este control sea total en cada parte del juego, sigue una estrategia Singleton que garantiza que una única instancia de esta clase es generada en tiempo de ejecución. Así, si en una escena se ordena que suene una música, se podrá, sin mayor complicación, ordenar su parada desde otra. Esto permite una gran flexibilidad ya que no hay que realizar comprobaciones del estado del gestor de sonido. Por otra parte, en SoundCte se almacenan todas las rutas de

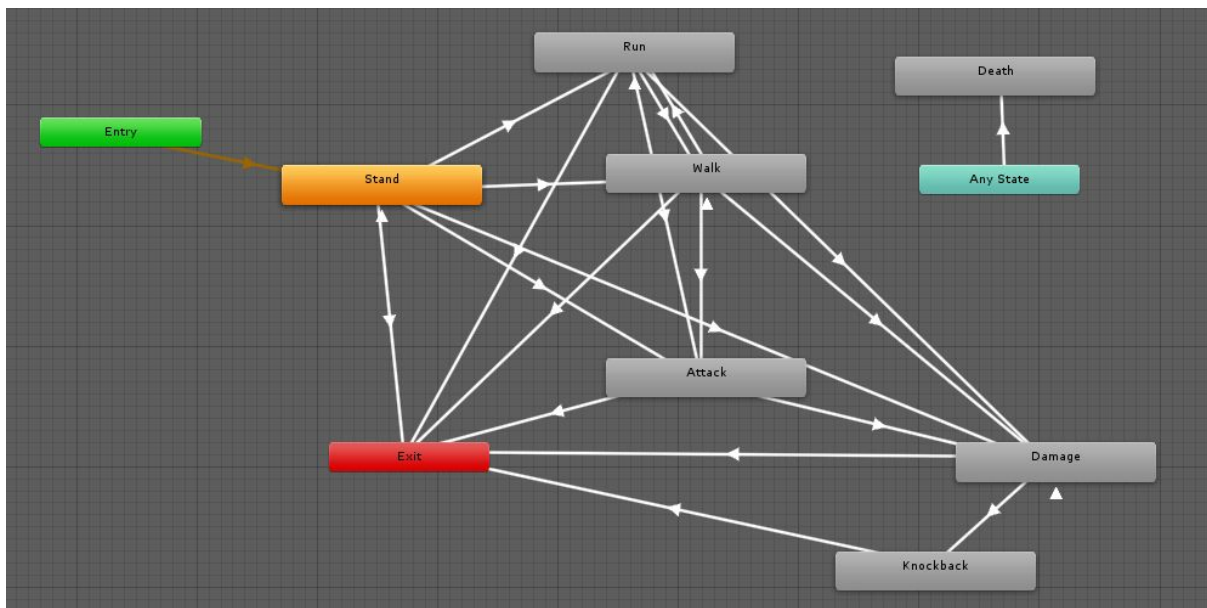
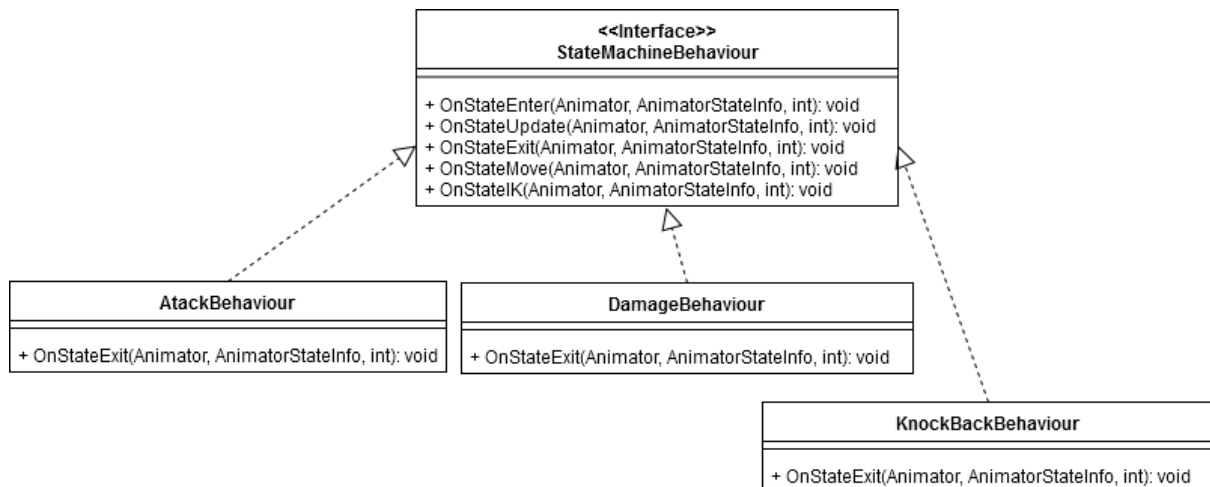
cada fichero de sonido, así como su método de cargado al juego en forma de AudioClip. El hecho de guardar las rutas como una constante de tipo String permite manipular los sonidos fuente, los archivos mp3, wav... de tal forma que solo es necesario cambiar este archivo de constantes tras las modificaciones oportunas. En SoundUtil encontramos una serie de métodos que simplifican aún más el uso de los sonidos, para, por ejemplo, aquellos que se quieren reproducir aleatoriamente dentro de una selección concreta, de cara a aportar variedad en el juego. En cuanto a SkeletonSoundScript, es un fichero que se encarga de dotar de efectos de sonido a los esqueletos enemigos, usando una de las funciones de SoundUtil. Pero, además, para que estos sonidos se mantengan en el tiempo, se ha implementado un contador en “waiter” para que reproduzca sus sonidos periódicamente. Los sonidos del protagonista y de colisiones son gestionados en las clases que los tratan por un motivo de eficiencia (se evita realizar el doble de comprobaciones). Por último, en cuanto a los ficheros que comienzan por “Audio...” son simples scripts que ocultan el uso del SoundManager (simplemente en cada función de las que poseen hacen una llamada concreta a SoundManager) para no llamarlo directamente desde Unity. El motivo es que así no se tendrán que manipular los componentes de la escena para cambiar de gestor de sonido, si fuese necesario.

2.2.2.2.5 - Sistema de cámara

| OTSCamera |
|--|
| + shoulder: GameObject + cam: Camera + target: Transform + distance: float + ySpeed: float + xSpeed: float + yMinLimit: float + xMinLimit: float + distanceMin: float + distanceMax: float + arriba: float - rigidBody2 : Rigidbody + offset: float + number: int + x: float + y: float |
| - Start(): void + LateUpdate(): void + ClampAngle(float, float, float): float |

La cámara utiliza un sistema “Over The Shoulder”, que orbita al personaje moviéndose en los ejes horizontal y vertical con el ratón, y acercándose o alejándose del personaje con la rueda del mismo. Todos los aspectos del movimiento de la cámara son configurables, tanto el objetivo a seguir, modificable mediante un offset vertical y horizontal, como la sensibilidad en los tres ejes de movimiento. La cámara dispara en cada actualización un RayCast, que golpea los objetos entre ella y el jugador, y se acerca para mantener el personaje en plano automáticamente si es necesario.

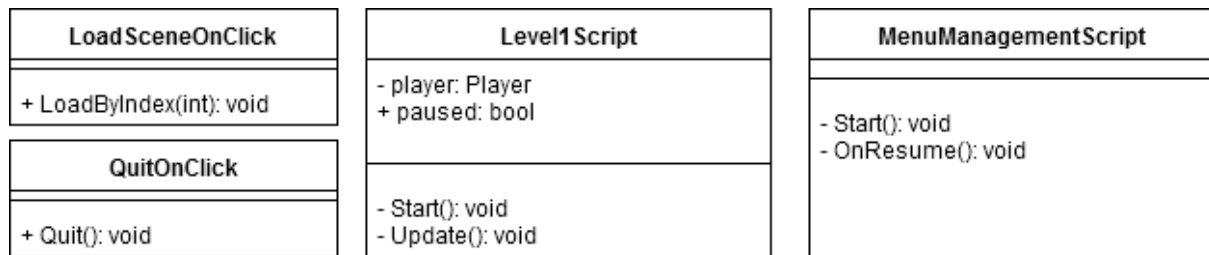
2.2.2.2.2.6 - Sistema de animación



Para las animaciones se usa el sistema de unity de AnimationControllers, se dejan las transiciones de animaciones en manos de unity y solo se implementan resets de parámetros en las animaciones de Ataque, Daño y Knockback en sus clases AttackBehaviour, DamageBehaviour, KnockBackBehaviour, respectivamente. También es realizado en estas clases el cambio al estado por defecto Stand, cuando la animación finaliza.

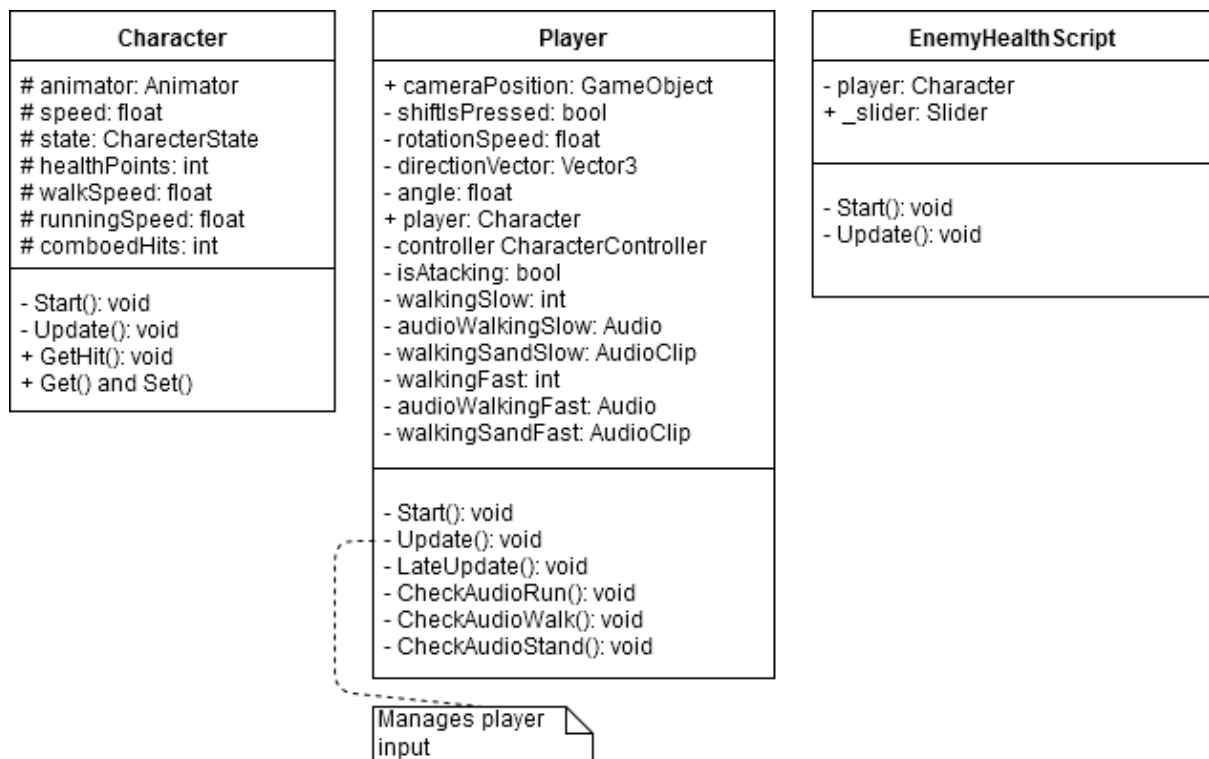
Los parámetros que gestionan las animaciones, son activados y desactivados desde el estados del personaje.

2.2.2.2.7 - Sistema de carga/descarga de escena



En el juego existen varias escenas, y por ello es necesario un sistema de carga de las mismas. Entonces, se han desarrollado los ficheros LoadSceneOnClick y QuitOnClick. El primero de ellos permite con su método cargar cómodamente una escena, por su índice dentro de la build del proyecto, mientras que el método Quit de la segunda clase permite salir del juego cuando se hace click en un determinado componente, como podría ser un botón de salida. Sin embargo, para un menú de pausa no basta con cargar la escena del mismo y volver a cargar la escena anterior, ya que se perdería el estado de la misma. Así, se genera un script para el nivel 1 que permite gestionar esto mismo. Se encarga de que al recibir el trigger del menú de pausa (tecla ESC en este caso), se mantenga el estado actual de la escena. Para ello carga el menú de forma aditiva y deshabilita el script del jugador para que no se reciban inputs para el personaje. Además, se marca el estado a paused. Cargado el menú de pausa, se activa el script MenuManagementScript que se encarga de que al salir del mismo, si la opción es reanudar la partida, se cargue el nivel apropiado en el estado en el que quedó.

2.2.2.2.8 - Jugador y enemigos



La clase Character es la encargada de gestionar la características de los personajes del videojuego. Así como notificar al estado del personaje que ha sido golpeado mediante el método GetHit.

La clase player se encarga del movimiento del jugador. En esta clase es necesaria la posición de la cámara porque todo movimiento es en función de esta. También se encarga de dar la orden de reproducir los sonidos ligados al player.

2.2.2.3 - Escena 3: Menú de Pausa

2.2.2.3.1 - Descripción

En esta escena se muestran 3 botones que permiten que el usuario pueda realizar varias acciones durante el transcurso de un nivel.

La primera opción es volver al nivel que se estaba jugando ("Resume"), la segunda opción sería volver al Menú de Inicio ("Main Menu") y por último, también se podría salir directamente del juego ("Quit").

2.2.2.4 - Escena 4: Menú de Muerte

2.2.2.4.1 - Descripción

En esta escena se muestran 3 botones nuevamente con los que el usuario podrá realizar las distintas acciones.

La primera será la opción de volver a intentar el nivel ("Repeat"), la segunda y tercera opción son, al igual que en la Escena 3, volver al Menú de Inicio ("Main Menu") y salir del juego ("Quit") respectivamente.

2.1.2.5 - Escena 5: Créditos

2.2.2.5.1 - Descripción

En esta escena se muestran, sencillamente, los autores del videojuego y sus contribuciones, junto con un botón ("Back") que permite volver a la escena del Menú de Inicio.

2.2.3 - Aspectos destacables

2.2.3.1 - Sistema de Menús

El juego dispone de una serie de menús que ayudan al usuario a realizar las opciones que desee proporcionando el feedback adecuado. De esta manera, lo primero que se encuentra es un menú de inicio que le permite elegir la acción deseada, entre las cuales se encuentran "Jugar", "Créditos" o "Salir". Además, existe un menú de pausa pulsando la tecla ESC el cual sirve para interrumpir la partida en caso de necesidad, desde el cual se podrá reanudar la acción o salir del juego. Por último, el menú de muerte permite repetir el nivel seleccionado o bien abandonar el juego.

2.2.3.2 - Cámara

La cámara utiliza un sistema “Over the Shoulder”, donde la cámara orbita al personaje, pudiéndose modificar tanto el offset horizontal como el vertical, y se mueve en ambos ejes con el ratón, y haciendo “zoom” con la rueda del ratón. Cualquiera de estos aspectos es configurable, tanto las velocidades horizontal y vertical como la de “zoom”, así como el nivel mínimo y máximo de zoom. También existe la opción de fijar a un enemigo, lo cual enfoca la cámara hacia él, manteniendo al personaje en plano al mismo tiempo.

2.2.3.3 - Bugs

La música desaparece al cambiar de ventana, desde la del juego a cualquier otra.

El cálculo de las rutas en la IA se realizan todas de forma síncrona, esto llega a. Podría arreglarse con una pequeña randomización en sus path.

2.2.3.4 - Gestor de sonido, efectos y música

El juego incorpora un gestor de sonido de una third-party, “EazyTools”, “EazySoundManager”. El hecho de seleccionar un componente externo viene a raíz del espíritu de la modularidad y reutilización de código. No obstante, se ha comprobado que el código de este gestor utiliza técnicas apropiadas para un óptimo desarrollo, como puede ser un patrón Singleton para evitar múltiples instancias de este gestor (es deseable que el control de sonido se haga desde un mismo objeto de cara a la manipulación de los disparadores de audio. Por contra, con varias instancias, una podría empezar un audio y desde otra intentar pararlo sin éxito). Además, se ha complementado este gestor con los ficheros de script “SoundUtil.cs” y “SoundCte.cs” dentro de “Assets/Scripts/Sound”, que adaptan las funcionalidades de este gestor a las del proyecto. En este último se hace referencia a los ficheros de sonido que incorpora el proyecto, de calidad y tratados específicamente para el mismo. Mención aparte para la música, la cual es propiedad del juego “Medieval II Total War”.

2.2.3.5 - Modelado de objetos y animación

El juego dispone de modelos de realizados a propósito para el mismo, con un gran nivel de detalle tanto en el aspecto 3d como en el de texturas, donde además se destaca el uso de bump mapping para lograr sensación de profundidad en la textura, pese a estar esta aplicada en una superficie plana. Asimismo, el 3d del protagonista, aunque no ha sido modelado, si ha sido animado mediante el uso de técnicas de captación de movimiento corporal (utilizando el famoso sensor Kinect y Blender) por parte del equipo de desarrollo, logrando unos resultados más que aceptables de cara al aspecto visual de la animación.

Se ha logrado, con éxito, la animación de los estados “Stand”, “Walk”, “Run” y “Hit” del protagonista.

2.2.4 - Manual de usuario

Click izquierdo de ratón: En menús: selección. En Juego: Ataque del usuario.

W, A, S, D: Movimiento del jugador

Shift: Correr

Esc: Menú de pausa

Para la ejecución del juego basta con hacer doble click en el archivo .exe

En una ventana inicial por defecto de unity, se podrá elegir la resolución del juego y la calidad gráfica del mismo. Para jugar, pulsar play con la configuración elegida.