COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

Endless Runner

Github Link:
https://github.com/202209979/EndlessRunnerRepository.git

Álvaro Fernández Gutiérrez
Pablo Bilbao Renuncio
Grupo A

Paradigmas y técnicas de programación

Índice

Introducción

El presente documento describe el desarrollo de un videojuego del género endless runner, llevado a cabo en el motor de desarrollo Unity. Este tipo de juego se caracteriza por la generación infinita de un entorno en el que el jugador controla un personaje que debe evitar obstáculos, recolectar objetos y alcanzar la mayor puntuación posible antes de que ocurra un fallo. El proyecto tiene como objetivo principal la implementación de un sistema funcional y eficiente que combine mecánicas bien definidas, generación procedural de contenido y una experiencia de usuario fluida y entretenida.

Para el desarrollo se utilizó Unity como motor de juego, junto con C# como lenguaje de programación principal. Las decisiones técnicas y de diseño se centraron en garantizar la estabilidad del sistema, la reutilización del código y la experiencia del jugador. Asimismo, se emplearon herramientas complementarias como sistemas de control de versiones y bibliotecas específicas para la creación de activos artísticos y sonoros.

Desarrollo y problemas encontrados

Para comenzar el juego, hemos diseñado y modelado el personaje principal, que se desplaza por el mundo generado de manera aleatoria. Se ha utilizado el componente CharacterController y Rigidbody para gestionar la física del movimiento del personaje, tanto los saltos y deslizamientos como el cambio de carriles. En el cambio de carriles, tuvimos ciertos problemas ya que se nos cambiaba el personaje de forma muy lenta y gradual. Para solucionarlo, implementamos un método de interpolación (Math.Lerp), con el que conseguimos el movimiento rápido que se puede apreciar en el juego. Otro problema que encontramos en esta parte del proyecto es que despues de tres saltos consecutivos, nuestro personaje se caía del mundo, traspasando el suelo. Finalmente, conseguimos arreglar este problema también cambiando un parámetro de Project Settings. A continuación, se han implementado animaciones para las acciones de correr, saltar, deslizarse y morir. Esto se ha conseguido mediante el componente Animator y su correspondiente script que gestiona dichas animaciones. Por último, un reto de este apartado ha sido asegurar que el control del personaje fuera fluido y coherente. Para ello, hemos ajustado los valores de la velocidad y la gravedad de manera que la jugabilidad sea intuitiva y agradable. Se pueden ajustar estos parámetros en el inspector a gusto del jugador.

La creación del entorno ha sido una de las partes clave del proyecto, y hemos trabajado en equipo para diseñar prefabs de bloques que se generan de manera infinita conforme el jugador avanza. Estos bloques sirven como las plataformas sobre las cuales el personaje corre y se regeneran constantemente mediante un objeto Regenerator para mantener la sensación de un mundo sin fin. Esto se consigue gracias a que la colisión del Regenerator con el collider de cada uno de los bloques que el jugador va superando añade dicho bloque al final del mundo. Hay distintos tipos de prefabs, y uno de ellos es un bloque en el que hay tres trenes ocultos de los cuales se activa uno de forma aleatoria cuando el personaje entra en dicho bloque. Este tren se mueve en dirección al jugador a una velocidad determinada, y cuando ya ha superado al jugador, se frena y se desactiva, para no aparecer como un obstáculo estático al regenerar el bloque. En esta sección del proyecto nos enfrentamos a dos problemas: el primero es el que acabamos de mencionar, y es que no éramos capaces de desactivar el tren, pero finalmente fuimos capaces de solventarlo; el segundo es que los bloques que contienen trenes se generan en nuestro mundo algo desplazados a la derecha, y este es un problema que a día de hoy no hemos sido capaces de resolver todavía, a pesar de las horas dedicadas a ello.
El sistema de generación del mundo ha sido desarrollado para que el mundo sea infinito, pero con una progresión dinámica. Hemos diseñado un sistema de generación procedimental que crea bloques nuevos y elimina los que ya no están en la vista del

jugador. Cada bloque tiene características aleatorias, como muchos vagones, rampas u obstáculos, lo que añade variabilidad al entorno. El reto ha sido encontrar un equilibrio entre la generación aleatoria y la coherencia del mundo. Para ello, hemos establecido un Pooler que establezca reglas, permitiendo que los bloques se organicen de manera lógica, pero sin perder la variedad.

Continuando con la creación del juego, decidimos incluir diamantes y multiplicadores de puntaje como los principales ítems del juego. Los diamantes permiten al jugador sumar puntos, mientras que los multiplicadores aumentan la suma de puntuación durante un tiempo determinado. También hemos establecido un sistema de guardado de diamantes y de récord de puntuación, de tal forma que cuando el jugador sale del juego se guarda su registro por si desea volver a jugar en un futuro.

El diseño de la interfaz de usuario (UI) ha consistido en la creación de una pantalla de inicio, un menú de inicio, un menú de ajustes y una pantalla de fin de partida. Todos estos menús contienen botones que se encargan del cambio de escenas. El menú de Game Over guarda los datos de puntuación y diamantes obtenidos en la partida y da opción de empezar una nueva partida o de volver al menú. El menú de ajustes, por su parte, permite ajustar el volumen de la partida mediante un slider, así como otras funcionalidades extra.

Por último, incluimos dos efectos gráficos importantes. Por un lado, el Post Processing y, por otro, el Audio, controlado por el SoundManager. De esta forma, hemos añadido música a nuestro juego, cuyo volumen se puede regular en el menú de ajustes, y hemos podido cambiar ciertos parámetros gráficos en nuestro mundo, como la saturación, el contraste o el color de las formas.

Diagramas UML

Hace unos meses hicimos un UML que nos sirviera para planificar como iba a ser el desarrollo del proyecto. Obviamente han surgido muchos cambios del que iniciamos desarrollando frente al UML final. En este apartado se van a comentar las principales diferencias, y cuáles han sido los causantes de estos cambios, veremos como algunos fueron por desconocimiento nuestro, frente a otros que han sido cambios necesarios en el producto final.

Scene Layer

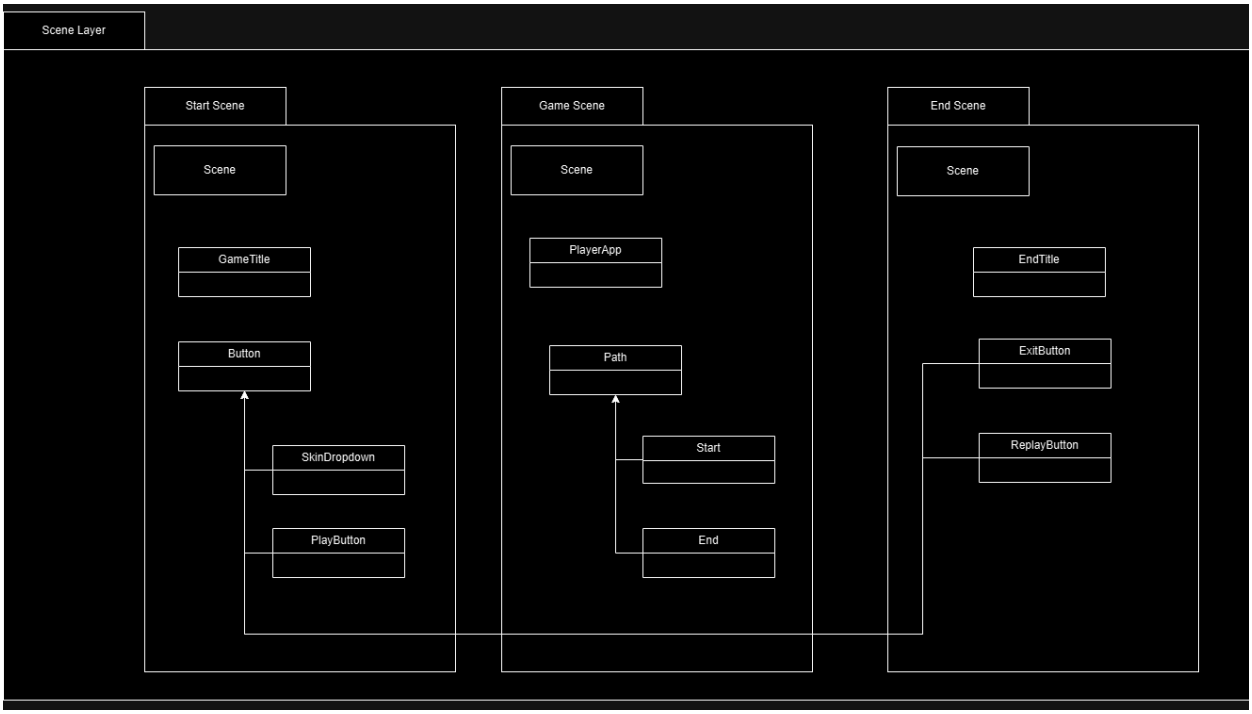

Figura 1. Scene Layer inicial

Este fue nuestro planteamiento inicial para el manejo de las escenas, en el que pretendíamos tener una pantalla de inicio, la del juego con el loop principal y la de Game Over. Esta fue la sección que menos cambios sufrió, ya que se mantuvo prácticamente igual, a excepción de un menú adicional añadido, en el cual se muestra tanto la mejor puntuación, como los diamantes acumulados. Los diamantes no tienen ninguna implementación actualmente, pero abriría la posibilidad de aumentar el alcance del proyecto dándoles cierto uso.

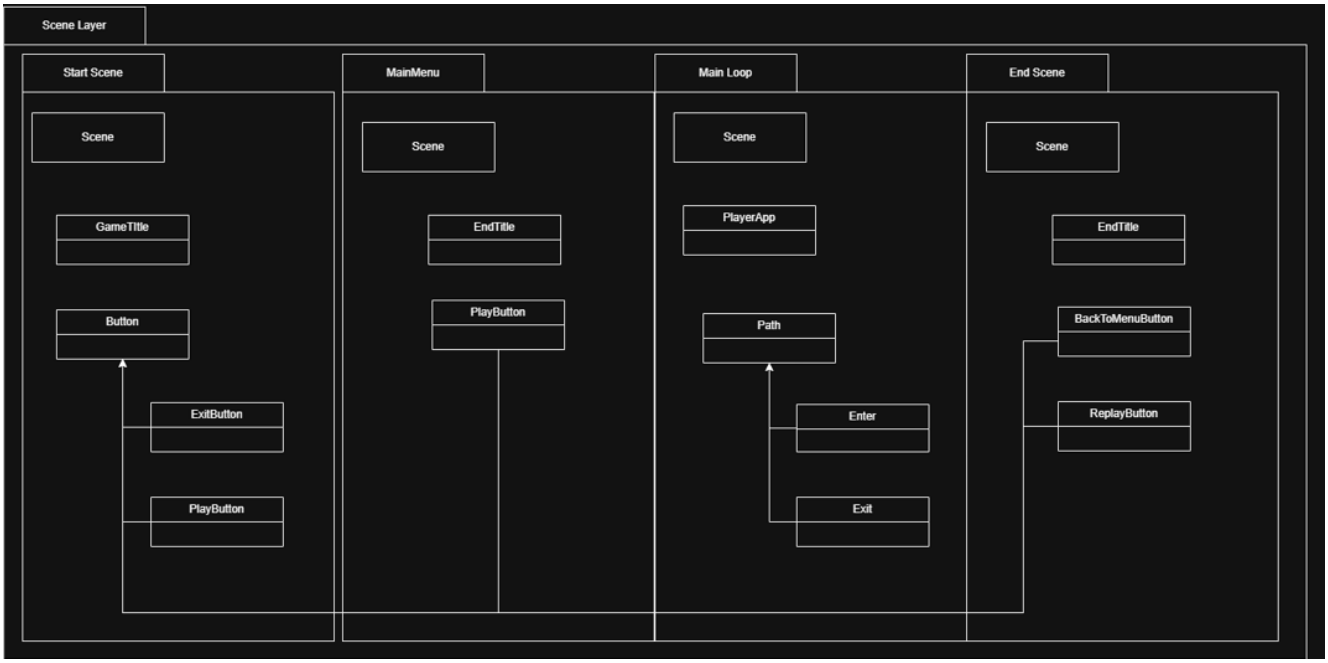

Figura 2. Scene Layer final

Manager Layer

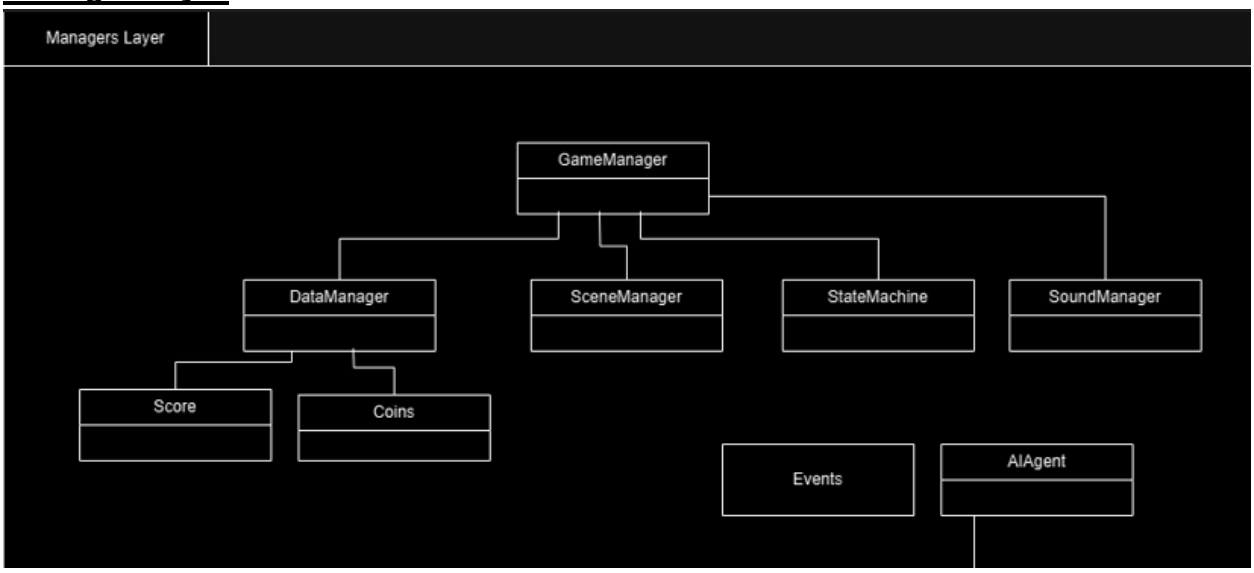

Figura 3. Manager Layer final

Esta sección es la que más cambios sufrió. No éramos conscientes de la cantidad de información que se tenía que manejar en un videojuego, y de la cantidad de distintos managers que hemos tenido que crear, para que ninguno de ellos tuviera varias tareas. Los eventos sí que se han mantenido, ya que el proyecto los utiliza constantemente, tanto para el UI como para la generación de bloques. En esta sección se han creado muchos Singleton, lo cual nos ha facilitado considerablemente el acceso a cada uno de estos managers, evitando que pudieran tener acciones contraproducentes los unos con

los otros. También se ha tenido que crear una maquina de estados y un fichero para el movimiento de la cámara, ya que si no causaba muchos problemas visuales y con los cambios de escena. Las animaciones se han manejado con el agente del jugador, permitiendo que siempre se corresponda tanto el collider con el movimiento pedido y con el gesto correspondiente. En la figura 4 podemos ver el resultado final, que, a pesar de las modificaciones, sigue teniendo aspectos en común con el producto inicial.

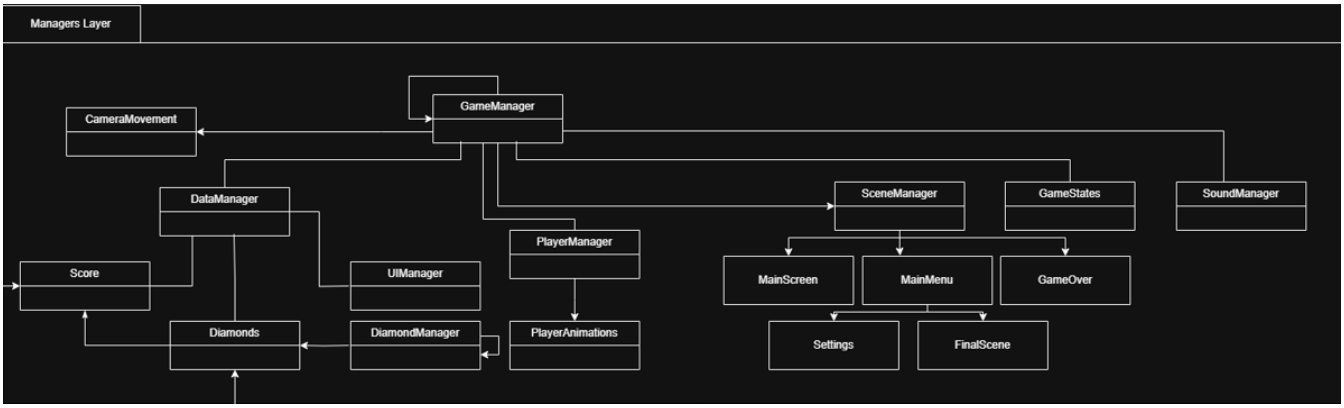

Figura 4. Manager Layer final

Circuit Layer

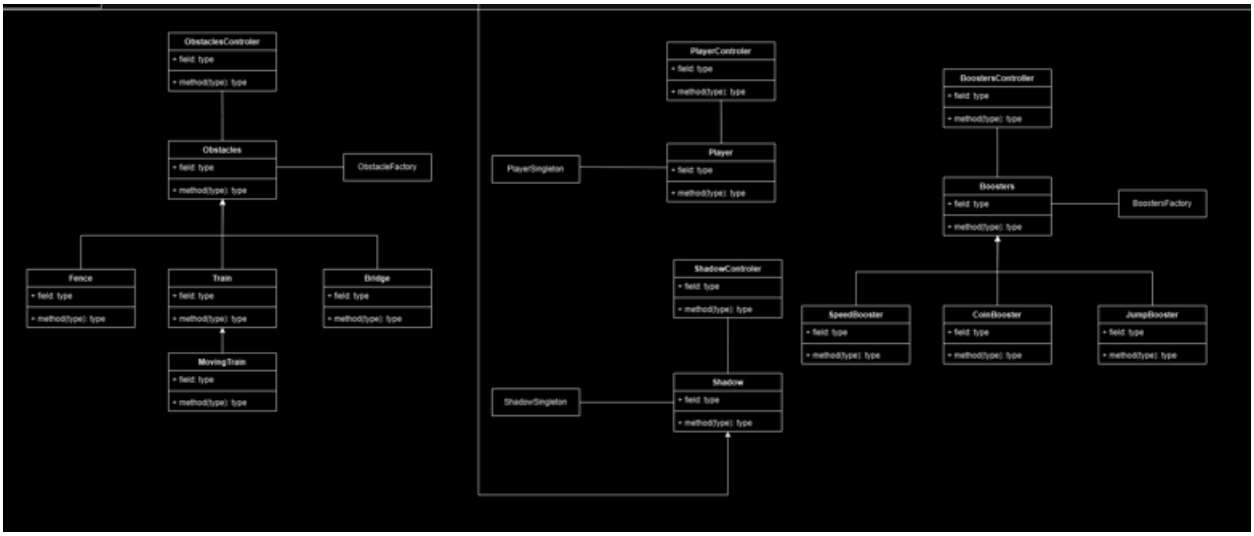

Figura 5. Circuit Layer inicial

Esta sección también ha sufrido cambios, pero en parte ha sido por un cambio en la idea del juego. Inicialmente, la idea era una carrera en contra de otro personaje con ciertos niveles, en el que el que cometiera menos errores acabaría ganando. No obstante, cambiamos la idea de proyecto a un endless runner, en el que el circuito se iría generando a medida que el jugador va avanzando, dados una serie de prefabs, y una serie de condiciones almacenadas en el pooler, de forma que se evite la consecución de ciertos bloques que llevarían a la muerte instantánea del jugador.

Además se utilizó un evento RequestNewBlock, que según estas condiciones irá generando los nuevos bloques junto con los diamantes y los power ups correspondientes siempre lo suficientemente avanzado como para que el jugador no tenga problemas de alcance. El regenerator persigue al jugador muy por detrás de él, y va eliminando los bloques para permitir la creación de más.

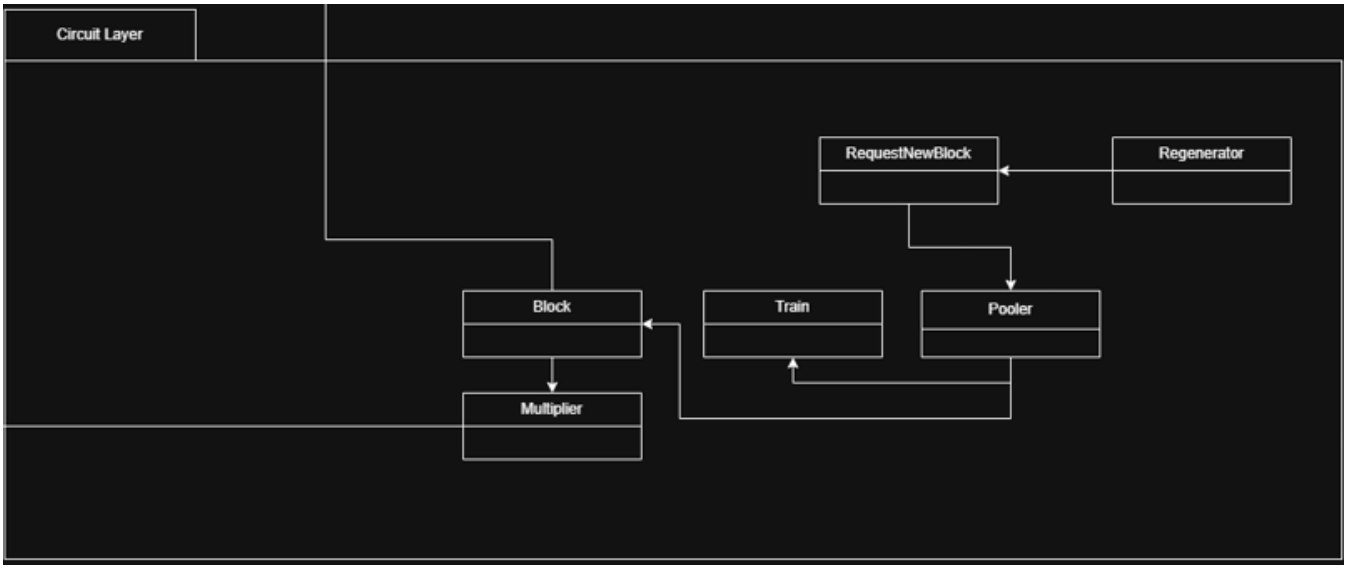

Figura 6. Circuit Layer final