Proyecto de Sonido en Videojuegos

David González y Patricia Cabrero

Introducción:

El juego que hemos elegido para sonorizar es el proyecto en el que trabajamos el año pasado para la asignatura de Proyecto 2. Entonces éramos un equipo de seis personas; sin embargo, para hacer esta extensión del motor de sonido sólo hemos participado dos: David González y Patricia Cabrero.

Proceso de diseño del juego:

Aspectos generales

El equilibrio universal del bien y el mal ha sido alterado debido al buen hacer de la gente del planeta. Como en el punto medio está la virtud y un exceso de bondad sería tan malo como uno de maldad, los Guardianes del Equilibrio crean a una niña con poderes cuyo cometido será acabar con las cosas buenas que sobran en el planeta. De esta manera se restablecerá el orden cósmico y el universo volverá a la normalidad.

El juego trata de derrotar a los enemigos que irán apareciendo en un ring por oleadas. La cámara seguirá una perspectiva en tercera persona situada a la espalda del personaje que irá eliminando a los enemigos mediante sus ataques **cuerpo a cuerpo** y habilidades o **magias y transformaciones.**

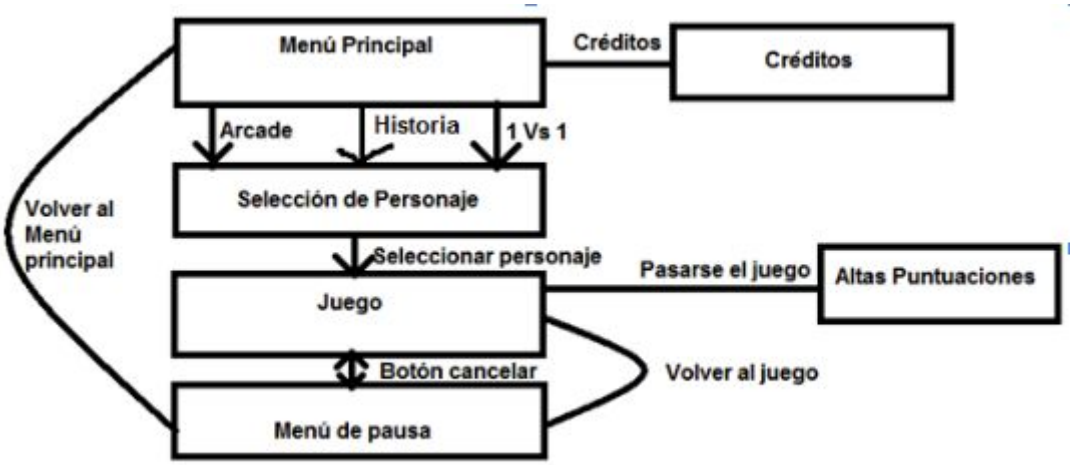
Menús y configuración

Menú principal:

- Historia → Empieza el modo historia.
- Créditos → Lleva a la pantalla de créditos.
- Salir del juego → Cierra la aplicación.

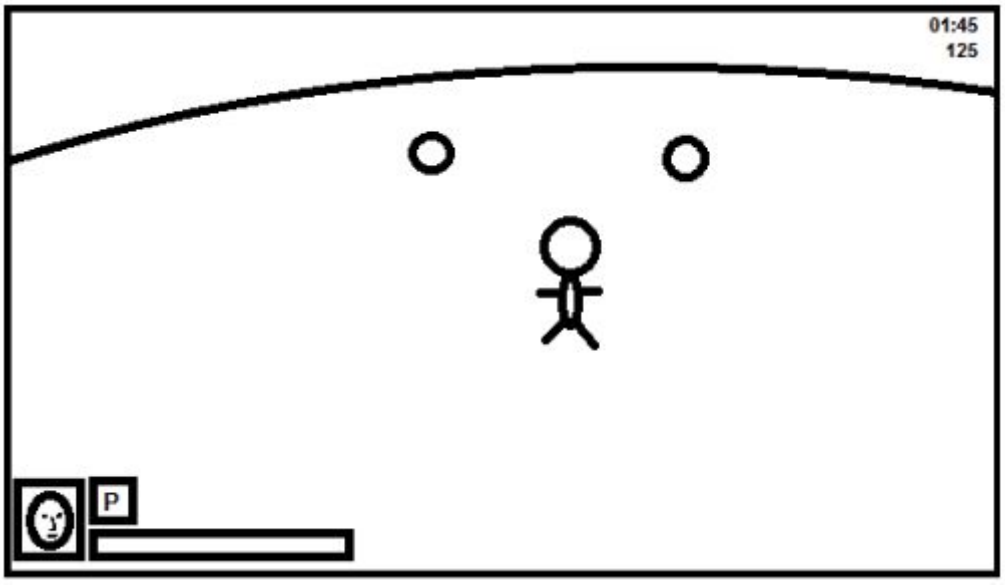
Durante el juego

- Menú de pausa → Se accede con el botón pausa, Tiene la opción de volver al Menú principal o de mirar los ataques que puede realizar el personaje.



Interfaz

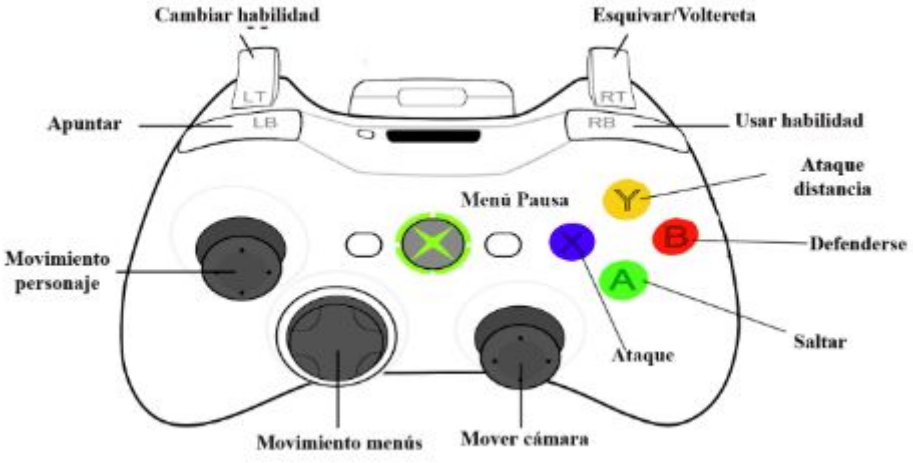
En la parte inferior izquierda de la pantalla aparece la vida del personaje y si tiene algún potenciador activado la imagen correspondiente a el.



Jugabilidad

El jugador debe utilizar los diferentes movimientos a su disposición moviéndose a través de un espacio tridimensional, utilizando los sticks del mando para controlar el movimiento del personaje y la cámara. Se utilizan los diferentes botones de acción para golpear y utilizar los diferentes ataques y habilidades para acabar con los enemigos. Cada golpe que de el personaje da puntos. Cada enemigo eliminado sin recibir daño aporta un multiplicador a la puntuación.

La cámara sigue al jugador y a una distancia de 3, 4 cuerpos desde una posición un poco elevada y el jugador puede reorientarla a su gusto. La cámara orbita alrededor del personaje usando el stick derecho.



Durante el juego el jugador puede recibir habilidades o bonificaciones por **superar niveles**. Las habilidades necesitan energía para ser utilizadas. Esta energía tiene que recargarse con el tiempo o a través de objetos situados en lugares aleatorios del mapa.

El juego se termina al acabar con todos los enemigos, o cuando la barra de vida llegue a cero. Para el modo arcade, al terminar la partida aparece la pantalla de puntuaciones, **dándote la opción de poner el nombre** y de ahí el juego pasa de nuevo al menú principal.

Personajes

Enemigo genérico

Habilidades:
Golpe (Se impulsa hacia adelante y golpea, luego recupera su posición anterior).

- **Dividirse** Aleatoriamente se divide en dos enemigos de menor tamaño y poder.
- **Lanzar fuego** Escupe fuego por la boca

Alaia

Habilidades:

- Tiene acceso a diferentes ataques que varían según el color que tenga en ese momento.
    - Azul onda vital
    - Rojo
    - Amarillo
    - Verde

Referencias

- *Ratchet and Clank Future: Tools of Destruction - Naughty Dog (2007)*
- *One Piece: Pirate Warriors 3 - Koei Tecmo, Omega Force (2015)*
- *Star Vs the Forces of evil - Disney, Daron Nefcy (2015)*
- *Super Mario Kart - Nintendo (1992)*

Herramientas utilizadas en el proyecto:

Puesto que se pedía la implementación del juego a bajo nivel, el motor lógico y de comunicación entre las diferentes entidades del mismo fue implementada por nosotros.
Por otra parte, incorporamos:

- Ogre3D como motor gráfico para las mallas, las texturas, las partículas y las animaciones.
- Bullet3, que es la librería que tomamos como base para el motor físico;
- CEGUI, para implementar el GUI del juego con el que interactuamos en los menús y que muestra la información relativa al propio juego en la partida.
- OIS, para gestionar la entrada de información por parte del usuario, ya sea mediante mando o mediante teclado.
- FMOD y FMOD Studio, para implementar el motor de sonido, que será en lo que nos centraremos.

Estructura del proyecto:

El juego tiene su punto de entrada en el método main del archivo ProyectoBasico.cpp. El main crea una instancia de la clase Juego y ésta llama a su método run, que se ejecutará hasta que el programa termine por una acción del usuario.

La clase Juego contiene una pila de estados entre los que están el menú, la pausa o el mismo estado del juego que se irán cambiando en ejecución. Además el juego es el encargado de inicializar las diferentes librerías que hemos incluido para el funcionamiento del proyecto y recoge las pulsaciones del usuario y se las manda al estado correspondiente.

Cada estado lleva un mapa que relaciona las entidades que contiene con sus respectivos nombres. También coloca la cámara del motor gráfico a conveniencia, el viewport y la luz en lo referente al apartado gráfico.
Tiene un objeto de la clase Fisic que crea la simulación del mundo físico con las propiedades alterables como la gravedad, el rozamiento, etc.
Contiene el puntero al System de FMOD que lleva las principales propiedades sonoras de la escena y permite la creación de sonidos y canales, aunque eso se hace en un objeto del que hablaremos después.
Además almacena la ventana del gestor del GUI para acceder a los diferentes paneles y botones del mismo.

Cada estado lleva también una cola de prioridad de mensajes, que son el mecanismo de comunicación entre entidades y componentes del juego.
Los mensajes funcionan de la siguiente forma: el estado hace un update, dentro del cual se hace el update de todas las entidades que haya activas en ese momento en él. La entidad es simplemente un contenedor de componentes, que son los que llevan el comportamiento de cada objeto del juego. Por lo tanto, en el update de cada entidad se llamará al update de cada componente, que realizará su función y si es necesaria su comunicación con otro componente (en la gran mayoría de los casos será así) crea un mensaje con una información determinada, un tipo y un subtipo y se encola en el estado especificando o no un destinatario. A continuación se ordenan esos mensajes por prioridad según su tipo y en el siguiente update se envían a su entidad y componente destinatario, que lo interpretará y realizará su función correspondiente en respuesta al mismo.

Ahora hablaremos de manera superficial de cada uno de los componentes, dejando para el final los referentes al sistema de sonido y profundizando en ellos.
Los componentes más destacables son:

- Camara: regula el funcionamiento de la cámara, su posición en función de la del personaje principal y su rotación, que varía según el movimiento del joystick derecho del mando.

- Consumible: este componente se aplica a las entidades que van a ser consumidas por la protagonista. Al colisionar contra ellas desaparecen y aportan un aumento de vida o maná en Alaia.

- FComponent: se trata del cuerpo físico del objeto que lo contiene. Establece un cuerpo rígido del tamaño proporcional al componente gráfico que puede consistir en un cubo, una esfera u otros cuerpos geométricos. Hace que las entidades no puedan atravesarse entre sí y nos da la información de si el cuerpo ha colisionado para poder enviar un mensaje en respuesta a ese encuentro. También lleva el control de los objetos triggers en escena y la respuesta a los mismos.

- GComponent: se trata del cuerpo gráfico del objeto que lo contiene. Contiene la malla de puntos y la textura del objeto, así como su posición en el mundo gráfico (que es diferente al mundo físico) y su orientación.

- Animation: este componente se comunica con el GComponent para reproducir y parar las animaciones que tiene un modelo.

- BalaComponent: es el comportamiento que lleva cada bala que lanza la protagonista como ataque. Le asigna una velocidad y una dirección además de poner en funcionamiento sus partículas.

- IABola: lleva la inteligencia artificial de los enemigos de Alaia (las bolas de fuego). Su desplazamiento hacia ella y su bipartición en otros dos enemigos de menor tamaño pasado cierto tiempo.

- LifeComponent: este comportamiento lo lleva Alaia y representa su salud, que decrece cuando se choca contra un enemigo y aumenta cuando se come el consumible correspondiente. Al llegar la salud a 0 el juego se termina.

- ManaComponent: paralelamente al componente anterior, éste lleva la cuenta del maná de la protagonista, que se gasta lanzando sus conjuros y se repone con el consumible correspondiente. Si el maná está muy bajo, no se pueden lanzar conjuros.

A continuación se exponen los componentes referentes al sonido:

SoundManager

Este componente lo lleva una única entidad en cada estado y es el que maneja el System de FMOD (de bajo nivel) y el de FMOD Studio, aunque este último no lo explotamos en el proyecto.

Contiene dos mapas: en el primero se almacenan los efectos de sonido y en el segundo se incluyen las diferentes músicas que suenan en el juego.

También lleva cuatro vectores; dos de ellos se encargan de las reverbs (uno de las de la música y otro de los de los efectos de sonido) y los otros dos tienen los canales en los que se van a reproducir los sonidos (también uno para los sonidos y el otro para la música).

En el método cargarAssetsAudio deja cargados (en modo FMOD_3D) e introducidos en el mapa todos los efectos de sonido y establece inicialmente su rolloff con una distancia minima de 40 unidades y la máxima de 5000, para después cargar todas las pistas de música ambiente, pero en este caso en modo FMOD_2D.

ReverbSwap() cambia entre distintos tipos de reverb.

ReproduceFx(...) manda reproducir un sonido en la posición que se le pasa por parámetro y con el parámetro wet especificado.

ReproduceAmbM(...) hace lo propio en este caso con la música ambiente. Y también se le puede especificar si queremos un fade y el valor de wet.

Cabe destacar que el número de canales con el que contamos es limitado, de manera que tenemos controlada la cantidad de memoria que usamos y evitamos que haya sonidos que no aparezcan por falta de espacio. Al reproducir cualquier nuevo sonido lo mandamos al primer canal que haya libre o en caso de que ninguno esté libre lo enviamos al que primero se ocupó.

Por último, el método paraAmb(...) detiene la música ambiente que esté sonando y también lo podemos hacer en un fade out.

El update de este componente espera mensajes de tipo Audio, que pueden tener tres subtipos:

- Musica: el mensaje lleva la información de la canción que se quiere reproducir o parar seguida de Play o Stop, y llama a reproduceAmbM() o paraAmb() respectivamente.

- Effect: el mensaje lleva Play, el nombre del efecto y la posición desde la que se quiere emitir. Con ello se llama a reproduceFx().

- ReverbChange: no lleva nada de información adicional. Si el sound manager recibe un mensaje de este tipo llama a reverbSwap().

Cabe destacar que los mensajes de reproducción de música se llamarán normalmente al inicio de cada estado (o escena de juego) y se mantendrán sonando en bucle mientras el estado permanezca activo.

Por otra parte los mensajes de reproducción de efectos se enviarán desde el personaje al realizar distintas acciones como correr, saltar o disparar, tomando siempre como posición la localización actual de la entidad que lo envía. También envían mensajes de efectos los enemigos cada cierto tiempo y al dividirse y atacar.Finalmente, los mensajes de tipo ReverbChange se enviarán en respuesta a una pulsación de botón del mando por parte del usuario.

SoundListener

Se trata del componente que se añadirá a la entidad que queramos que actúe como listener de todos los sonidos de la escena. En nuestro caso, el listener en la escena de menú se colocará en un lugar por defecto (puesto que el menú no cuenta con una sonorización en 3D) y es después en el propio estado del juego en el que al inicio se asigna el sound listener a la entidad del personaje protagonista.

Cuenta con un método update que recibe mensajes del tipo Audio y con el subtipo ReposicionListener. Estos mensajes se envían cuando se desplaza la el componente físico (y por lo tanto también el gráfico) del personaje protagonista, que el SoundListener interpreta y cambia la su posición en el espacio 3D. Por otra parte el movimiento de la cámara también envía mensajes de reposicionar el listener, pero en este caso lo que está variando no es la posición, sino el vector forward que nos indica la orientación del oyente.

OclusionComponent

Por último, este componente se asigna a entidades con presencia física que pueda suponer un obstáculo en la transmisión del sonido. En nuestro caso hemos colocado un muro en la escena para mostrar su funcionamiento. Es un componente simple que no tiene update.

En el momento en el que se asigna toma las medidas del cuerpo físico que lo contiene para establecer un geometry de FMOD que haga el efecto de oclusión del sonido con la posición y el tamaño del propio objeto.

Posibles mejoras y añadidos

Al no disponer de todo el tiempo que nos hubiera gustado para pulir la parte sonora y además partir de un juego bastante limitado jugablemente, no hemos podido añadir una sonorización excesivamente elaborada. Sin embargo sí que hay algunas ideas que quedaron en el tintero y que nos gustaría comentar:

- Explotación del system de alto nivel de FMOD Studio: con la importación de bancos de sonido con diferentes efectos habríamos podido añadir variedad en cuanto al sonido de pasos, disparos, saltos o gruñidos de enemigos. Conociendo el funcionamiento de variables implementadas por nosotros en esos efectos, podríamos modificar el pitch, la velocidad u otros aspectos de las muestras que tenemos consiguiendo así enriquecer el entorno del juego.

- Jugar con efectos y reverbs: la excesiva limitación del nivel jugable ha hecho que no pudiéramos introducir efectos o reverbs como de nivel submarino o de sala cerrada, ya que al final lo único de lo que disponíamos era la escena abierta en la que nuestra protagonista se desplaza acabando con los enemigos que se encuentra. De cualquier manera, a modo de ejemplo hemos añadido la posibilidad de cambiar entre distintas reverbs pulsando un botón del mando para saber más o menos cómo se escucharía el juego en caso de haber implementado distintas localizaciones con espacios sonoros diversos.

Imágenes ingame del proyecto: