PEC3: La Cueva de los Condenados

Pedro Jesús Sánchez Illescas

June 10, 2024

Índice

1 Librerías utilizadas

- SDL 2.30.0
- SDL-mixer 2.8.0
- SDL-ttf 2.22.0

2 Proceso de creación

Este proyecto ha tenido un desarrollo bastante rápido, puesto que al igual que las prácticas anteriores, está basado en el mismo motor de efectos utilizado en la PEC1 con unas ligeras variaciones:

a) proceso de entrada de teclado, y

b) Carga general de imágenes eliminada para quitar la dependencia con la librería SDL-image.

Esto hace que toda la lógica importante para la PEC esté concentrada en la clase *EffectCave*. El juego consta de tres estados lógicos:

1. Fase de juego: Ocurre la interacción con el jugador
2. Fase de finalización: Según la condición de finalización del juego, sonará la música de victoria o de derrota
3. Fase de juego terminado: El programa espera a que el jugador pulse la barra espaciadora para empezar una nueva partida

Las fases de finalización y juego terminado son muy simples. En la fase de finalización hay que esperar a que termine la música que corresponda a la condición de finalización, esto es, salir de la cueva o que el monstruo se coma al jugador. Al mismo tiempo sale un mensaje por pantalla explicitando el final. A su vez, la fase de game over, o de start, el programa espera a que el jugador pulse la barra espaciadora para empezar una partida nueva.

Casi la totalidad de la complejidad de la práctica está en la fase de juego, donde el jugador puede moverse en una cuadrícula con un control WASD tipo tanque (movimiento hacia adelante y hacia atrás, girando hacia los lados) en una pantalla oscura intentando encontrar el origen de una cascada, cuyo sonido aumentará de intensidad cuanto más cerca está del mismo. A su vez, en la cueva también hay un monstruo durmiendo que hay que evitar usando únicamente sus ronquidos para intentar intuir su posición. El monstruo se moverá una casilla en dirección aleatoria cada vez que el jugador intente moverse a una casilla en la que haya una pared, oyéndose además el sonido del golpe.

Para implementar este gameplay, se ha usado un filtro a los sonidos de ronquidos y de la cascada de modificación de su intensidad vía la función *Mix_SetPosition*, que modifica el volumen de un canal como si se escuchara alejado del foco de sonido y el paneo para simular la orientación de la fuente de sonido. Hay que tener en

cuenta que *Mix_SetPosition* considera una "distancia" normalizada entre 0 y 255, donde 0 significa que la fuente de sonido y el jugador comparten posición y 255 es la máxima distancia a la que se puede estar del sonido, quedando inaudible. Con esto en mente, la regla para modificar el argumento de distancia para *Mix_SetPosition* es $f(d) = 255 \cdot d/BOARD\_DIAGONAL$ usando la distancia entre dos puntos $\vec{p_1}$ y $\vec{p_2}$, $d = \sqrt{(\vec{p_1} - \vec{p_2}) \cdot (\vec{p_1} - \vec{p_2})}$. Para ello se ha usado como límite superior el valor de la diagonal del tablero, que como es de 10x10 casillas, tenemos que $BOARD\_DIAGONAL = 10\sqrt{2}$, asumiendo que las casillas son cuadrados de 1m de lado.

Y de la misma manera, para calcular el ángulo a la que se encuentra la fuente de sonido, usamos la función $\theta((x, y)) = atan2(x \cdot d_y - y \cdot d_x, y \cdot dy + x \cdot d_x)$, siendo $(x, y) = (\vec{p1} - \vec{p2})/d$ el vector normal que apunta desde el jugador a la fuente de sonido y (d_x, d_y) es la dirección a donde el jugador está mirando.

Como extra, se ha añadido un modo de depuración donde se "dibuja" el nivel con caracteres ASCII, para poder evaluar de una forma más fácil si los efectos de distorsión del sonido se estaban aplicando de forma correcta. Este modo de depuración puede activarse y desactivarse pulsando la tecla T del teclado. Los códigos de las casillas es el siguiente:

- XX: Pared
- M: Monstruo
- P[N—W—S—E]: jugador junto a la dirección a la que se encuentra (norte, oeste, sur y este)
- S: Casilla de salida
- IT: Casilla destino

3 Compilando la aplicación

El programa se compone de los fuentes localizados dentro de la carpeta */src*. Para poder lanzar correctamente el ejecutable generado, la carpeta */assets* debe estar en su mismo directorio para poder cargar los sonidos necesarios.

4 Conclusiones

Esta práctica ha consistido en hacer un juego sin gráficos donde se comprueba que un apartado sonoro trabajado puede, además de hacer un gameplay interesante, aumentar la inmersión dentro del juego, mejorando muchísimo la experiencia del usuario. Para su implementación se ha partido del motor simple de efectos implementado para la PEC1, resultando muy fácil la implementación del juego como un efecto completo, quedando su implementación reducida a su propia clase, teniendo toda la lógica concentrada en un mismo archivo.