Gestor de sonido con FMOD

Animación por Computador y Videojuegos

Carlos Martínez Pérez (carmar45@fiv.upv.es)

ÍNDICE

1.Introducción

Durante el desarrollo de la asignatura de ACV, se han estudiado metodologías, técnicas y herramientas (apis, motores de físicas...) que permiten el desarrollo de un videojuego.

En este trabajo se va a implementar un gestor de sonido en un videojuego ya desarrollado. El objetivo es poder extraer todo el trabajo de gestionar el sonido al propio gestor, asi quedaría independiente al videojuego y cuando quisiéramos cambiar de gestor simplemente podríamos utilizar otro API.

Además, con esto se consigue una mejor implementación, mucho más clara, y si también podríamos exportar nuestro gestor a otro proyecto.

El videojuego que vamos a modificar es el Space Invaders, y el gestor de sonido que vamos a implementar utilizará la API del FMOD versión 3.75.

2.Fmod

El fmod es un api muy conocida de sonido. Puede ser utilizada en la mayoría de plataformas importantes de la industria (consolas, ordenadores, móviles...). Actualmente la versión es la 4.50, nosotros utilizaremos una previa, la 3.75. Se adjuntará la documentación del API 3.75 para futuros desarrollos.

Como hemos podido constatar a lo largo del recorrido del código original, se utilizan llamadas al propio FMOD desde el código principal, lo cual provoca muchos problemas a la hora de realizar cualquier cambio (añadir músicas, cambiar la ruta de las mismas...).

De los tipos de datos internos del FMOD vamos a utilizar los FSOUND_STREAM* al ser uno de los tipos que permiten trabajar con MP3 (los archivos de música que tenemos son de esta extensión).

FMOD se utiliza siguiendo los siguientes pasos:

1) Inicialización: Se llama a la función FSOUND_Init del propio API.
2) Carga de los ficheros: Se utiliza FSOUND_Stream_Open pasándole la ruta y otros parámetros. Solo se debe realizar una vez.
3) Reproducción de los ficheros: Llamamos a la función FSOUND_Stream_PlayEx. Tantas veces como queramos.
4) Cierre de los FSOUND_STREAM* y cierre del FMOD.

Una vez hemos explicado más o menos como funciona el FMOD vamos a pasar a explicar la parte de desarrollo.

3. Implementación

A pesar de la que la implementación del gestor de sonido se ha realizado principalmente en los ficheros del SoundManager y de Sound, hemos estado obligados a recorrer prácticamente todo el proyecto. Desde la inicialización hasta el cierre de la aplicación, pasando por la carga de los niveles, los estados para la reproducción de la música...

Primero se realizaron algunas modificaciones en el parser de ficheros de entrada, en el cual se obtiene la ruta donde van a estar los ficheros de sonido (y se le indica al SoundManager, el cual la almacena).

```
SoundsManager.setTrackPath(CGM_MUSIC,rText.GetString());
```

Se ha dejado preparado el gestor para poder albergar más de un sistema de sonidos, por ejemplo si tenemos músicas por un lado, voces por otro etc (se pueden añadir más elementos).

```
typedef enum
{
        CSM_MUSIC,      ///<Music
        CSM_SOUNDS,             ///<Sound
        CSM_VOCAL,      ///<Vocals
        CSM_MAX_SOUND_TRACKS        ///<For accounting purpouses only
} CSM_SOUND_TRACKS;
```

Una vez realizada la lectura del fichero de inicialización y tenemos el directorio donde se almacenan las músicas, se realiza la inicialización del gestor de sonido y de los sonidos, para dejarlos preparados para la reproducción.

Los nombres de los sonidos se almacenan en el archivo GameSounds.h, en un tipo enumerado y los ficheros a los que llaman se pueden encontrar en GameSounds.cpp.

Por otro lado ,se pretende implementar en un futuro más apis, por ello se ha hecho distinción en el código al usar el api de fmod con estas ordenes:

```
#ifdef CSND_FMOD375
                //Orden del FMOD
#elif CSND_FMODEx
                /// Complete
#elif CSND_OAL
                /// Complete
#endif
```

Se deberán completar en un futuro las dos opciones nuevas (OAL y FMODex).

Comentar también que todas las llamadas al FMOD en el proyecto han sido sustituidas por llamadas al SoundsManager.

3.1. Sound

La clase Sound representa un sonido, con ella se establecerá el enlace entre los sonidos y el FMOD (en su mayor parte). El SoundsManager sólo deberá manejar los sonidos para el correcto funcionamiento de la aplicación.

Un objeto de clase sound contiene estos atributos:

1)*FSOUND_STREAM* Sound*: Donde se almacena el Stream del sonido.
2)*int Volume*: El volumen asociado al sonido
3)*int Position*: Posicion que queremos utilizar con el sonido (para futuras implementaciones)
4)*char Path*: Ruta al sonido.
5)*char Name*: Nombre del sonido.

6)*char Ruta*: Puntero a el nombre del directorio donde está el sonido

Y contiene los siguientes métodos:

1)*void Init(char *P,int v):* Inicializa un nuevo sonido
2)Constructores.
3)*void Play (FSOUND_DSPUNIT* dspUnitSounds,int v):* Reproduce el sonido con un volumen determinado (y modifica el volumen propio del sonido).
4)*void setVolume(int v):* Modifica el volumen asociado al sonido.
5)*void setPosition(int position)*: Modifica la posición asociada al sonido.
6)*FSOUND_STREAM* getStream()*: Devuelve el stream asociado al sonido.
7)*void close()*: Cierra el Stream del sonido.

También tenemos los diferentes volúmenes posibles de un sonido:

```
#define CSND_MUTE           0      ///< Volume muted
#define CSND_LOW_VOLUME     32     ///< The low value of volume
#define CSND_MEDIUM_VOLUME  128    ///< The medium value of volume
#define CSND_MAX_VOLUME     255    ///< The maximun value
```

3.2 SoundsManager

La clase SoundsManager es aquella que permite manejar todos los objetos de la clase Sound. Además, enlaza todas las llamadas necesarias a los sonidos desde el resto de la aplicación.

Para ello, la clase SoundsManager posee los siguientes atributos:

1)*char TrackPath[CGM_MAX_SOUND_TRACKS][CSM_LONG_PATH_NAME]*: Ruta de los sonidos (music,sounds,vocals...)
2)*FMUSIC_MODULE * my_mod_about*: Necesario para abrir el FMOD.
3)*std::vector<CSound> Sound*: Vector of the sounds of the app.

Y usa los siguientes métodos:

1)*void LoadSound(void)*: Carga todos los sonidos que van a ser utilizados en el juego.
2)i*nline void Play (const unsigned char s,int vol){Sound[s].Play(dspUnitSounds,vol);}*: Reproduce un sonido (llamando a la clase sound).
3)*void SetVolume(const unsigned char s, int vol)*: Modifica el volumen de un sonido.
4)*void SetVolumeAll(int vol)*: Modifica el volumen de todos los sonidos.
5)*void SetPosition(const unsigned char s,int pos)*: Modifica la posición de un sonido.
6)*static char *ErrorMsg(const unsigned int)*: Devuelve un error del SoundsManager.
7)*void setTrackPath(CGM_SOUND_TRACKS Track,const char* path)*: Establece la ruta donde estarán los sonidos.
8)*const char* getTrackPath(CGM_SOUND_TRACKS Track):* Devuelve la ruta donde están los sonidos.

Como se puede apreciar, el SoundsManager no utiliza métodos del FMOD, solo se limita a llamar a los métodos de cada Sound que si las utilizan. La inicialización y el cierre del FMOD si se hacen desde aquí.

4. Conclusiones

Como conclusiones para el trabajo, decir que la experiencia ha sido muy completa y satisfactoria, se ha podido comprobar el funcionamiento de un videojuego desde una visión global y se ha recorrido una buena parte de él (desde la incialización hasta el final).

Se ha conseguido extraer toda la parte del manejo del sonido fuera de la aplicación principal, con lo que podremos utilizarla en otras implementaciones de una manera más sencilla.

El proyecto en si no ha sido difícil, pero si costoso de entender, al tener todo el código mezclado. Una vez se ha entendido ya es mucho menos costoso implementar nuevos elementos.