Memoria de un juego hecho en Unreal

Sergio Roselló

June 2018

Contents

1 Inspiración e idea general

El juego esta basado en el ejemplo de clase. He reutilizado el aspecto visual del personaje, además de el mapa. Esto me ha servido para saber como funcionaban las cosas en Unreal ya que es la primera vez que me enfrento a este motor. Poco a poco siento que he ido cogiendo soltura y he podido expandir sobre lo aprendido, como por ejemplo el uso de Inteligencia Artificial con los enemigos.

2 Controles

Las mecánicas en este juego son poco intuitivos ya que esta pensado para que tenga una ligera curva de aprendizaje. Con las teclas "w, a, s, d" se puede controlar el movimiento del personaje. Para que el personaje apunte, tienes que mover el ratón, como en todos los juegos, la diferencia en este es que la cámara se mueve con el movimiento de rotación del personaje, de forma que este este siempre mirando hacia delante y que parezca que se mueva el mapa en lugar del personaje. Este no es el caso.
Para abrir el menú de pausa, se presiona la tecla M de "Menú".
Para disparar, se usa el boton izquierdo del raton.

3 blueprints

Prácticamente todos los elementos del juego tienen algún tipo de lógica controlada por blueprints. Este sistema visual, aunque fácilmente entendible, me ha parecido bastante complicado de empezar a usar porque tenia que acordarme de los nombres exactos de las cajas. Al final, con practica, puedo entender porque la gente usa el sistema ya que puede llegar a ser mas rápido que implementar la lógica del juego en c++, siempre y cuando tengas practica.

3.1 player

El player blueprint, además de gestionar su comportamiento, se encarga tambien de la logica de los HUD. Los comprtamientos que tiene son:

- Movimiento: Usa los ejes de movimiento que interpreta Unreal para mover al player en el mundo.
- Disparo: Se encarga de disparar cuando se presiona el botón izquierdo del raton.
- Vida: Cuando se llama al evento AnyDamage, se reduce en uno el numero de vidas que tiene el player hasta que se el numero de vidas llega a 0. En esa caso, vuelve a cargar el nivel 1. Cuando recibe daño, ademas de reducir la vida, pone la variable "Invulnerable" a True durante 0.6 segundos.
- Aspecto visual de que se le ha quitado vida al jugador: Cuando Invulnerable es verdadero, parpadea al jugador.

3.2 Enemy

Este enemigo camina en direcciones aleatorias mientras dispara al jugador. Si colisiona con una pared, cambia de dirección y si el valor aleatorio entre 0 y 1 supera 0.98, también cambia de dirección. Tiene una serie de métodos personalizados que simplifican el blueprint general.

- GetVectorFromDirection: Convierte los datos del enum a vectores que puede entender Unreal.
- SetRandomDirection: Segun un valor aleatorio entre 0 y 3, cambia el valor de la variable Direction a una distinta a la anterior.
- GetDirectionFromInt: Según el entero de enrtada, devuelve un enum direction distinto.

El enemigo se encarga de moverse, calcular direcciones nuevas a las que moverse y disparar. La lógica de la primeras dos secciones se convierte prácticamente trivial gracias a los métodos explicados anteriormente. Para disparar al jugador, el enemigo debe averiguar la posición del mismo, por tanto lo hace mediante una resta de vectores.

3.3 SecondEnemy

Este enemigo parte de la base del enemigo anterior, así que sigue teniendo los mismos métodos que el anterior. Aunque parece que hace lo mismo desde fuera, usa IA para detectar al jugador en un rango dado por movimiento. Cuando lo percibe, cambia su comportamiento y deja de moverse de forma aleatoria para dispararle mientras se mueve hacia el. Los métodos que usa son:

- Fire: Igual que el del enemigo anterior excepto que depende de si puede ver al player. Si no le puede ver, no dispara.
- AIPerception: Si percibe al enemigo en el rango definido, cambia el booleano "CanSeeEnemy" a True para que el enemigo dispare.
- Movement: Tiene dos comportamientos, depende de la percepción del jugador. Si no lo percibe, se mueve de forma aleatoria, como el enemigo "enemigo". Si lo percibe, se mueve hacia el.

3.4 Boss

El comportamiento del Boss es parecido al del segundo enemigo, aunque mas agresivo. Al enfrentarte a el en uno contra uno, le he puesto vidas. Cuando muera, se carga la pantalla de victoria.

3.5 DoorToNextLevel

Se abre si has matado a todos los enemigos de la sala. Quería poder poner varios tipos de enemigos en la misma sala, en el mismo nivel, por tanto, tenia que cambiar la lógica de la puerta de ejemplo. Se le pasan a la puerta todos los enemigos de la sala en el editor, esto hace que pueda ir comprobando y eliminando los que ya ha matado el player. Una vez todos los enemigos han muerto, se desliza hasta dejarte paso.

3.6 Bullets

Las balas que disparan, tanto los enemigos como el personaje. Tienen varios métodos en común:

- Muere si colisiona con una pared
- Mueve la bala en una dirección a una velocidad
- Reproduce un sonido cuando se activa
- Muere si pasan mas de 5 segundos
- Hacen daño con quien colisionan

Ambas balas son bastante parecidas.

3.7 Niveles

El juego tiene dos niveles, la pantalla de inicio, en la que se le presentan al jugador varias opciones, jugar, ajustar la resolución y salir del juego. Si presiona jugar, se carga el primer nivel del juego.

3.8 User Interface

Tengo tres interfaces hechas con UMG.

- HUD: Interfaz que contiene la vida del jugador y la puntuación. Ambas variables están controladas por sergioGameInstance, que asegura que no se van a perder cuando se cargue un nuevo nivel. En este HUD, tengo dos metodos para actualizar la vida y la puntuación del jugador.
- MainMenu: Esta interfaz es la que aparece cuando el juego se inicia.
- PauseMenu: Esta interfaz se activa pulsando la tecla M el el teclado. Presenta dos opciones, continuar con la partida o salir al menú principal.
- You Win: Aparece cuando mato al boss final.

3.9 SergioGameInstance

Este GameInstance es el que me he creado para almacenar las variables que quiero que el jugador mantenga durante los niveles. Estas variables son: PlayerHealth y PlayerScore. El playerHealth se lo asigno al player cuando se carga el primer nivel, de esta forma no se reestablecen las vidas al cargar siguientes niveles. También asigno la puntuación del player a 0.

3.10 Otros

El resto de objetos como musica, el enum, Fuentes, GameMode, imágenes, materiales y Mesh sirven para completar el juego.