FMOD X SDL

Índice

3. Resultado

1. Introducción

Como proyecto final pensamos en hacer un motor, usando SDL, una librería para renderizado, gestion de input, y otras muchas cualidades, en este proyecto solo hemos usado esas 2, y FMOD, una librería de audio 3D, la que ha sido explicada en clase.

Nuestro objetivo era conseguir un motor sencillo para poder hacer un juego sencillo. Esta es la lista de las características del motor que hemos intentado realizar:

- Con una única escena
- Objetos con comportamiento basado en componentes
- Sistema colisiones sencillo
- Poder realizar mapas con la herramienta Tiled
- Renderizado gráfico sencillo
  - Figuras geometrícas sencillas
  - Cámara básica
- Gestion del sistema de sonidos
  - Creación de sonidos
  - Gestión de sonidos
  - Play/Pause/Volumen
  - Movimiento
  - Orientación (conos de sonido)
  - Creación de geometrías
  - Gestión de geometrías
  - Moverlas
  - Cambiar propiedades
  - Creación de reverberaciones
  - Gestion de reverberaciones
  - Gestión del `Listener`
  - Posición
  - Orientación
- Crear documentación

---

2. Desarrollo

Como dijo Jack el Destripador, vayamos por partes.

Una única escena

Esto no causo apenas problemas, decidimos que era la mejor opción ya que sino empezariamos a complicarnos demasiado y no estar en lo importante, el sonido. En la clase `SDLApp` se gestiona dicha escena, mediante una lista de objetos, renderizandola y actualizandola.

Objetos basados en componentes

Esto era clave para nuestro *motor*, ya que con este tipo de estructuras solo necesitas hacer una buena base y la creación de contenido se simplifica mucho más, sobre todo a la hora de crear comportamientos. Esto lo conseguimos de forma sencilla haciendo que los componentes sean aislados y no pueden tener dependencias de otros componentes del objeto.

Sistema de colisiones

Hay colisiones que se comprueba activamente, si dicho objeto colisiona con otro, todo esto lo gestiona la clase `CollisionWorld`.

Realizamiento de mapas con herramienta

Podemos gracias a una *Librería* que realizo Fran en 2º de carrera para la asignatura Proyecto 2. TMXReader.

Renderizado gráfico sencillo

Todo lo relacionado con los gráficos lo lleva la clase `GraphicManager`.

Figuras geometrícas sencillas

Dibujamos rectangulos, lineas, puntos y hasta circulos, con color y todo, de todo esto se encarga SDL, decidimos no perder mucho el tiempo en que podemos dibujar.

Cámara básica

Hay camara, pero todavía se podría perfecionar algo más para tener algo más decente, como poder trackear un objeto en particular y esas cosas. Y que fuera un componente para tener mas coherencia con la idea del *motor*

Gestión de sonido

Todo lo relacionado con la gestion del sistema de sonidos esta en la clase `SoundManager`.

Creación de sonidos

Puedes crear sonidos, solo necesitas el origen del fichero que quieres abrir y puedes crear un sonido, y un canal, un problema es que no se gestiona si ese sonido ya ha sido metido, simplemente se crea un nuevo sonido con un canal en loop.

Gestión de sonidos

Toda la gestión de un sonido de forma indvidual, lo gestiona el componente `SoundComponent`

Play/Pause/Volumen

Puedes hacer estas 3 cosas

Movimiento

Podemos posicionar el sonido a la posición del objeto del que está asociado.

Conos

Lamentablemente no hemos realizado está parte

Creación de geometrías

Podemos crear geometrías, para bloquear sonidos o quitarles fuerza. Solo creamos geometrías en el plano XY ya que el juego es

en 2D, y usamos la dimensión Z de `FMOD` como la Y en el motor.

Gestion de geometrías

Toda la gestión de las geometrías de forma indvidual, lo gestiona el componente `WallComponent`

Moverlas

No las movemos no nos parecio relevante una vez que lo pensamos bien

Cambiar propiedades

Puedes cambiar la oclusión directa y de reverberación, la propiedad de dos caras, no, no lo consideramos relevante.

Creación de reverberaciones

Podemos crear reverberaciones, solo con el radio máximo, el radio mínimo es un 1/4 del máximo

Gestión de reverberaciones

Toda la gestión de las reverberaciones de forma indvidual, lo gestiona el componente `Reverb3DComponent`. Puedes cambiar la X y la Y. Pero nada más.

Gestión Listener

Toda la gestión de las reverberaciones de forma indvidual, lo gestiona el componente `ListenerComponent`.

Posición

Puedes cambiar la posicion del Listener

Orientación

Puedes cambiar la orientación

Creacion de documentación

Esto ha sido una pena no poder conseguirlo ya que comentamos todo el código de tal forma que podriamos un XML para luego procesarlo, pero no supimos hacer el último paso.

---

3. Resultado

Haciendo un resumen con la lista de características anterior hemos conseguido lo siguiente:

- [x] Con una única escena
- [x] Objetos con comportamiento basado en componentes
- [x] Sistema colisiones sencillo
- [x] Poder realizar mapas con la herramienta Tiled
- Renderizado gráfico sencillo
  - [x] Figuras geometrícas sencillas
  - [ ] Cámara básica
- Gestion del sistema de sonidos
  - [x] Creación de sonidos
    - Gestión de sonidos

- [x] Play/Pause/Volumen
- [x] Movimiento
- [ ] Orientación (conos de sonido)
- [x] Creación de geometrías
  - Gestión de geometrías
- [ ] Moverlas
- [x] Cambiar propiedades
- [x] Creación de reverberaciones
- [x] Gestion de reverberaciones
  - Gestión del `Listener`
- [x] Posición
- [x] Orientación

- [ ] Crear documentación

Hubiera estado bien tener algo jugable a partir de este *motor*, pero no ha sido posible, por varios motivos.

Gracias a esto hemos conseguido aprender más la API de `FMOD` y perfeccionado nuestras habilidades a la hora de programar.