Sonorización dinámica de un proyecto de *Unity* usando *FMOD*

Autores

Miguel Ramírez Castrillo (migram05@ucm.es)
Alejandro Massó Martínez (almasso@ucm.es)

Link al video
Link al repositorio de GitHub (ver apartado de *releases* para descargar el ejecutable)

Idea del proyecto

La idea propuesta para el proyecto era la de sonorizar un juego de tipo *first person shooter* en Unity. Además, el proyecto incluye multijugador en línea, por lo que la sonorización funcionaría para todos los jugadores de una partida. Para ello, usamos el editor de la aplicación *FMOD Studio* para diseñar todos los eventos sonoros y los parámetros que los modifican. Luego, dentro del motor de videojuegos *Unity*, usamos el plugin de *FMOD* para llamar dichos eventos. Además, hacemos uso también de la *API* de *FMOD*, para poder aplicar algunos efectos directamente desde código.

Desarrollo del proyecto

Como punto de partida, usamos el plugin *Multiplayer FPS Template*, de Alteruna como base del proyecto. Este plugin proporciona todo el código necesario para crear y unirse a sesiones en línea. Además de un controlador, modelos y animaciones para un proyecto de disparos en primera persona.
Una vez configurada la plantilla para este proyecto ya teníamos la base preparada para expandirla. Cabe destacar que el plugin venía con sonidos incorporados que hacían uso del motor de audio de *Unity*. Habiéndose suprimido el motor de *Unity* por el plugin de *FMOD*, hemos decidido reusar algunos de los sonidos que incorporaba la plantilla con nuestro proyecto, añadiendo, eso sí, más para crear una mejor ambientación.

Para poder gestionar todos los eventos de *FMOD Studio*, y las llamadas a la *API* desde un mismo punto, hemos creado la clase *SoundManager*, que sigue el patrón de diseño *Singleton* para que sea accesible desde cualquier parte del código. Esta clase es la encargada de guardar y lanzar todos los eventos de *FMOD Studio*, así como de incorporar llamadas que permiten modificar los parámetros. Y, adicionalmente, es desde esta clase desde donde se hacen todas las llamadas a los métodos de la *API* de *FMOD*.

Pasos, caídas y disparos

Empezamos el proyecto creando los eventos de *FMOD Studio* para los sonidos de disparos, pisadas y caídas, debido a que su comportamiento es muy similar: eventos cortos que se van a reproducir frecuentemente y que queremos que tengan ciertas variaciones entre cada

ejecución. El primer paso fue buscar sonidos de uso gratuito en la página de *Freesound* (ver referencias al final del documento) que ajustamos ligeramente usando *Audacity*.
Una vez procesados los audios, creamos dos eventos 3D dentro de *FMOD Studio*.
Para el evento de disparos usamos un *scatterer* con las muestras de sonido, al que le aplicamos una pequeña variación en el *pitch* y en el volumen con cada reproducción.
El caso de los pasos y caídas es similar, pero utiliza tres *multi instruments* que se lanzan dependiendo del material sobre el que se esté caminando o cayendo. Al igual que para los disparos, se aplica una ligera variación al *pitch* y al volumen con cada lanzamiento del evento.

Sonidos de botones

Para sonorizar los menús del juego hemos decidido añadir un sonido cada vez que el usuario hace clic en cualquiera de los botones. Hacer este evento fue sencillo, partimos de un sonido de *Freesound*, que incorporamos a un evento 2D de *FMOD Studio*. Este evento está compuesto únicamente por un instrumento simple que lanza el sonido anteriormente mencionado, cada vez con unos parámetros de *pitch* y volumen ligeramente diferentes.

Recarga del arma

Para la recarga de la pistola del jugador, tenemos un evento posicional en *FMOD Studio* compuesto por un *multi instrument* con dos audios diferentes. Este evento tiene dentro tres regiones de loop que separan los diferentes sonidos que ocurren al recargar una pistola, además, estas regiones de loop han sido automatizadas para que estén completamente insonorizadas.
Para poder salir de una región de bucle es necesario que un parámetro de *FMOD* “supere una cuota”. Este parámetro de *FMOD* se va aumentando progresivamente durante la ejecución a medida que la animación del arma avanza. De esta forma, el sonido va sincronizado con la animación, independientemente de la velocidad de esta.

Resonancia

Para los sonidos posicionales (disparos, recarga y pasos), hemos aplicado un efecto de resonancia utilizando el plugin de *FMOD Studio Resonance Audio*, creado por *Google*. Este plugin permite tener una sensación de sonido más realista en entornos cerrados, lo cual aprovecharemos para utilizar en la parte inferior de nuestra estructura en el mapa. Para ello hemos reemplazado todos los *spatializers* por defecto de nuestro proyecto de *FMOD*, sustituyéndolos por un *Resonance Audio Source*, que incorpora el plugin, y, posteriormente, en *Unity*, en los lugares donde queríamos tener un buen control de la resonancia de los sonidos, hemos creado objetos vacíos a los cuales les hemos aplicado un componente *Fmodresonanceaudioroom*, estableciendo sus propiedades como mejor nos ha parecido.

Música de fondo

La idea para la música de fondo era tener varias canciones distintas, y que se seleccionase una entre ellas para reproducir, dependiendo del nivel de peligro del jugador. En nuestro caso, tenemos tres canciones que se reproducen en función de la vida restante del jugador. Para ello modificábamos un parámetro de *FMOD Studio* en función del estado de la vida y dependiendo del valor, automatizamos las pistas para que se silenciaran todas menos una.

Además, incorporamos una automatización adicional por la cual, al recargar se aplicaría un filtro de paso bajo a la canción actual hasta que terminase la recarga. Para hacer eso, usamos eventos de animación, al igual que para el evento de recarga anteriormente mencionado.

Por último, queríamos modificar la velocidad de reproducción de la música de fondo en función de la velocidad del personaje. En un principio abordamos esta tarea usando únicamente *FMOD Studio*: automatizando el *pitch* en función de un parámetro y compensando la variación con el evento *pitch shifte*r. No obstante, aunque funcionaba bastante bien, se podía percibir un poco de variación de la frecuencia de la música durante el juego. Esto se debe a que el *pitch shifter* no es capaz de enmascarar a la perfección la modificación de frecuencia en todo momento.
Para solucionar esto, decidimos buscar *módulos* que estuvieran compuestos para *trackers*, de manera que se pueda acelerar el ritmo al que se lanzan los sonidos de una canción sin alterar su velocidad de reproducción.
Para buscar los *módulos* hicimos uso de la página web *The Mod Archive*. Las tres canciones que usamos se encontraban en el mismo archivo .mod, por lo que para poder separarlas en eventos y editarlas ligeramente, usamos el tracker *OpenMPT*.
El lanzamiento de eventos así como la creación del efecto de paso bajo se hace desde el singleton *SoundManager*.

Aunque no se usa, el proyecto de *FMOD Studio* contiene los eventos relacionados con la música de fondo, así como sus automatizaciones correspondientes y todos los parámetros que se iban a utilizar en un principio.

Lista de sonidos utilizados

- *Multiplayer FPS Template*, de Alteruna
- *xtd_-_game.mod*, de Xtd
- *Blood Frame PNG*, de PngKey
- *Button_Click1*, de BaggoNotes
- *pistol reload sound*, de GFL7
- *M9 Reload*, de oneshotofficial
- *Pistol Shot 1*, de TB0Y298
- Pistol Shot, de EvanBoyerman
- *9mm pistol shot*, de michorvath
- *Footsteps of a runner on gravel*, de florianreichelt
- *Concrete footsteps*, de SoftDistortionFX
- *Steps-Dirt-Falling_1a.ogg*, de nuFF3
- *Steps-Tile-Falling_1a*, de nuFF3