Unreal Test

The Breach Studios

Miguel Escuer Sánchez 22/05/2022

Introducción

En este documento explicaré cual ha sido el proceso que he seguido para realizar la prueba en Unreal Engine, comentar problemas encontrados e implementaciones que, debido a la falta de tiempo por un *deadline* en mi trabajo actual (que me ha llevado horas extra) me hubiera gustado poder hacer.

En primer lugar, me gustaría comentar que el proyecto subido no es del todo jugable y que para poder probar *gameplay* es necesario abrir el *ThirdPersonMap*, donde podremos ver el sistema de disparos desde 2 clientes o más. Se ven replicados en todas las máquinas simuladas desde Unreal Engine tanto los jugadores como los proyectiles disparados.

A continuación, se presenta cada una de las partes que componen el proyecto.

1. Gameplay

Comencé este proyecto creando los componentes del *character.* En particular he implementado las clases *HealthComponent* para la gestión de la salud del jugador o IA y *WeaponComponent* para la gestión de las armas utilizadas.

Si bien se ha implementado el juego gestionando jugadores y una sola clase de arma (lanzador de bolas), el hecho de haber implementado las clases como componentes permite añadir escalabilidad al uso de los mismos componentes a una posible IA futura, así como gestionar múltiples armas. Si hubiera introducido las propiedades de salud o armamento directamente al *character*, esta escalabilidad sería más compleja.

Tras la implementación de los componentes del *character* investigué cómo funciona un sistema multijugador (ya que nunca he trabajado con ello) y así encontré toda la información y documentación relacionada con *GameState*, *GameMode*, *PlayerState* y para que se usaba cada uno de ellos.

Con respecto a la implementación del sistema multijugador, hay que tener en cuenta que la información sobre el jugador como el *team* al que pertenece, nivel o estado (vivo o muerto) debe ser conocido por el *PlayerState*, además de quedar registrada en *GameMode* (servidor) y copiada en el *GameState* del resto de los clientes (toda o parte de la información). Para ello, he utilizado RPCs (etiqueta *server*) para actualizar información del *GameMode* (transcurso de la partida) por parte de los jugadores y para informar sobre si se ha ejercido daño a otro cliente con el proyectil lanzado por el *WeaponComponent* básico.

2. Sesiones

Tras estudiar como funciona el sistema de redes propio de Unreal implementé un sistema de creación, unión, listado y destrucción de sesiones de juego apoyándome en un tutorial sobre este tema. Este sistema también podría ser usado para la creación,

listado y unión de los jugadores a las partidas en una futura implementación del menú principal.

La idea propuesta es que, una vez iniciada la partida, apareciese un selector de héroes con un contador de tiempo. De forma interna, mientras se elige dicho héroe, haría un *matchmaking* consistente en sumar el nivel de cada jugador (nivel completamente ficticio) con todas las posibles combinaciones (sin dejar una desproporción de equipos mayor a 2/4 en número de jugadores) hasta dar con la más equilibrada (cuya resta sea más próxima a 0).

Tras la selección de héroes y transcurrido el tiempo indicado, se inicia la partida. En dicha partida, yo hubiera optado por implementar un sistema sin *respawn*, donde una vez quede un equipo sin jugadores vivos, este equipo pierde la partida, notificándose al *GameMode* y volviendo al *lobby*. Tras ello, se incrementaría el nivel pertinente a cada jugador vivo y se podría volver a empezar una nueva partida.

3. UI

Al considerarse inicialmente como lo menos relevante para la prueba se ha dejado la interfaz para lo último, a pesar de ser uno de los campos donde más experiencia poseo con Unreal Engine. En concreto, se han creado las clases para el HUD y los distintos *widgets* usados *ingame* y en el *lobby* o *prematch*, que pretendían ser totalmente programados en C++ haciendo solamente uso de *widgets blueprints* (emparentados con los *widgets* C++) para el posicionamiento en pantalla de estos. Como se puede apreciar no se ha implementado nada funcional referente a UI. Pudiendo ver en *Entry Level* un sutil esbozo del menú de creación de sesiones y unión.

4. Otras implementaciones.

IA hecha con *behaviour trees* de Unreal Engine, creando sus nodos de comportamiento desde C++ utilizando clases con herencia de la clase *UBTTaskNode*. Esta parte me interesaba implementarla, ya que he tenido experiencia previa programando la IA de un juego multijugador en pantalla partida, pero tratándose de un apartado opcional he preferido centrarme en otros elementos, lo cual me ha impedido desarrollar la IA.

El planteamiento propuesto es, para evitar tener *teams* de proporción dispar, rellenar los *teams* con *bots* en caso de faltar jugadores. Dichos *bots* tendrían diversos niveles de dificultad de IA en función del nivel de los jugadores del *team* rival vs el resto de jugadores de los del *team* del que forma parte el *bot*. Para estos distintos niveles de dificultad se pueden tener en cuenta variables como: cantidad de movimientos aleatorios durante el *Go to* para esquivar ataques, mayor o menor probabilidad de tomar cobertura, mayor o menor probabilidad de disparar y de acertar en el disparo. Quedarían descartadas “trampas” como aumentar velocidad, vida o daño, pudiéndose interpretar injustas por los jugadores rivales.

El sistema de cobertura planteado constaría de puntos de interés conocido por la IA, de los cuales tendría información sobre si están ocupados, si están en el campo de visión

de los enemigos (por ejemplo, trazando un *raycast*) así como la distancia a los mismos, pudiendo aumentar la probabilidad de tomar dichos puntos de cobertura en función de la salud del *bot* controlado por la IA.

Conclusión

A pesar de no haber tenido casi tiempo para hacer la prueba por el motivo laboral ya comentado y mi desconocimiento previo casi absoluto de juegos en red, he disfrutado mucho aprendiendo como funciona el sistema en red de Unreal Engine.

Si bien el resultado no es el que me hubiera gustado presentar, me llevo una experiencia que ya se queda conmigo y me sirve para cada vez conocer más Unreal Engine y mejorar día a día como desarrollador.

Muchas gracias por la oportunidad laboral y por la oportunidad de aprender cosas nuevas 😊.