PROYECTO SONIDO

1 INTRODUCCIÓN

Nuestra idea original para el proyecto era introducir sonido a algún videojuego o práctica que hubiésemos desarrollado durante la carrera. No obstante, nos dimos cuenta de que, en ninguno de ellos, por la causa que fuera, había potencial como para basar el proyecto de esta asignatura sobre él. Así que nos decantamos por realizar lo que llamamos una "demo técnica" realizada en Unity.

Lo que hemos tratado de hacer en este trabajo es utilizar muchas de las herramientas que nos ofrece FMOD para añadir sonidos a un juego, utilizando Unity como motor gráfico y físico. Para ello hemos utilizado la integración de FMOD en Unity, que no es más que un conjunto de Scripts que ayudan a integrar las liberarías nativas de FMOD en el motor de Unity.

2 LA DEMO

En la demo podemos controlar a un jugador (esfera roja) en tercera persona que se encuentra en un entorno cerrado.

2.1 EL ESCENARIO

El jugador aparece en un pasillo por el que puede moverse. En el escenario existen distintas habitaciones y salas a las que el jugador puede acceder libremente. Dentro de las salas se experimentan diferentes sonidos y músicas. Cada sala se centra en un aspecto del motor FMOD diferente y en las que se puede interactuar para experimentar con el sonido.

Imagen 1. El jugador

Por todo el escenario y en las salas se encuentran letreros que indican cómo interactuar en esa sala en concreto o que aportan información sobre lo que el jugador está oyendo.

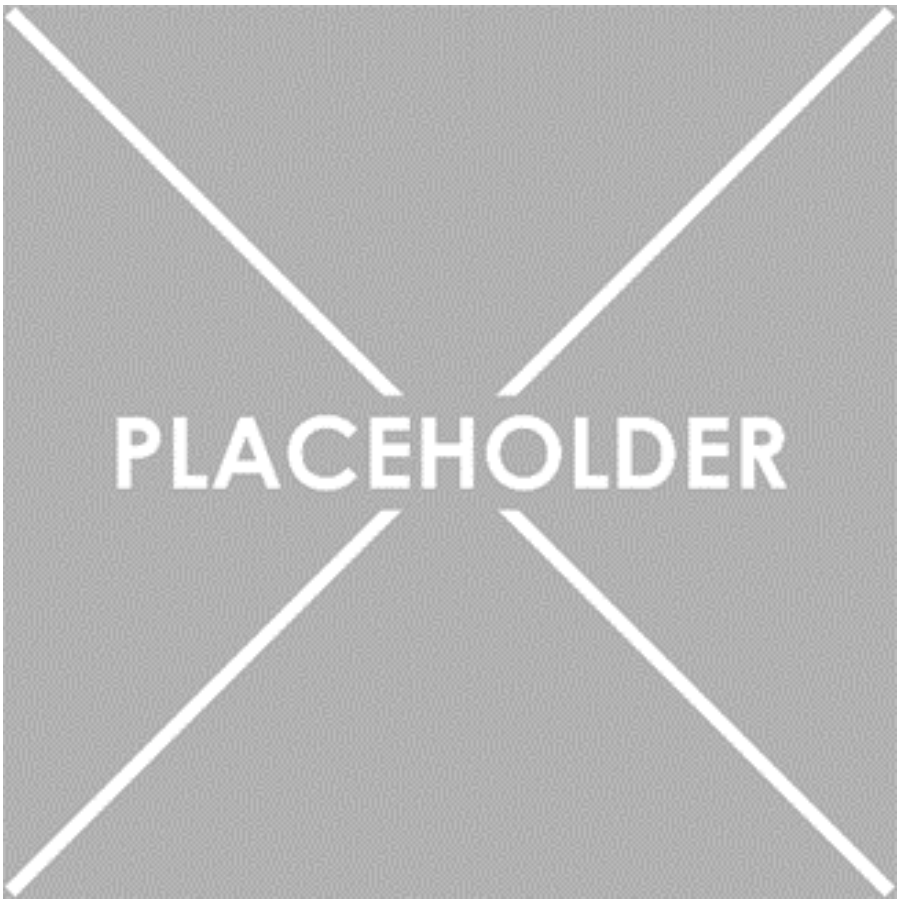

Imagen 2. El escenario

2.2 CONTROLES

Para mover al jugador utilizamos las teclas **AWSD** para moverlo en las 4 direcciones. Para rotar la cámara, podemos utilizar las **flechas de dirección** del teclado o el **ratón**. El jugado también puede saltar pulsando la tecla **espacio**, y además puede correr mientras se mueve pulsando la tecla **C**. En aquellas salas o lugares donde se pueda interactuar, se pulsa la tecla **E** para llevar a cabo la acción.

3 IMPLEMENTACIÓN

En este apartado vamos a hablar de las tecnologías que hemos utilizado para realizar el proyecto.

Para el apartado gráfico y físico hemos utilizado Unity como ya se ha dicho, ya que es un motor que nos facilita la creación de un entorno "jugable" y en el que tenemos experiencia.

Para el apartado sonoro hemos utilizado la integración de la librería de sonido FMOD para Unity. Hemos querido exprimir esta librería, y para ello decidimos utilizar tanto la API de FMOD Studio como la API Low Level que proporciona. Ambas dan una funcionalidad que coincide en algunos aspectos, pero a continuación vamos a detallar para que hemos utilizado cada una dentro de la demo.

3.1 FMOD

El Script principal que carga e inicializa la librería es el archivo **SoundManager.cs.** Es el principal componente de un *GameObject* con su mismo nombre y utiliza el modelo *Singleton*. Este se ocupa de, a la inicialización de la aplicación, cargar tanto el sistema de FMOD Studio como el sistema de Low Level. Además, hace como gestor de la librería mediante funciones públicas, por las que los demás objetos del juego le pueden pedir ambos sistemas.

3.1.1 FMOD Studio

Cuando hablamos de FMOD Studio hablamos tanto del programa que se utiliza para genera eventos y bancos de sonido, como de la API que nos proporciona la librería en su integración en Unity para manejar todos estos eventos e introducirlos dentro del jugo.

Así pues, el principal uso que le hemos dado a FMOD Studio ha sido para añadir los sonidos del jugador utilizando primero el programa y después creando y reproduciendo esos eventos dentro del juego.

Imagen 3. FMOD Studio

Los sonidos que produce el jugador se dan al andar o correr y al saltar. Para ello, en FMOD Studio, hemos creado diferentes eventos con sus parámetros que son lanzados y ajustados desde un Script que utiliza el objeto que gestiona todo el sonido del jugador. Por ejemplo, cuando el jugador se mueve el sonido de los pasos que produce es coherente con la velocidad de movimiento que lleva en ese instante.

3.1.2 FMOD Low Level API

Para el resto de los sonidos que no son producidos por el jugador hemos utilizado la API que proporciona FMOD a bajo nivel, y que habíamos utilizado antes durante la asignatura. El uso, a grandes rasgos, que le hemos dado ha sido el siguiente:

- **Sound3D (Script)**: durante todo el escenario, el jugador va a encontrarse con objetos que producen diferentes ruidos y sonidos. Esta funcionalidad viene dada por el Script Sound3D. Así, todos estos objetos sonoros tienen este componente. Los principales parámetros que pueden ajustarse en este Script son los siguientes:
    - **Name**: ruta y nombre del archivo de sonido que tiene que reproducir el objeto.
    - **Loop**: si se quiere que el sonido se ejecute en bucle o solo una vez.
    - **Min y Max**: distancias mínimas y máximas del sonido.

  Una vez ajustados estos parámetros el Script crea un sonido (*sound*), lo añade a un canal (*channel*) y llama a la función *System::playSound* para comenzar la ejecución del sonido. Además, para el posicionamiento del sonido en FMOD, se utiliza la posición del objeto en Unity, de manera que, si el objeto se mueve, la fuente del sonido siempre va con él.

- **SoundManager (Script):** además de ser el gestor de la API de la librería, también controla el sonido ambiental durante el juego.
- **Reverb3D (Script):** como indica su nombre, este componente añade reverb a un objeto que ya tenga asignado algún sonido.
- **FMODWall (Script):** cuando este componente se añade a un objeto que tiene como *MeshRenderer* un plano, se crea un *Gemoetry* en FMOD del tamaño del plano representado en la escena. Esto añade oclusión al sonido entre los objetos y el jugador. Se pueden ajustar los siguientes parámetros:
    - **Direct Occlusion:** parámetro que ajusta la oclusión directa que añade la pared entre el sonido y el jugador. Va de 0 (min) a 1 (max).
    - **Reverb Occlusion:** ajusta la oclusión de reverb que añade la pared entre el sonido y el jugador. Va de 0 (min) a 1 (max).
- **ParticleSystemCallback (Script):** cuando una explosión comienza, el sonido tiene que ajustarse con lo que el jugador está viendo para que éste sea coherente. Para ello, este script controla el momento en el cual el sistema de partículas debe emitir el sonido para que el resultado sea coherente.

4 RESULTADO

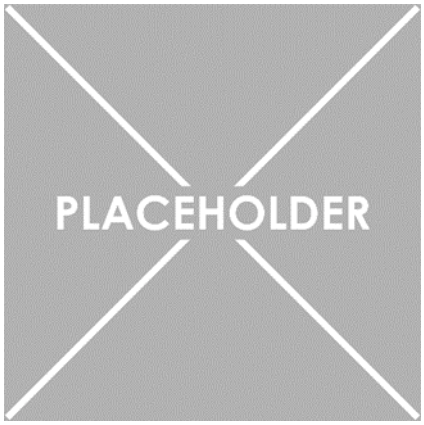

Imagen 4. Gameplay

Con todo esto, hemos querido utilizar en un sencillo juego las funcionalidades que nos resultaban más interesantes proporcionadas por la librería FMOD. El jugador puede experimentar mientras juega diferente sonidos y sensaciones, pudiendo interactuar con el escenario para alterar estos sonidos.

Al haber utilizado tanto FMOD Studio como la API de Low Level hemos aprendido a utilizar la librería de forma que hemos tocado todos los aspectos que querrían utilizarse a la hora de incluir sonido a un videojuego.

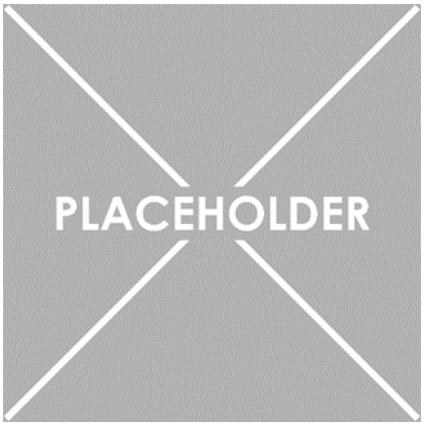

Imagen 5. Gameplay

5 COMPLICACIONES Y MEJORAS

Uno de los principales inconvenientes con los que nos topamos es que los eventos generados con la API de FMOD Studio y los sonidos generados en la API de Low Level **no** conviven en un mismo mundo. Esto quiere decir que, por ejemplo, un efecto de reverb aplicado en el sistema de Low Level solo afectará a aquellos sonidos producidos por ese sistema, mientras que los sonidos producidos por los eventos no se verán afectados. Continuando con el ejemplo, esto provoca que, si queremos que los pasos del jugador tengan reverb, este añadido tiene que hacerse en el evento en el programa de FMOD Studio, y modificar esos parámetros en el juego en tiempo real (por ejemplo, puede usarse RayCasting para ver el tamaño de la sala en la que se encuentra el jugador y si hay que añadir o no reverb).

Por tanto, una de las mejoras que podrían hacerse es añadir ese efecto de reverb a los pasos el jugador cuando se encuentra en una sala o el pasillo, por ejemplo.

Otra mejora podría ser el efecto de Fade Out y Fade In de la música ambiente cuando se cambia de sala, ya que, ahora mismo, la música se cambia de manera abrupta.