2019

IES San Vicente

César Martín Sogorb

[SISARBROS]

SisarBros es un juego de plataforma inspirado en los juegos clásicos de plataformas de los 90

Índice

1. Introducción

Este proyecto se basa en el desarrollo de un juego plataformas 2D para cualquier dispositivo, he utilizado el motor gráfico de juegos Unity 3D, realización lógica del juego en el lenguaje C#.

La creación de este juego se basa en que a mí personalmente son los juegos más divertidos a los que he jugado ya que tienen lo ideal para un juego que son: jugabilidad y sencillez. Este juego está basado en un juego clásico de las décadas de los 80's y 90's.

El juego va a consistir en realizar un juego de plataformas, con sus niveles cada vez más difíciles, según los objetos que vaya consiguiendo el personaje tendrá unas habilidades u otras, podrá obtener monedas durante la partida que se encontrarán en sitios secretos y con estas monedas luego en el menú podrás conseguir logros y objetos especiales. Y poseerá habilidades especiales a la hora de que vayan consiguiendo logros, etc.

El personaje si choca con algún personaje enemigo morirá y se empezará la partida desde el principio o por el check-point que estará en el centro exacto del mapa.

El juego antes de empezar tendrá la posibilidad de elegir el personaje con el que se va a jugar la partida.

Para realizar el proyecto correctamente se han establecido una serie de objetivos que son los siguientes:

- Crear la lógica del personaje con sus movimientos, daño, muerte, animaciones.
- Crear pantallas con sus diferentes escenarios.
- Crear enemigos con diferentes lógicas de movimiento para cada uno.
- Crear un coleccionable para que sume puntuación y desaparezca cuando se coja.
- Crear un menú donde se tendrán diferentes opciones a elegir: iniciar partida, créditos y salir.
- Crear pantalla de muerte, donde se mostrará la puntuación obtenida y el posible regreso al menú principal.

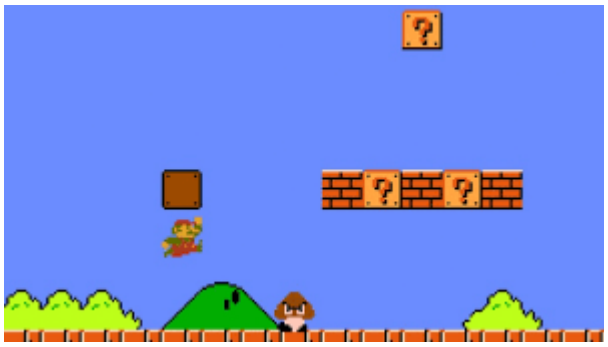
- Crear pantalla final para cuando se finalice la pantalla, donde se mostrará la puntuación obtenida y el posible regreso al menú principal.

A lo largo de esta memoria veremos antecedentes de juegos reales en los que me he basado para la realización de este proyecto, se explicará el análisis del proyecto, se visualizará el diseño del proyecto explicando cada fase, se expondrán las conclusiones obtenidas al haber realizado el proyecto y se aportará la bibliografía que se ha utilizado para el desarrollo del proyecto.

2. Antecedentes

Super Mario Bros

Principalmente me he basado sobre este juego porque para mí lo tiene todo, y es un juego al que le he dedicado mucho tiempo. En mi opinión es el juego perfecto, ya que lo tiene todo, jugabilidad, sencillez y música pegadiza.



Super Sonic

Otro de los grandes juegos de plataformas 2D que se basa en coger monedas y desplazarse hacia la derecha, misma jugabilidad y sencillez que Mario Bros. En mi opinión estos juegos son los que más he dedicado horas de juego.

Existen muchos más pero estos son los que más me gustan y he jugado, una gran jugablidad y sencillez que los hacen grandes juegos para todos los públicos.

3. Análisis

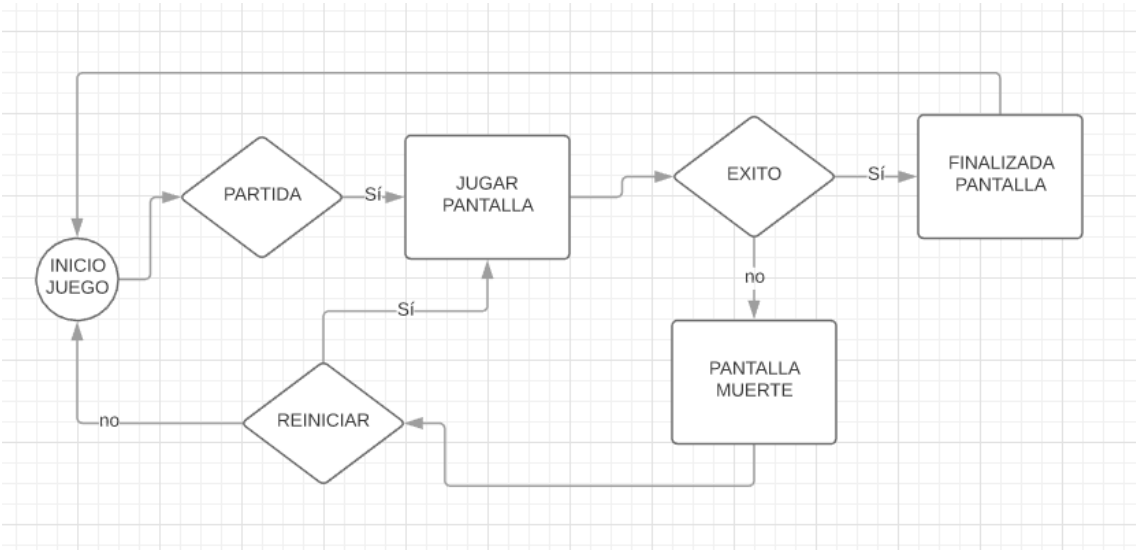
El objetivo es crear un juego de plataformas clásico, en el que la jugabilidad y la sencillez son los elementos básicos en este juego para garantizar diversión y entretenimiento a cualquier tipo de persona que le gusten los videojuegos.

El objetivo principal del juego es avanzar hacia la derecha y pasar los niveles, ya sea eliminando los enemigos o evitándolos, esquivando los objetos del mapa y almacenando los distintos coleccionables que habrá por el mapa.

El juego al ser de plataformas podría alargarse demasiado, y para evitar eso hemos creado un muro que avanzará detrás del personaje para que no se haga eterna la partida y sea más dinámica.

No se podrá guardar la partida, pero todos los datos almacenados antes de empezar la partida, quedarán en una base de datos donde se podrán mostrar en la pantalla de menú.

También se han añadido mejoras al juego, como por ejemplo aumentar la dificultad en cada escenario y según vaya avanzando, aumentando el número de enemigos y haciéndolos más resistentes y con diferentes movimientos.



4. Diseño

Para el desarrollo de este proyecto se ha elegido Unity 3D como motor de juegos, aún a pesar de usarse sobre todo para el desarrollo de juegos en 3D ofrece todo lo necesario para juegos en 2D, incluyendo opciones de desarrollo específicas para este tipo de juegos.

Los scripts que realizaremos para este proyecto serán elaborados en el lenguaje C#.

4.1 Diseño de Escenarios

Para el diseño de las pantallas o “Scenes”, Unity ofrece la posibilidad de usar uno o varios “TileMaps” dentro de cada “Scene”.

Para poder usar estos componentes se hace necesario disponer de uno o varios TileSet.

Un TileSet es un conjunto de imágenes que casa perfectas entre ellas con una relación de aspecto cuadrada, de forma que si las ponemos una a continuación de otra forman una imagen conjunta.

Esto nos permite crear un “TilePallete” y crear nuestra pantalla simplemente dibujando con el cursor el cuadrado seleccionado, a través de esta tecnología se hace más sencillo la creación de pantallas, ya que nos permite plasmar la pantalla realizada en la diseñada con facilidad.

Elementos dinámicos:

Dentro de las pantllas se incluirán elementos dinámicos que no pueden ser plasmados a través de los TileMaps, para ello se crearán GameObjects con un script de lógica para darles una cierta usabilidad según se crea conveniente para el juego.

Cámara:

La configuración de la cámara será la misma para todas las escenas, contendrá diferentes elementos asociados a ellas para que se muevan con ella.

Ítems:

En todas las pantallas se añadirán ítems que el jugador puede coger para sumar puntuaciones y obtener diferentes mejoras de habilidades para superar los distintos obstáculos del mapa.

4.2 Personaje Principal.

El personaje principal será un dinosaurio, todas las imágenes que se utilizarán para el desarrollo del juego han sido descargadas de la página web www.gameart2d.com, en el

apartado de la zona gratuita, en ella encontraremos diferentes caracteres gratuitos y de pago, para el desarrollo del juego.

Para los movimientos de este se crearán una serie de métodos públicos a los que llamarán los botones que tiene asociado al teclado del ordenador, estos métodos lo que harán será que si se está pulsando la tecla de movimiento correcta, el personaje se moverá hacia un lado u otro.

Estas variables se comprobarán en el método FixedUpdate() que ejecuta Unity de forma cíclica, realizando los movimientos necesarios para este.

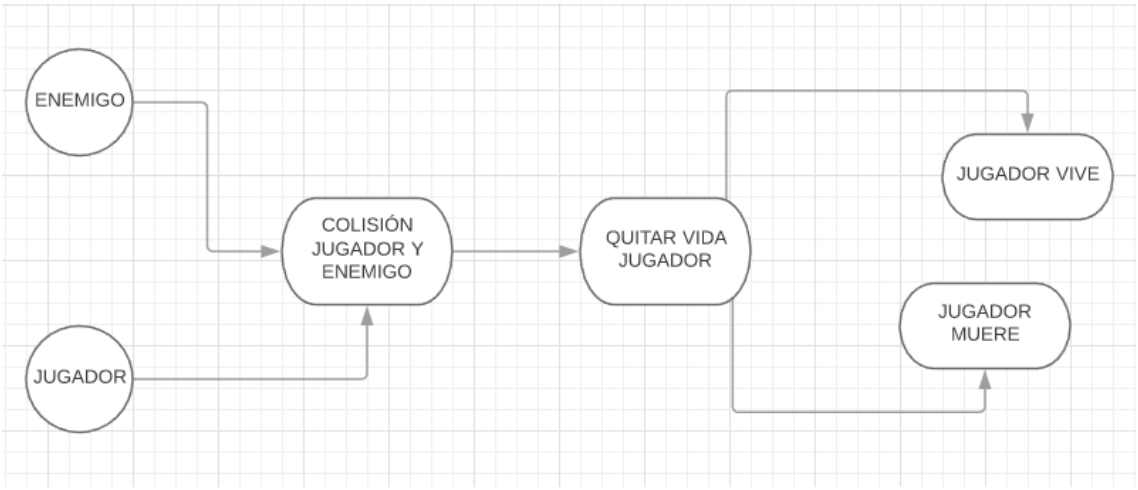
Componente Animator:

Una de las claves del dinamismo del juego son las animaciones y las transiciones entre ellas, para ello Unity nos ofrece un componente con el que gestionar ambas.

Las animaciones en un juego son una parte fundamental para adquirir una vistosidad y dinamismo altos, en un proyecto real serían competencia de los diseñadores gráficos.

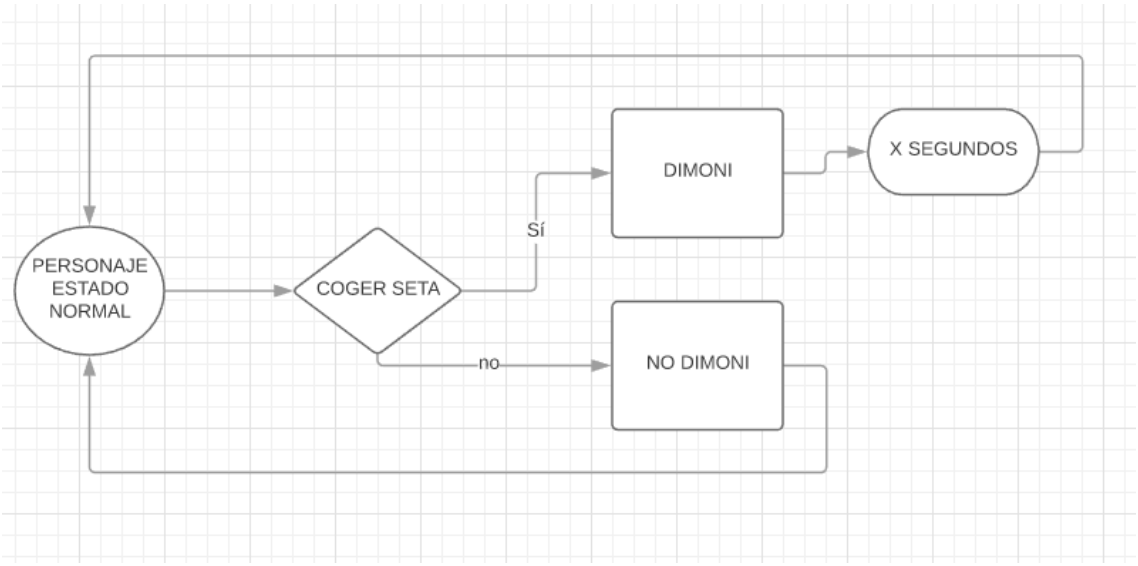
En la mayoría de los casos se pueden crear con unos pocos fotogramas del carácter que se pretende animar, formando un clip, pero esta solución fue descartada porque quedan poco reales según la investigación realizada.

En los caracteres descargables de internet suelen venir estos fotogramas para poder crear la animación, pero una opción más recomendable es realizarlas con la herramienta que ofrece Unity, que normalmente multiplica casi por 10 los fotogramas usados, ya que solo generando 3 o 4 frames clacula las posiciones intermedias del carácter y los completa generando automáticamente el resto.



4.3 Modo Dimoni

Será exactamente igual que el personaje ya que es el mismo personaje solo que se convierte si consigue las setas.



4.4 Enemigos.

Al igual que el personaje principal, todos los enemigos que se usarán para las diferentes pantallas provienen de la web www.gameart2d.com, el desarrollo de estos es parecido al del jugador, con la diferencia de han de realizar movimientos por si mismos.

He diseñado una serie de movimientos para los enemigos, para conseguir el efecto de un juego clásico de plataformas, estos siguen un patrón fijo de movimiento. Se ha diseñado una estructura de variables y métodos que implementan todos los enemigos del juego para manejar sus habilidades.

Los enemigos según de que tipo sean tendrán unas características u otras:

-Se mueven hacia una dirección y te empujan.
-Se mueven hacia una dirección y te quitan vida.
-Saltan cada X segundos y si te golpean te quitan vida.
-Corren hacia ti y si te tocan te quitan vida.

Para los distintos mapas existen diferentes tipos de enemigos:

4.3.1 Láser:

Es el principal enemigo que tenemos en el juego que nos va persiguiendo y va cogiendo velocidad en cada nivel, se podría decir que es la "gracia" del juego para que tenga tensión y estés activo jugando.

4.3.2 Bosque:

Para el primer escenario se ha elegido la seta como enemigo, lo que hará es empujarte hacia un lado para evitar que avances con normalidad.

Con la creación de un rigidbody lo que se logrará es que este enemigo tenga velocidad y masa a la hora de empujar a nuestro personaje principal.

4.3.3 Desierto:

Para el segundo escenario se ha elegido el pistolero como enemigo, lo que hará es saltar hacia arriba cada 3 segundos, si te golpea mueres, pero puedes esquivarlo cada vez que salta este enemigo.

Con la creación de un rigidbody lo que se logrará es que este enemigo tenga fuerza de impulso para lograr el salto, masa y gravedad a la hora de lograr el salto que esto hara que dependa del nivel se conseguirá el salto más prolongado o no.

4.3.4 Nieve:

Para el tercer escenario se ha elegido la calabaza andante como enemigo, lo que hará es correr hacia ti para atacarte pero este solo con tocarte te matará, gracias al Box Collider establecido.

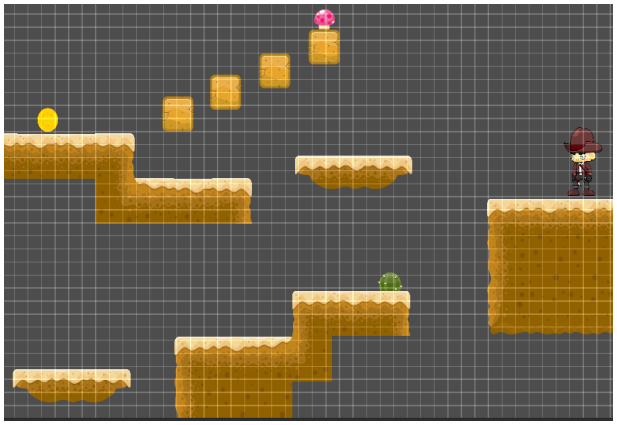
Con la creación de un rigidbody lo que se logrará es que este enemigo tenga velocidad y masa a la hora de correr hacia nuestro personaje principal.

5. Resultados

5.1 Escenarios

En este proyecto se han realizado varios “TileMaps” por pantalla para así poder configurarlos de forma diferente en función de su contenido.

Ejemplos de los diferentes elementos para el escenario y una vez implementados en el juego quedarían como en la imagen de la derecha.



Elementos dinámicos:

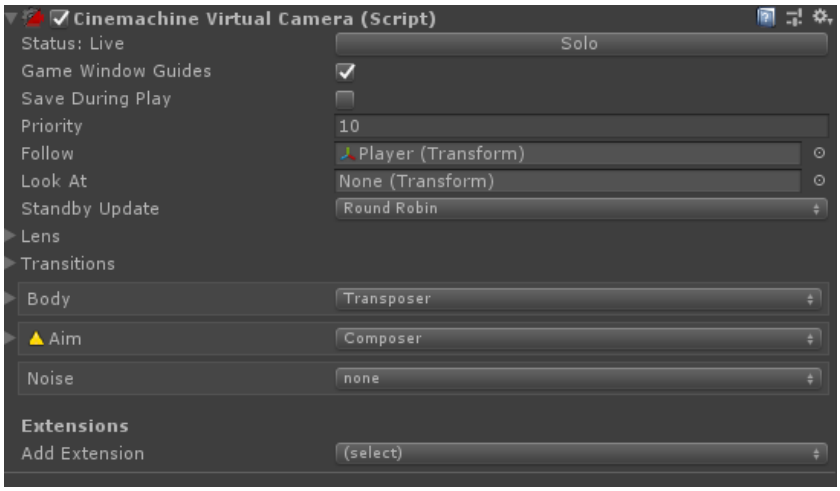
Se han creado diferentes elementos para según qué escenario y se han desarrollado los scripts configurables para adaptarlos a las necesidades de uso.

KillZone y EndZone, que son dos collider invisibles que lo que hacen es cuando tocas el killzone (la zona de abajo del mapa, cuando caes) te matará y te mandará al menú principal y la endzone es cuando terminas de pasarte el juego y se registra tu score y tu nombre en un fichero .txt para mostrarlo luego en la pantalla de Scores.

Cámara:

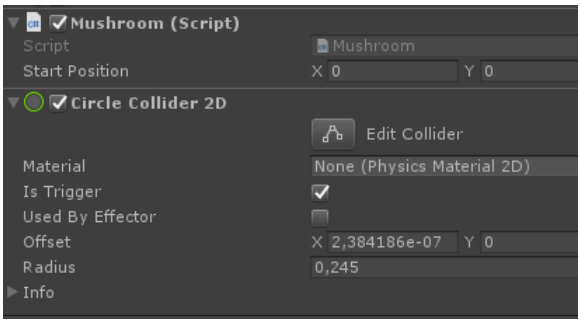
Los elementos que se han añadido a la cámara son:

- Cinemachine Virtual camera, que lo que hace es seguir al personaje principal que hemos seleccionado para que se tenga en vista en todo momento al personaje en el centro de la pantalla.



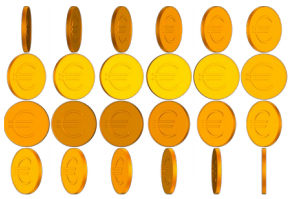
Ítems:

Se han creado varios scripts según el ítem que se coja para dar unas cierta habilidades u otras, las cuales son las siguientes para todos son parecidos los collider:



-Seta: Que cuando se recoge hace que el personaje prinicpal obtenga más velocidad durante X segundos y se convierta en “Dimoni” con la siguiente función.

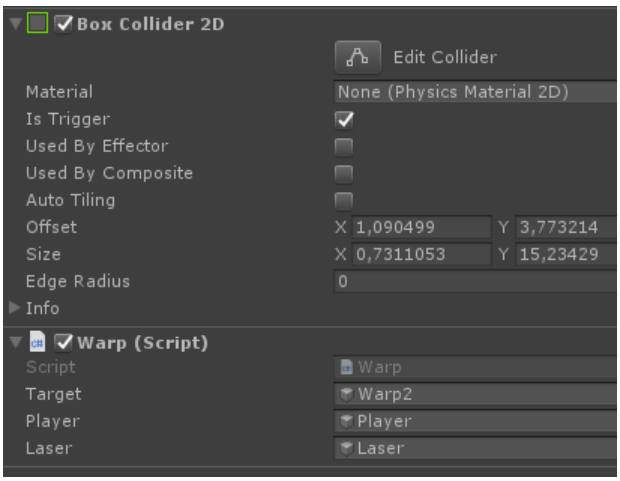
-Coin: Que cuando se obtiene suma 1 al contador de monedas, estas se van acumulando a lo largo de la partida.



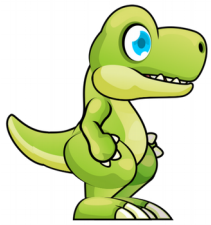
-Cactus: Que cuando se recoge hace que el personaje principal obtenga más salto durante X segundos para lograr a sitios no alcanzables.

-Muñeco de nieve: Que hace lo mismo que el cactus, ya que son los llamados "Powerups"

-Warp: Que cuando se pasa por el teletransporta al personaje principal al siguiente nivel. Este es invisible por lo que no se ve, y en el script se especifican los dos gameobject hacia donde se mueve el personaje principal y lo que se mueve que seria nuestro personaje principal llamado Player.



5.2 Personaje Principal.



Este será el sprite de nuestro personaje principal.

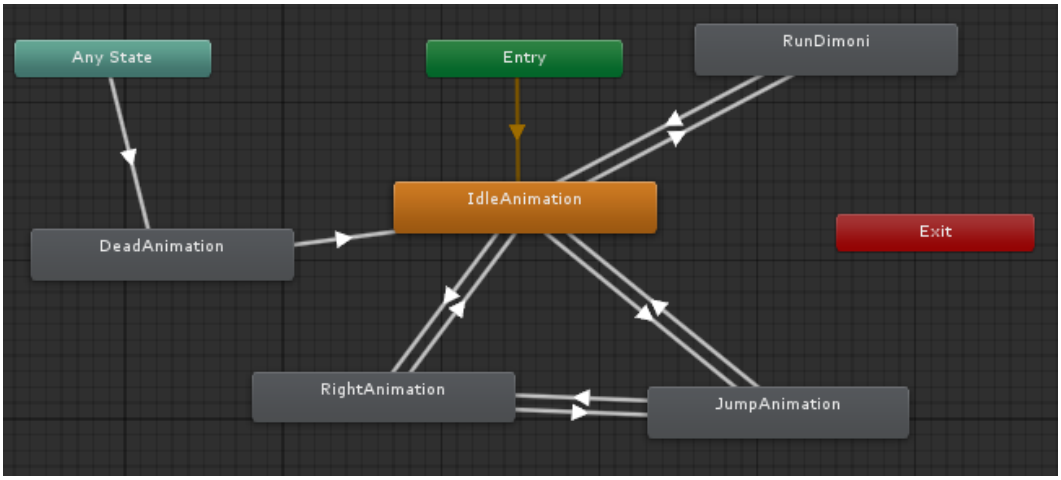
A continuación se detalla las opciones de configuración del script asociado al GameObject del personaje principal.
Movimiento: Aquí se definen las variables de movimiento horizontal y vertical así como la vida del personaje y los componentes que hacen posible saber si el personaje está tocando el suelo.

Se ha añadido al personaje principal un componente RigidBody, para el movimiento en el eje X me decantado por la propiedad velocity del RigidBody que a mi juicio proporciona un movimiento más natural teniendo cierta inercia al dejar de pulsar los botones de movimiento. Para realizar el salto se utiliza la función AddForce del mismo componente, de esta manera la subida del personaje viene definida por la fuerza aplicada y la caída se realiza por gravedad gracias a la librería Physics2D que obtiene unos muy buenos resultados en el comportamiento de los objetos.

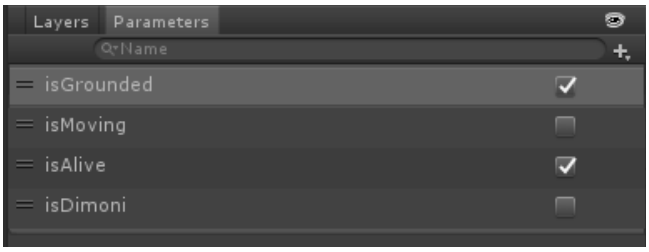
Componente Animator:

Unity tiene la capacidad de reconocer los archivos arrastrados a la vez al juego de manera que generá un archivo.anim y genera un Prefab con todas las animaciones y el controller necesario para gestionarlas.

Para manejar las transiciones de las animaciones se han creado las variables necesarias en el script y se han asignado al componente animator del personaje, para que funciones hay que crear las relaciones entre estas en el controller.



Controller del personaje principal



Variables booleanas de las transiciones según el estado del personaje principal

5.3 Modo Dimoni

En este apartado nos basamos en el personaje principal ya que tiene todas las mismas características, pero cuando el personaje principal obtiene el ítem seta, nuestro personaje entra en este modo durante unos segundos y obtiene más velocidad y cambia de color:

De manera que en la seta hemos establecido que el componente animator sea verdadero y de esta manera obtenemos al personaje como hemos visto en la anterior imagen.

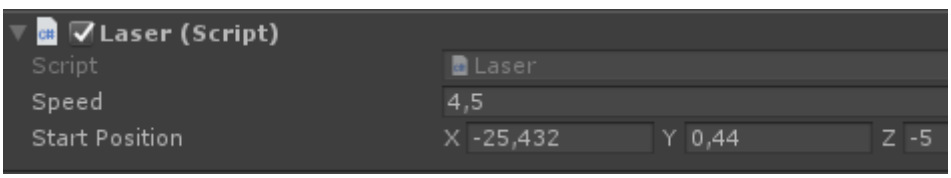
5.4 Enemigos

Al igual que ocurre con el jugador principal los enemigos disponen de un componente Animator que se encarga de las transiciones entre animaciones, además en el caso de estos se han añadido eventos a las animaciones que actúan como triggers haciendo más sencilla la muerte de estos.

Todos los enemigos detectan la posición del jugador en la escena y se activan cuando están dentro del rango que configuramos, gracias a los collider y de esta manera al tocar con nuestro personaje principal lograrán que lo maten.

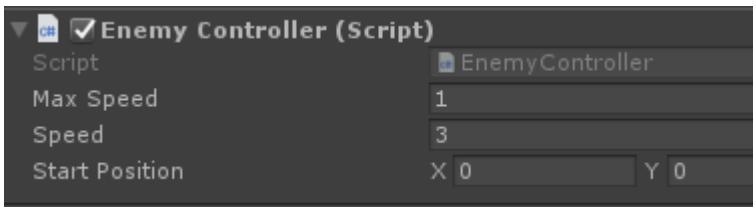
5.4.1 Láser

Este es el enemigo principal del juego que posee un collider que cuando colisiona con el personaje principal te mata y te va persiguiendo durante la partida.



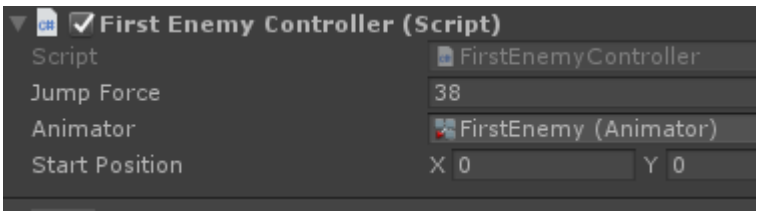
5.4.2 Bosque

Este enemigo lo que hace es empujarte, tiene ridigdbody y un collider para poder colisionar con nuestro personaje principal, y posee una cierta velocidad para empujarte hacia nuestro Laser.



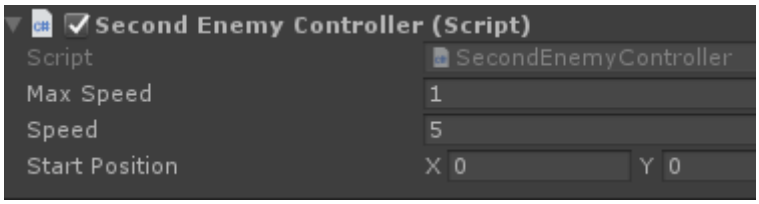
5.4.3 Desierto

Este enemigo lo que hace es detectar cuando esta en el suelo y se espera X segundos y cuando pasan los segundos necesarios lo que hace es saltar y tu pasas por debajo porque si chocas con el te matará.



5.4.4 Nieve

Este enemigo lo que hace es correr hacia el personaje principal de manera que si choca con el lo matará.



Para los llamados Warp lo que se ha querido hacer es lograr el efecto de teletransporte haciendo un delay para cuando se pase por uno y nos lleve al otro.

Precisamente debido al delay entre la creación de objetos, se generaba un problema al ejecutar este método en el hilo principal, ya que hubiera supuesto la paralización del juego mientras dure el teletransporte, se contempló la posibilidad de crearlos teniendo en cuenta el tiempo, pero se hacía complejo manejar esto y la cantidad creada al mismo tiempo.

Para solucionar el problema en un principio se optó por la ejecución de este método en un hilo diferente, pero Unity ofrece la posibilidad de simular una ejecución multihilo sin necesidad de crearlos realmente a través de las "Corutines". El uso de una Corutine es muy sencillo, solo debemos hacer que el método que queramos convertir en esta devuelva un IEnumerator, y dentro de este añadir:

```
StartCoroutine(Teleport());
```

```
IEnumerator Teleport()
{
    yield return new WaitForSeconds(1);
```

donde 1 es el tiempo en segundos que queremos que pase entre una ejecución y otra. Lo que lograremos que al pasar por el Warp al pasar 1 segundo nos teletransportará al segundo y así sucesivamente por los niveles del juego.

5.5 Pantalla Scores

Se ha elaborado en un apartado del programa un script que lo que hace es guardar los scores de todos los jugadores en un fichero .txt de manera que quedarán todos registrados cuando se pasen el juego con sus respectivos records.

Tambien se ha elaborado un script que guarda en un fichero .txt los logs de los usuarios que van accediendo al juego con su nombre de equipo y hora del momento en el que se accede al juego.

6. Conclusiones

6.1 Descripción del desarrollo del proyecto

Para este proyecto se ha alcanzado el tiempo previsto alrededor de unas 20 horas, una vez casi finalizado, gran parte del desarrollo del proyecto ha sido para la parte del diseño gráfico del juego, ya fuese buscando por internet sprites gratuitos y crear los mapas del juego.

Conseguir un movimiento fluido en todo el juego ha sido un pequeño problema que se ha solventado lo mejor posible para la jugabilidad del juego.

6.2 Conclusiones trabajo realizado

El juego resultante se hace entretenido pero de corta duración, para prolongarlo se hacen necesarias más pantallas y escenarios, invirtiendo en recursos para poder crear más escenarios diferentes se podría lograr un juego bastante más largo.

La creación de caracteres propios para el desarrollo del juego se hace inviable en un proyecto de tan pocas horas de duración, e incluso la creación de las animaciones de los caracteres es un proceso que requiere invertir mucho tiempo para un resultado óptimo, por ello conseguir Sprites con animaciones ya creadas reduce significativamente el tiempo de desarrollo.

La mayoría de objetos utilizados en el juego se obtiene mejor resultado creando scripts que admitan todo tipo de configuraciones que después podemos adaptar a cada objeto, esto es muy interesante porque se pueden reutilizar estas configuraciones.

Hemos tenido dificultades a la hora de usar StreamReader ya que daba errores y no sabía solucionarlos, no se si por problema de usar StreamReader con Unity o por qué, por lo que se ha usado ReadAllLines.

En el proyecto se ha llegado hasta un punto que no estaría nada mal, pero me hubiera gustado haber tenido más tiempo para realizar más personajes y más enemigos en el juego.

6.3 Trabajos futuros

Como trabajos futuros, se podrían añadir más pantallas a los escenarios y enemigos junto a las pantallas y la posiblidad de añadir un personaje a elegir antes de empezar el juego.

Me hubiera gustado añadir más pantallas y logros cada vez que se fuese alcanzando un nivel y determinados objetos.

7. Bibliografía

Para el desarrollo del juego se ha utilizado los siguientes recursos de apoyo:

-Guía oficial de Unity, tanto como foros y videos:

https://docs.unity3d.com/ScriptReference/

https://forum.unity.com/

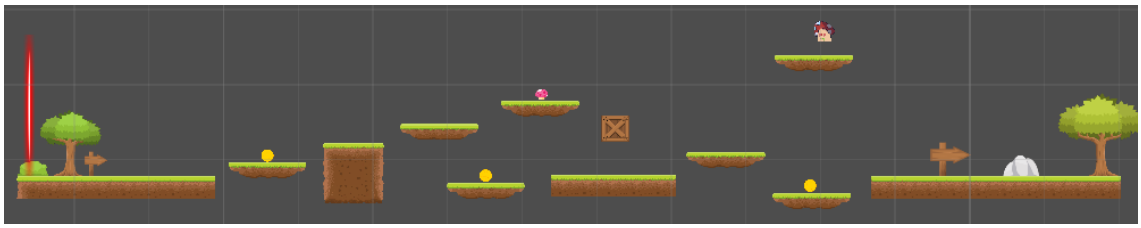
https://youtube.com/ Ya sea todo tipo de video relacionado con unity para visualizar alguna cosa concreta

- Material de apoyo proporcionado en la asignatura de Programación multimedia y dispositivos móviles de 2º Curso de DAM.

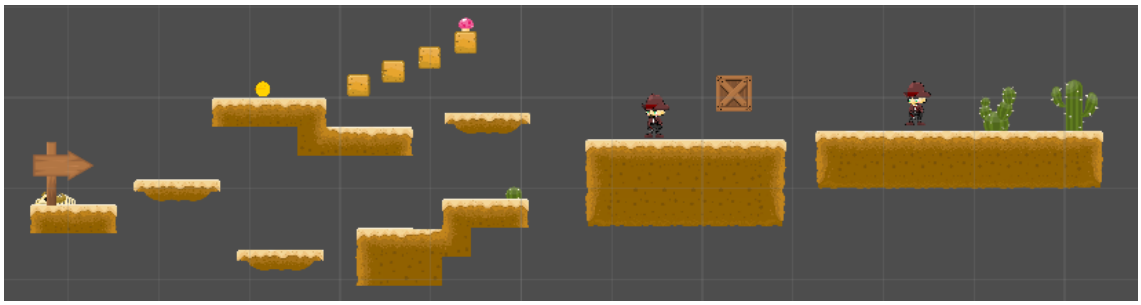
8. Apéndice

Pantallas del juego:

Bosque:



Desierto:



Nieve: