Cursed Island

Mónica Almaraz Lozano
Grado Multimedia
Videojuegos

Albert Sánchez Amo
Joan Arnedo Moreno

Junio 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Cursed Island*
Nombre del autor:	*Mónica Almaraz Lozano*
Nombre del consultor/a:	*Albert Sánchez Amo*
Nombre del PRA:	*Joan Armedo Moreno*
Fecha de entrega (mm/aaaa):	06/2021
Titulación::	*Grado Multimedia*
Área del Trabajo Final:	*Videojuegos*
Idioma del trabajo:	*Catellano*
Palabras clave	*Plataformas, móvil, 2D*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Cursed Island es un juego de plataformas para teléfonos móviles Android creado en el entorno de desarrollo Unity. Este juego es una demo de un solo nivel, a pesar que desde un principio se quiso implementar 3 niveles pero se optó por invertir el tiempo en crear elementos más divertidos y enrriquecer el contenido de un nivel.

Para crear Cursed Island lo principal ha sido tener una idea clara de lo que se quería para poder llevar a cabo una planificación y dividir el trabajo en las 4 entregas correspondientes. Se realizó un “Roadmap” con cada fecha de entrega como tope con el trabajo que hacer y durante estos meses se han seguidos los objetivos marcados. Por otro lado, se han podido añadir nuevas funcionalidades al dedicarse a un solo nivel., por lo que, el juego pegó un cambio importante desde la primera entrega hasta la última debido a dos factores, el primero es que en la primera entrega estaba aprendiendo a usar Unity e Inkscape para los sprites, y en segundo lugar por el feedback del consultor que nos dió metodologías de creación de niveles.

Para finalizar, el grado de satisfacción es muy alto, teniendo en cuenta que se comenzó sin tener conocimiento de Unity ni de como realizar los sprites y tener practicamente 1 mes de aprendizaje ha sido un reto muy difícil. A pesar de haber querido hacer tres niveles, el resultado final ofrece un nivel divertido y jugable por cualquier usuario.

Abstract (in English, 250 words or less):

Cursed Island is a platform game for Android mobile phones created in the Unity development environment. This game is a demo of a single level, although from the beginning it was wanted to implement 3 levels but it was decided to invest the time in creating more fun elements and enriching the content of one level.

To create Cursed Island the main thing has been to have a clear idea of what you wanted to be able to carry out a planning and divide the work into the 4 corresponding deliveries. A "Roadmap" was carried out with each delivery date as the ceiling with the work to be done and during these months the set objectives have been followed. On the other hand, it has been possible to add new functionalities by dedicating itself to a single level, so the game hit an important change from the first installment to the last due to two factors, the first being that in the first installment I was learning to use Unity and Inkscape for the sprites, and secondly because of the feedback from the consultant who gave us level creation methodologies.

Finally, the grade of satisfaction is very high, taking into account that it began without having knowledge of Unity or how to make the sprites and having practically 1 month of learning has been a very difficult challenge. Despite having wanted to do three levels, the end result offers a fun and playable level for any user.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

La industria del videojuego ha crecido exponencialmente durante los últimos años, tanto, que nos podemos encontrar con diferentes tipos de juegos, con diferentes títulos, distintos gráficos, jugabilidad e incluso con públicos objetivos muy concretos. Hacer un videojuego no es fácil porque la competencia es muy alta, los usuarios finales son muy meticulosos con estos productos y lleva muchísimo tiempo desarrollarlos. Se podría decir que es difícil marcar la diferencia.

Los años 90 y principios de los 2000 eran décadas que contenían videojuegos que pasarían a la historia como pueden ser *Zelda: Ocarina of time* (incluso su secuela *Zelda: Majora's mask*) o *Mario Bros* con sus juego en la SNES. Son nombres a día de hoy muy relevantes que es difícil que alguien no los conozca u oído nunca. Eran décadas distintas, porque los videojuegos no eran productos tan demandados como ahora y la gente valoraba más su contenido y lo que ofrecían.

Entonces, ¿qué puede llevar a crear un proyecto tan complicado como es un videojuego? El querer aprender a desarrollar videojuegos, entender como funcionan e intentar conseguir un producto que al final, aun que no sea un *Mario Bros*, si consiga el efecto de diversión que podía aportar antes.

Aunque se quiera la diversión que se podía aportar antes hay que adaptarse a los tiempos de ahora, por lo que el videojuego se ha desarrollado para dispositivos Android, de esa forma más gente puede jugarlo.

La finalidad del juego es entretener, sobretodo al ser un juego para móvil donde se usará en muchas ocasiones como por ejemplo cuando estés en el autobús esperando a llegar tu destino o esperando a un amigo. El juego pretende rellenar los tiempos muertos que tenga el usuario, por lo que, el juego se ha hecho para que sea fácil de usar y tratando de dar una dificultad equilibrada sin que sea muy sencillo pero tampoco frustrante.

1.2 Objetivos del trabajo

- Crear un diseño propio y original para el videojuego.
- Tratar de hacerlo lo más parecido a la idea original que se tenía.
- Realizar un nivel con una dificultad estándar, con diferentes recursos visuales y una duración mínima de 3 minutos.
- Proporcionar diversión.
- Que funcione en dispositivos móviles.
- Aprender el uso de Unity y el proceso de desarrollo de un videojuego.

1.3 Enfoque y método seguido

Para la entrega del videojuego se ha optado por hacerlo todo nuevo. De esta forma se ha podido planificar mejor que tipo de juego realizar, se le ha dado desde un nivel visual un diseño único que es uno de los objetivos planteados, se aprende como implementar animaciones en Unity, que recursos se necesita para ello y se ha podido adaptar el juego en mejor medida para dispositivos móviles.

Si se hubiera tenido que basar en otro videojuego, no se podrían haber hecho algunos de los objetivos, ya que se buscaba un diseño original y un concepto propio. Entonces, al ser una idea clara lo que se quería hacer, se optó por esta vía.

1.4 Planificación del Trabajo

La planificación ha sufrido cambios a medida que se ha avanzado en el proyecto debido a una serie de cambios, el principal es que el juego al principio iba a disponer de 3 niveles pero al final solo tenemos 1 con más elementos añadidos. El segundo cambio ha sido debido al rediseño y físicas del personaje ya que en la primero versión de la entrega el personaje era poco “dinámico”. Para ver la diferencia se mostrará el Roadmap inicial:

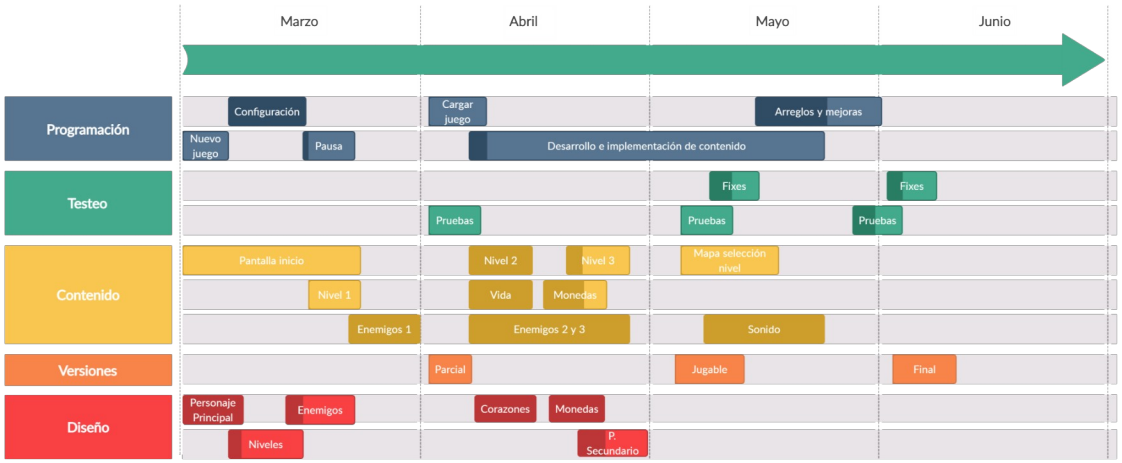

Figura 1: RoadMap PEC1

A continuación se muestra el diagrama final que se ha seguido para conseguir el resultado final:

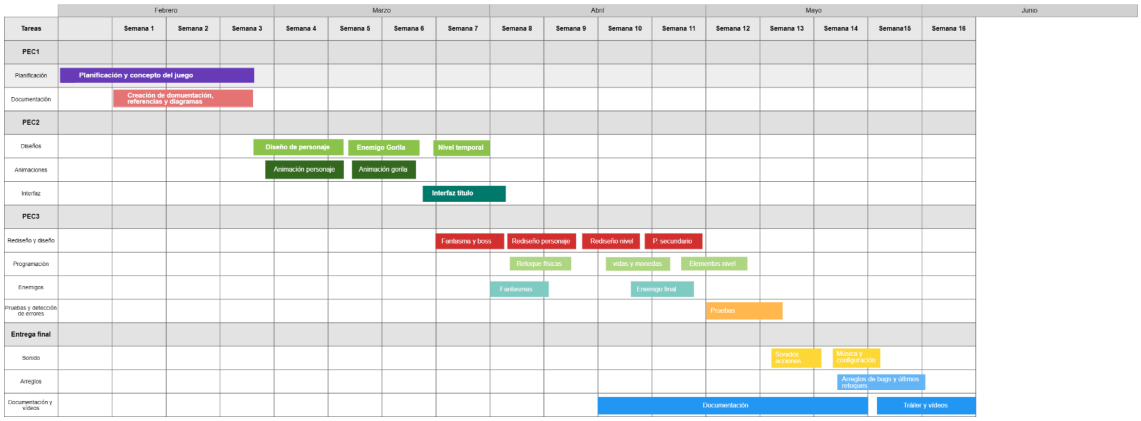

Figura 2: Diagrama de Gantt actualizado

El mayor peso del proyecto ha sido para la entrega jugable de mayo, es algo que no varía entre ambos esquemas.

1.5 Breve sumario de productos obtenidos

- Versión *Gold Master* del videojuego.
- Un tráiler.
- Vídeos mostrando las distintas fases en las que se encontraba el juego.
- Informes entregados con cada versión del juego.
- La memoria del proyecto.

1.6 Breve descripción de los otros capítulos de la memoria

En estos capítulos se hablará de los diferentes aspectos del juego a nivel de detalle para entender cómo ha sido el desarrollo de los distintos apartados. De esta forma tendremos lo siguiente:

- **Definición del juego:** en este apartado se quiere explicar bien el concepto de juego de plataformas y como ha llegado a ser lo que es hoy en día, además de una explicación de los juegos hoy en día en móviles y como se ha adaptado el concepto a dispositivos Android..

- **Diseño Artístico:** aquí se explicará y se mostrarán los bocetos utilizados para crear la parte visual del juego, tanto personajes, elementos del entorno, enemigos, etc.

- **Aspectos técnicos:** en este apartado se quiere explicar como se le ha dado vida al juego, sobretodo a un nivel técnico de como se ha creado y cómo se han hecho mecánicas usadas.

- **Diseño de nivel:** en este apartado se explicará como se realizó el diseño de nivel y los cambios que hubo respecto a la idea original.

- **Manual de usuario:** se explicará como jugar y que requerimientos necesita para su uso.

- **Pruebas de usuarios:** se expondrán las opiniones de otras personas y con los distintos teléfonos que han utilizado para comprobar el funcionamiento del mismo.

2. Definición del juego

2.1 El género de plataformas

Cursed Island se trata de un juego en el que debes superar obstáculos y enemigos corriendo, saltando o atacando hasta llegar al final del nivel. Esta descripción corresponde al género de *plataformas* que lleva existiendo desde 1980.

Este género nace a raíz de las famosas máquinas de arcade con juegos como *Donkey kong*, que innovó a la hora de añadir nuevas mecánicas como las famosas escaleras y saltar para esquivar los barriles hasta llegar a rescatar a la princesa. Si nos ponemos en contexto en esa época, marcó muchísimo la diferencia ya que los juegos que había estilo *Pacman* no tenía programados la gravedad, que será una característica que se usará en la mayoría de los juegos.

Figura 3: Donkey Kong Arcade

Aunque *Donkey Kong* fue uno de los pioneros de este género, no será hasta que llegue el famoso *Super Mario Bros* para *NES* que revolucione el género. La recolección de 100 monedas para añadir una vida o la división de mundos y distintos niveles hace que sirva como precedente para el resto de juegos.

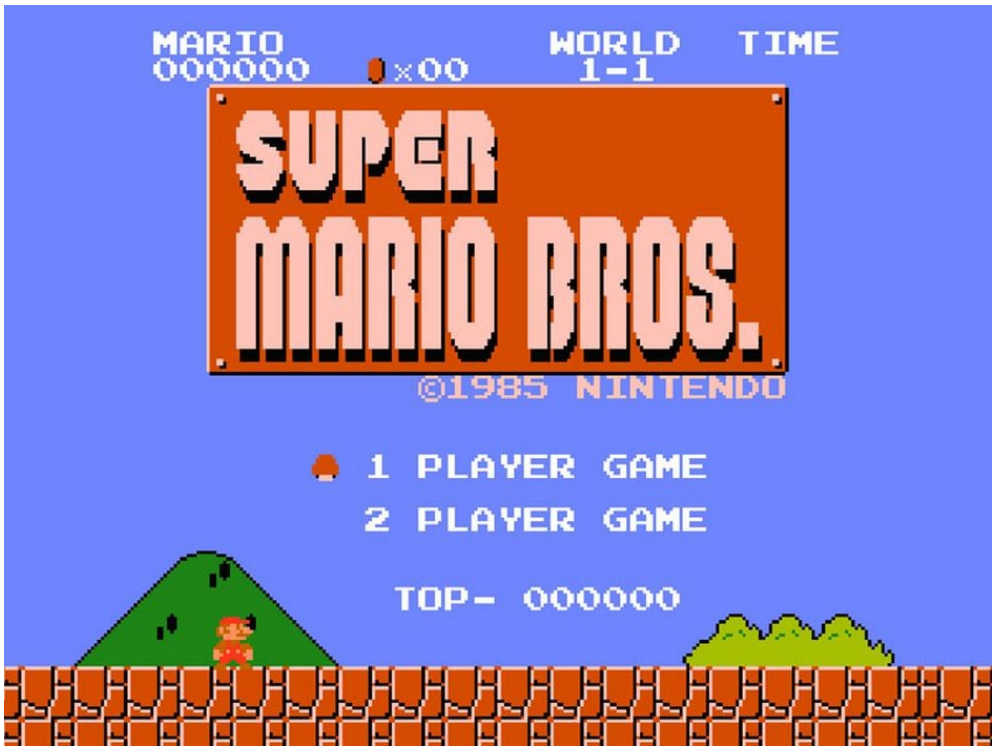

Figura 4: Super Mario Bros NES

Gracias a que la gente tenía en sus casas la *NES* se pudieron hacer juegos mucho más largos que los que había en las máquinas arcade, así nacieron la famosa saga de *Castlevania* o *Metroid*, de hecho aquí nació una subrama del género de plataformas llamada "plataformas de exploración" (aunque se conoce más como Metrovania).

Figura 5: Castlevania III Dracula's curse

Este nuevo subgénero (en aquel momento todavía no se se consideraba como tal) tenía diferencias con su predecesor, mientras que las plataformas clásicas es de ir de punto A a punto B de una forma lineal, en estos juegos invitaban más a la exploración, a conseguir objetos o habilidades antes de acceder a otros lugares, además de tener una trama más desarrollada.

Avanzando un poco más salen al mercado varios juegos añadiendo nuevas mecánicas, jugabilidad e historias como puede ser *Rayman*. Aunque me gustaría destacar el juego de *Tombi!* (conocido en Japón como *Tomba!*) el cual su primera entrega es distinta a la segunda, apostando este último por un sistema 2.5D.

Este juego muestra como ha aprendido de sus predecesores, teniendo que esquivar, matar enemigos, usar incluso objetos para acceder a otro punto del nivel, sistema de vidas... pero tiene un añadido interesante, ya que se mezcla con un toque del género del rol. En *Tombi!* Se puede subir de nivel para mejorar la vida, tiene misiones, hay unos puntos para poder abrir una cajas, etc.

Figura 6: Captura de juego de Tombi!

Tombi 2 como se ha mencionado antes usa un sistema 2.5D, las desarrolladoras apostaban por el 3D para evolucionar más el género. Irónicamente, a día de hoy, se está prefiriendo realizar juegos de plataformas en 2D. El caso de *Rayman* su primera entrega fue en 2D, la segunda se desarrolló en 3D, la tercera en *Playstation 2* fue en 3D también hasta que decidieron desarrollar *Rayman Legends* y *Rayman Origins* los cuales decidieron desarrollarlo en 2D. Esto es debido a los menores costes de desarrollo, tener un diseño más elaborados y evitar los problemas de cámara.

De hecho, continuando con la historia del juego *Rayman* y dando el salto a las plataformas móviles, *Rayman Adventures* sigue con la misma apuesta artística y con los elementos propios de un juego de plataformas, llegar a un destino usando las habilidades y recolectando “hadas”.

Figura 7: Captura de Rayman Adventures

Esta apuesta de *Rayman* al salto a móviles no es algo arbitrario al igual que la vuelta de sus gráficos al 2D. Actualmente hay tres mil millones de personas jugando en plataformas móviles, esto es debido a que la mayoría de la gente ya no dedica su tiempo a jugar, sino que juega mientras hacen otras tareas. Esto cambia la jugabilidad de los juegos que a diferencia de los mandos de consola, los móviles usan pantallas táctiles, lo que cambia la forma de interactuar con el juego.

2.2 Juegos de Plataformas en móviles

Una vez explicada la evolución de los juegos de plataformas para entender sus mecánicas y en que consisten, este apartado se enfocará en los juegos móviles, para el cual se ha desarrollado Cursed Island.

Hemos hablado de saltos, movimientos, ataques, herramientas para avanzar en el juego, etc. Pero, ¿cómo funciona todo este sistema en juegos para móviles?

Actualmente los juegos para móviles se les trata mayormente como “casual”, juegas con ellos solo un rato y luego lo dejas, con diseños minimalistas y sobretodo son gratis. Son juegos pensados para cualquier tipo de persona, independientemente si está acostumbrada a jugar a juegos o no. En su gran mayoría están hechos para matar el rato.

Al tratarse de juegos casual, se debe crear un sistema sencillo, fácil de entender y que no resulte agobiante para el jugador. Al desarrollar un juego en dispositivos táctiles, se debe pensar como hacer acciones tan sencillas como correr o saltar, como tocar la pantalla móvil para realizar estas acciones o cómo verá el jugador la interfaz.

Figura 8: Captura de Cursed Island

Para Cursed Island se han combinado mecánicas comentadas anteriormente para tratar de hacerlas divertidas y generar apego al juego. En el cual disponemos de un sistema de vidas, recolección de monedas y superar enemigos y obstáculos para llegar a la meta, aunque en este caso se ha querido hacer más fiel a los juegos clásicos en los cuales al jugador elige cómo moverse y que no fuera automático como podemos ver en otros ejemplo estilo *Super Mario bros Run.*

Los movimientos automáticos de movimiento le da más sencillez al juego pero también hace que un juego de móvil si de por si es casual, sistemas como estos hacen que un usuario se canse antes al tener poca interacción con el juego. Por ello, Cursed Island busca ser casual pero que no se desinstale una vez probado el juego (aunque de momento solo disponga de un nivel).

3. Diseño artístico

En este apartado se mostrará como se ha creado la parte visual del juego. Para todos y cada uno de los diseños se ha usado el programa *Inkscape*, por lo que todos los archivos están en .SVG y las capturas se mostrarán desde su interfaz.

3.1 Creación del personaje

Al comienzo del semestre se creó un modelo diferente al de la versión final. Pero debido al poco dinamismo que tenía se optó por crear otro.

Figura 9: Personaje inicial

Para la creación del personaje final se ha basado en un modelo gratuito que disponía *Unity Assets*:

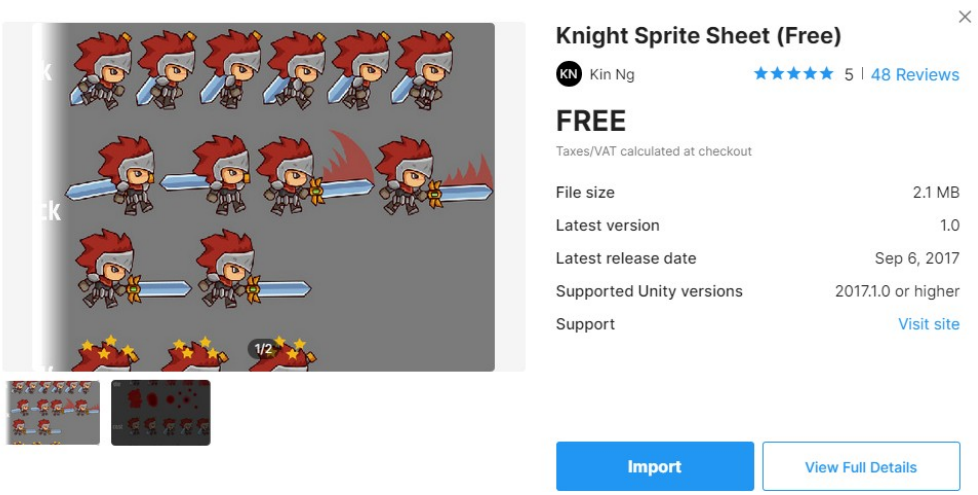

Figura 10: Captura de Knight Sprite Sheet en Unity Assets

Por lo que, a partir de este se comenzó a hacer al personaje final tratando de darle más suavidad al movimiento de este. El resultado final fue el siguiente:

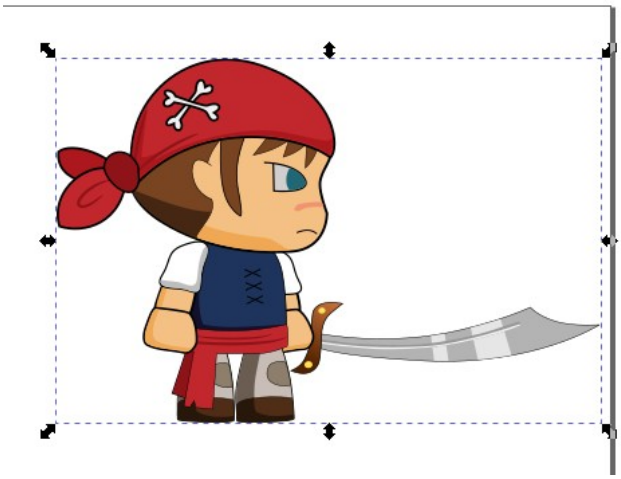

Figura 11: Captura del personaje final desde Inkscape

Una vez que se tiene la base del personaje, también desde el propio *Inkscape* se hicieron las animaciones. En total el personaje cuenta con 6 animaciones diferentes.

Animación Idle (estando quieto hace como que respira) compuesta por 3 imágenes:

Figura 12: Captura personaje con animación Idle

Animación de salto compuesta por 1 imagen:

Figura 13: Personaje saltando

Animación de ataque compuesto por 3 imágenes:

Figura 14: Personaje con animación de ataque

Animación de correr siendo esta la más larga con una secuencia de 6 imágenes:

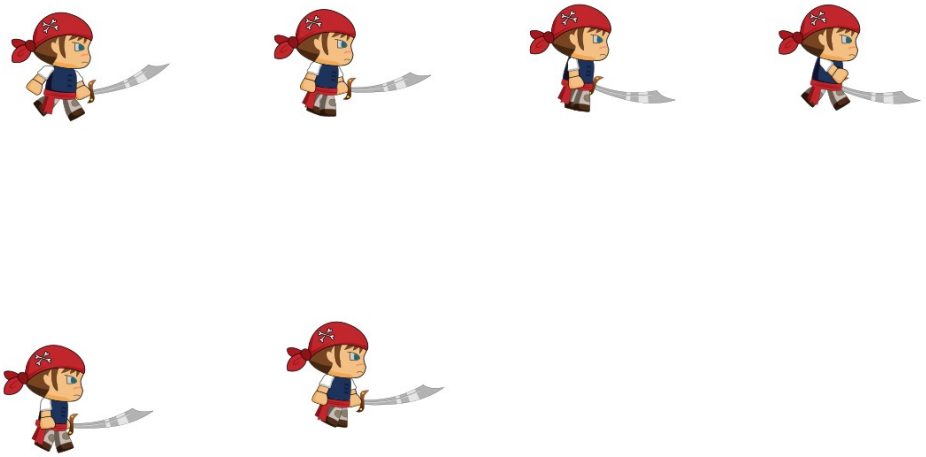

Figura 15: Personaje con animación de correr

Animación del personaje recibiendo daño con una secuencia de 4 imágenes:

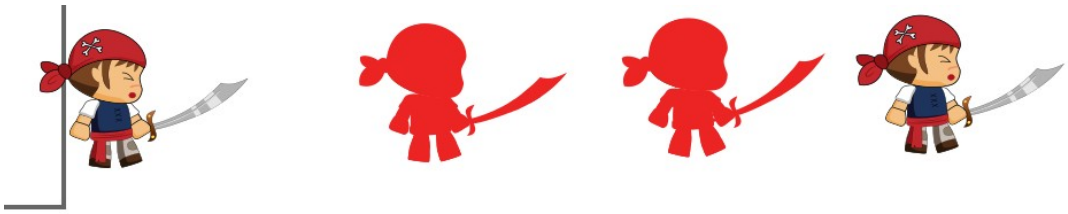

Figura 16: Personaje con animación de herido

Una vez hechas las animaciones, se exportaron las imágenes en formato .PNG de una en una a Unity donde se le dio vida al personaje.

3.2 Creación de los enemigos

El juego tiene tres tipos de enemigos, por lo que se hará una subdivisión en 3 secciones.

3.2.1 Creación de los gorilas

Este enemigo es el primero al que el jugador conoce cuando comienza el juego. Dispone de dos animaciones que es el de Idle y cuando se mueve, el efecto rojo que dispone es un efecto que posteriormente se incorpora en Unity.

Figura 17: Enemigo gorila

A continuación se pondrán las secuencias de imágenes del gorila corriendo:

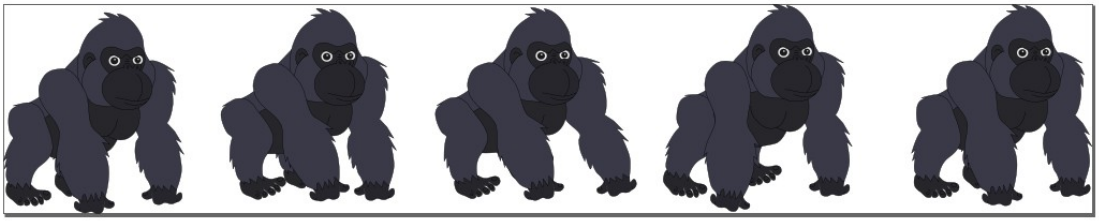

Figura 18: Movimiento del gorila

3.2.2 Creación de los fantasmas

En la segunda parte del nivel se vuelve de noche y aparecen fantasmas. Se distinguen dos tipos, el “fantasma eater” y el fantasma normal. El primer fantasmas no dispone de animaciones pero si que se le cambiar el rostro dependiendo de si nos sigue o no. El segundo tipo tiene una animación de movimiento. Estos fueron los resultados:

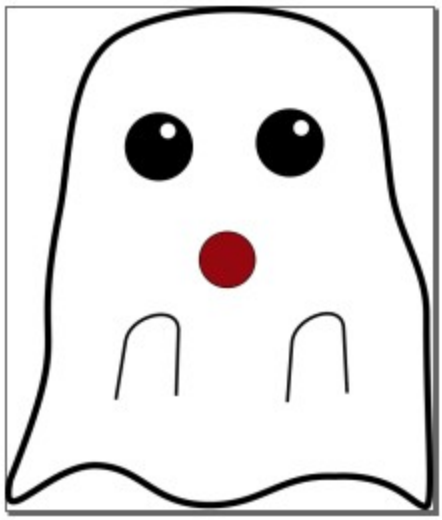

Figura 20: Fantasma en reposo

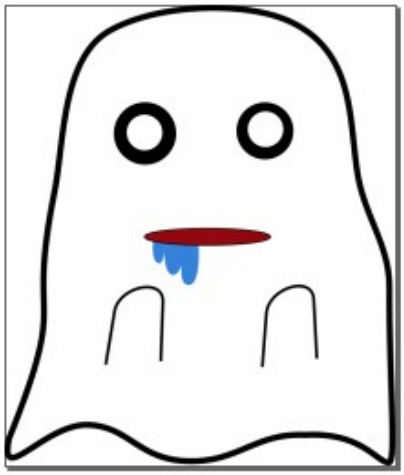

Figura 19: Fantasma al ataque

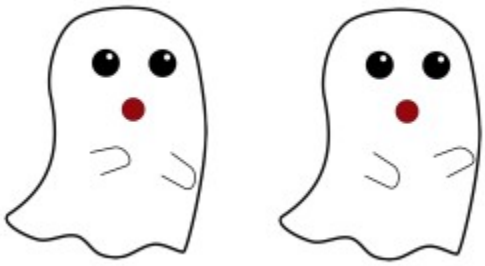

Figura 21: Animación de fantasma normal

3.2.3 Creación del jefe final

Para el jefe final se ha optado por hacer una imagen a parte para que avise al jugador de que va a lanzar una bola fuego. También dispone de por si una animación de quiero en el que el plasma de su alrededor se mueve, el cual,no deja de ser un degradado radial que se le ha añadido y movido para dar ese efecto.

Figura 22: Imágenes de jefe final

Figura 23: Bola de fuego

3.3 Elementos del juego

3.3.1 Elementos decorativos

En el nivel de Cursed Island hay muchos elementos decorativos, para sumergir al jugador en la experiencia del juego de estar como en un lugar salvaje y tropical.

Para empezar, se hablará del fondo del juego:

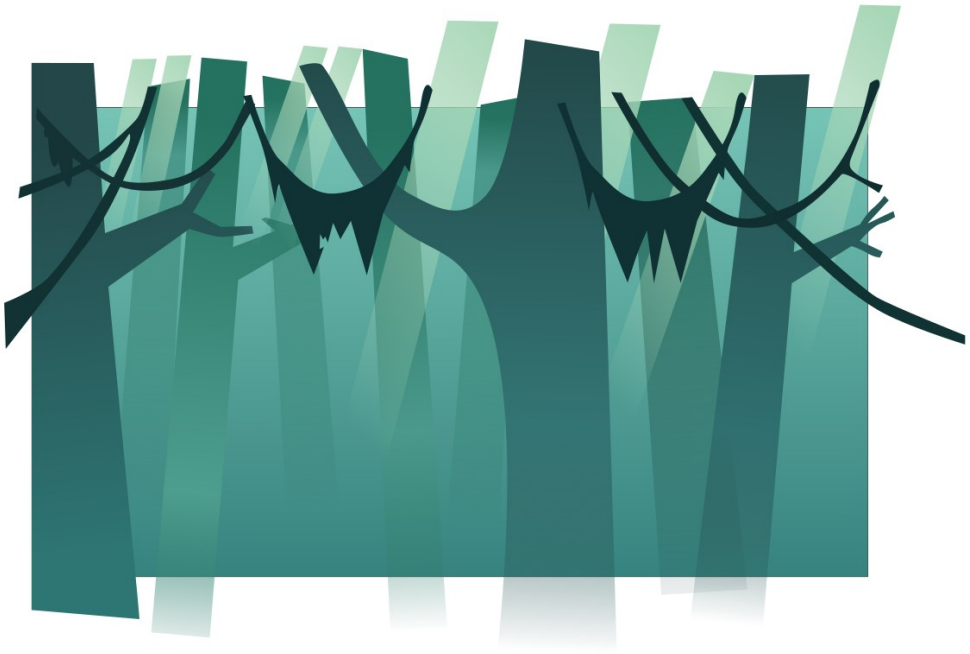

Figura 24: Fondo del juego en inkscape

Como se puede ver en la imagen en el resultado final está recortado y las lianas no forman parte del fondo. Esto es debido, a que, a la hora de exportar la imagen, se ha recortado para que todo quede al mismo nivel que la dimensión del cuadro azul de fondo y las lianas se exportaron a parte para hacer el efecto *Parallax*. El color se modifica en Unity.

Las imágenes de referencia usadas han sido las siguientes:

Figura 25: Imagen jungla de i.pinim

Figura 26: Otra imagen de jungla de i.pim

Estas imágenes dieron la idea para crear el fondo de árboles con las lianas, ya que se buscaba algo más sencillo, se usaron incluso colores que disponen estas imágenes y se les ha aplicado un degradado a los árboles.

Ahora se hablará del segundo fondo:

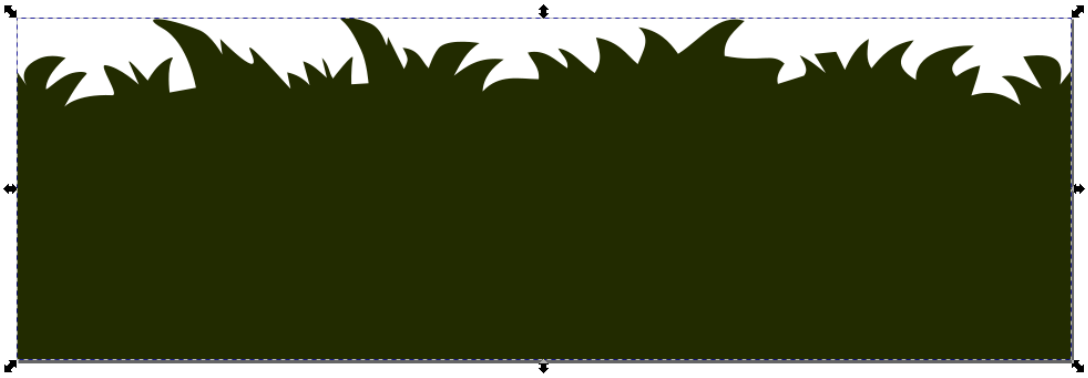

Figura 27: Arbustos como segundo fondo

Esta imagen se pondrá en el juego como una secuencia de arbustos puestos uno al lado de otro. Se hizo tan solo los trazos de las formas de los arbustos, procurando que el principio y el final coincidieran en los mismo vértices para que quedaran iguales. Se usó un tono de color correspondiente a la figura 24.

Las siguientes imágenes son ejemplos de elementos decorativos usados a lo largo del nivel para enriquecerlo y sumergir al jugador en un ambiente tropical. Entre esos elementos contamos con una piedra, una palmera, un arbusto, las lianas vistas previamente, agua, los puentes y la fogata. Cada uno de estos elementos se han hecho con trazados en *Inkscape*.

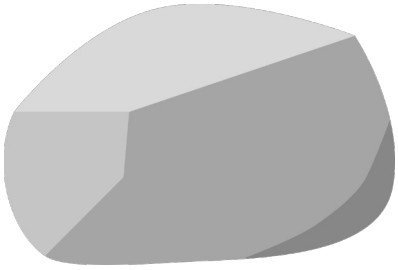

Figura 29: Roca decorativa

Figura 28: Arbusto decorativo

Figura 30: Rama para las lianas

El agua y el fuego disponen de su propia animación para tratar de darle vida y que el jugador sepa de que se tratan de estos dos elementos.

Figura 31: Agua de Cursed Island

Figura 32: Imagen fuego

Mencionar que el puente a partir de las siguientes dos imágenes se construyó en Unity, de esa forma se le puede dar profundidad y que el personaje quedase delante de uno de los soportes del puente.

Figura 33: Soporte del puente

Figura 34: Tronco del puente

Figura 35: Puente de Cursed Island

Por último, hay que hablar del suelo. Esta parte fue realizada a partir de los *tiles* de un recurso gratuito de *Bayat games*.

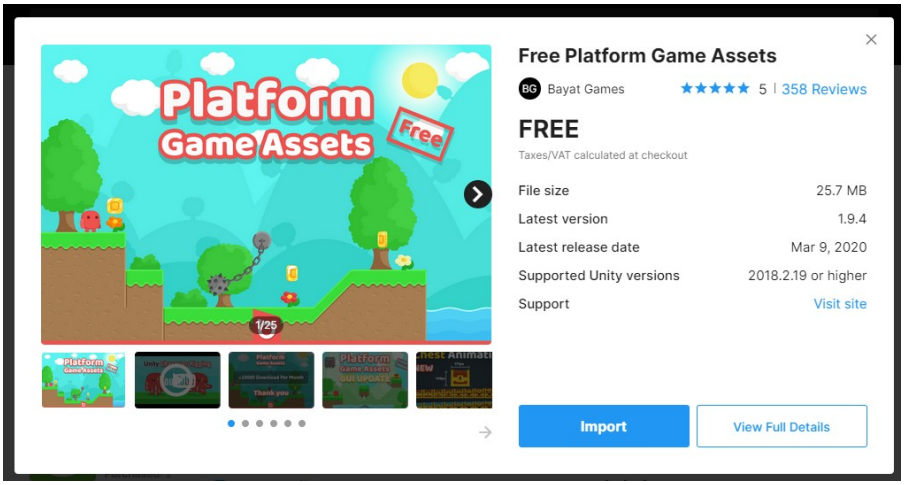

Figura 36: Imagen Free Platform Game Assets

De aquí se importó el recurso del suelo. Al descargar el .ai se hizo una serie de modificaciones para hacer un diseño más acorde con lo que se quería, añadiendo césped, cambiado los colores, etc. Después a partir de los .PNG generados se crearon los *Prefab*. El resultado final es el siguiente:

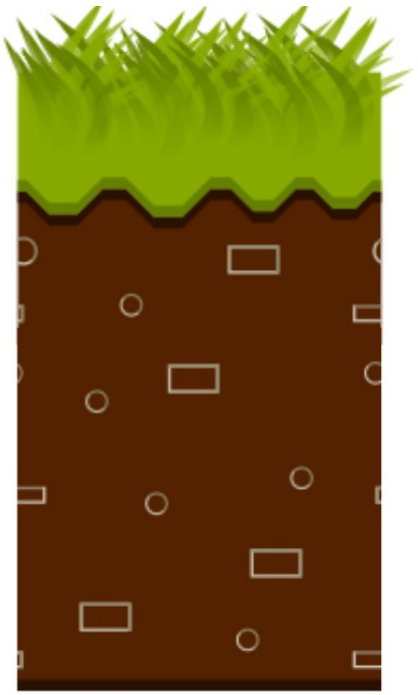

Figura 37: Ejemplo del suelo de Cursed Island

3.3.2 Elementos interactivos

En este apartado se hablará de los elementos interactivos y aquellos que tienen que ver con el jugador como las monedas y los corazones.

Comenzaremos hablando de los carteles de guardado que es una de las primeras cosas que se encuentra el jugador.

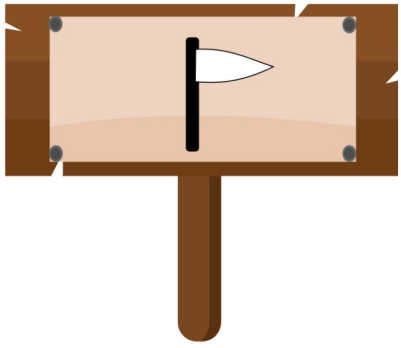

Figura 38: Cartel de guardado

Hay más carteles a lo largo del mapa pero son puramente informativos, con excepción del cartel de guardado y de prohibido el fantasmas los cuales cuando el jugador colisiona con ellos, guarda la partida o hacer desaparecer al fantasma que sigue al jugador. Se ha usado una representación de una bandera, dando a entender que se trata de un *checkpoint*.

Este elemento nos permite bajar un puente que entorpece el camino. Este diseño es único ya que para la activación de la palanca simplemente se realiza un *flip* sobre él como si se hubiera activado.

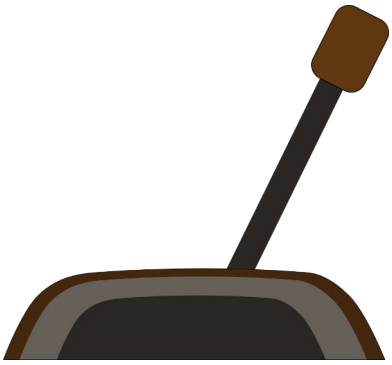

Figura 39: Palanca que activa el puente

Por último, disponemos de las monedas con su propia animación y de los corazones de dos tipos, los que tienes activados y los que le han quitado al jugador.

Figura 41: Imagen del plátano del juego

Figura 42: Imagen de la moneda del juego

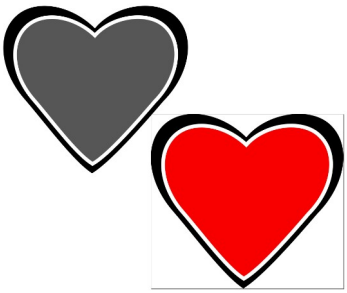

Figura 40: Corazones del juego

3.4 Interfaz

Figura 43: Captura de interfaz en Unity

Los botones de pausa, salto, atacar y las flechas se han creado en *Inkscape*, usando tonos verdes y marrones propios de una zona salvaje. Las pestañas de game over, pausa o el final del juego junto a sus botones son todos creados a través de Unity. Además, la vida del jefe final, también ha sido creado con Unity, salvo que se le ha añadido una imagen de un cuadro blanco que se le vuelve rojo y será lo que le irá restando la vida al jefe.

Figura 44: Captura del menú principal en Unity

En esta ocasión los pergaminos y el fondo de pantalla fueron descargados de una página con imágenes vectoriales gratis. Aún así se personalizó los botones para el juego y el logo.

4. Aspectos técnicos

Para comenzar con la explicación técnica, se hablará de la estructura de carpetas.

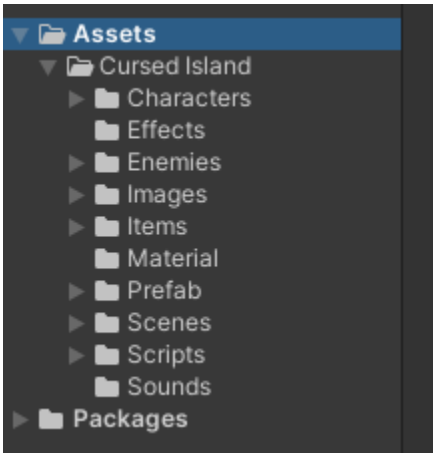

Figura 45: Carpetas del proyecto

- **Characters**: aquí están los modelos del personaje y del pirata final. También se pueden ver las animaciones de cada uno.

- **Effects**: en esta carpeta tenemos las explosiones y animaciones de muerte de los enemigos.

- **Enemies**: en enemigos tenemos al igual que los personajes, las animaciones y las imágenes vistan en el apartado de diseño de los enemigos.

- **Images**: en esta parte guardamos imágenes referentes de la interfaz. Como botones, logos, etc.

- **Items**: Imágenes y animaciones de los items con los que podemos interactuar como los plátanos o las monedas.

- **Material**: aquí se guarda el blink que hacen los enemigos al ser golpeados y el quitar la inercia de los objetos para que el personaje no se quede pegado a las paredes.

- **Prefab**: en esta carpeta guardamos los suelos creados, como plataformas, suelos grandes, esquinas, etc. Así no hace falta crear de 0 todos los puntos que se reciclen.

- **Scenes**: aquí se guardan las escenas que se usarán en el juego.
- **Scripts**: en este apartado se guardan todos los scripts usados en el juego.
- **Sounds**: por último, en esta carpeta guardamos los sonidos que tiene el juego.

4.1 Desarrollo del jugador

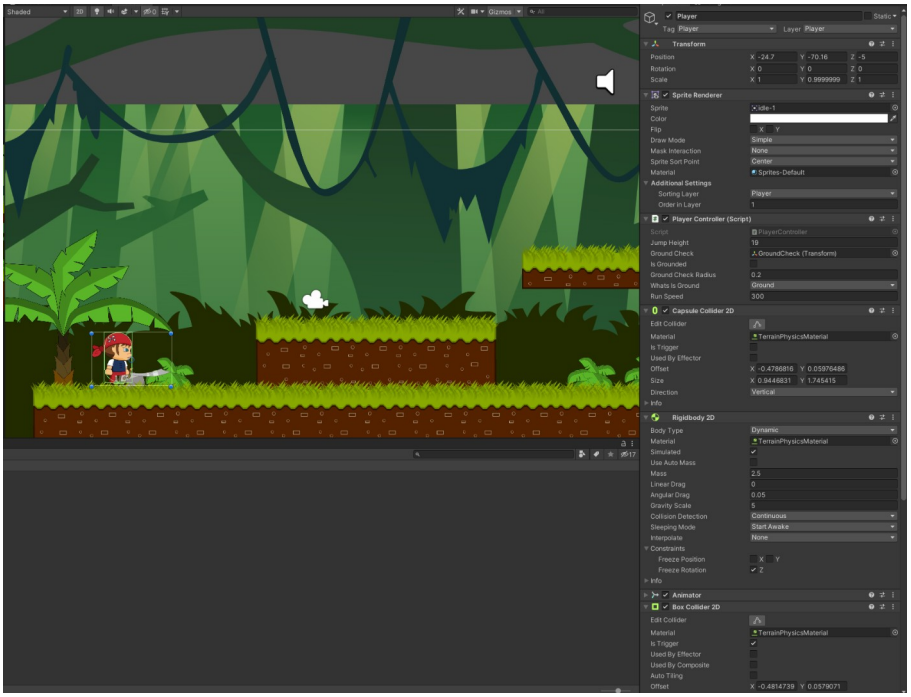

Figura 46: Captura de Unity con las propiedades del jugador

El jugador dispone de tres tipos de movimientos:

- Mover: el jugador se mueve a los lados con las flechas de la interfaz.
- Saltar: permite saltar obstáculos o enemigos.
- Atacar: usando la espada del personaje puedes atacar a los enemigos o activar mecanismos. Podría ser incluso un botón de acción.

Estos movimientos están orquestados por código que se puede encontrar en la carpeta de *scripts* → *player.* En total son tres clases las que le dan lógica para distintos momentos, desde la vida que le quitamos a los enemigos, el daño que se recibe, hacer *respawn* si te caes o incluso que botones necesitas tocar para llevar al cabo un movimiento.

Para la física del salto del jugador, se ha optado por añadir más gravedad y más masa para que las caídas sean más rápidas y no tan lentas como eran al principio, de hecho en la propia figura 44 se pueden ver los valores usados en el *rigidbody 2D*.

Los *colliders* nos ayudan a detectar las colisiones que ha sufrido el jugador, como por ejemplo detectar enemigos, saber si lo que pisamos es suelo para la lógica del salto, etc.

Por otro lado, cuando se activa la animación de atacar, existe un *collider* en el arma que se activa en el momento de hacer la animación. Este *collider* es el que hará saber a los enemigos si ha sido golpeado por el arma, ya que si la colisión es con el jugador, será a este al que le reste vida. También por ejemplo la palanca al detectar "weapon" en la colisión, esta se activará para bajar el puente.

Por último, se quiere destacar en la documentación la manera en la que se han usado los botones de la interfaz para darle movimiento al jugador. Unity de normal en sus preferencias tiene un sistema por defecto de detectar el movimiento horizontal y vertical en un teclado físico. En este caso se usan botones digitales, por lo que ese sistema de Unity no sirve. Así que para solucionar cuando sabemos si se mueve de izquierda a derecha y si tiene que activar una animación u otra se han puesto funciones distintas con *bools* que son verdaderos o falsos en función de si se ha dejado de pulsar el botón o no.

4.2 Desarrollo de los enemigos

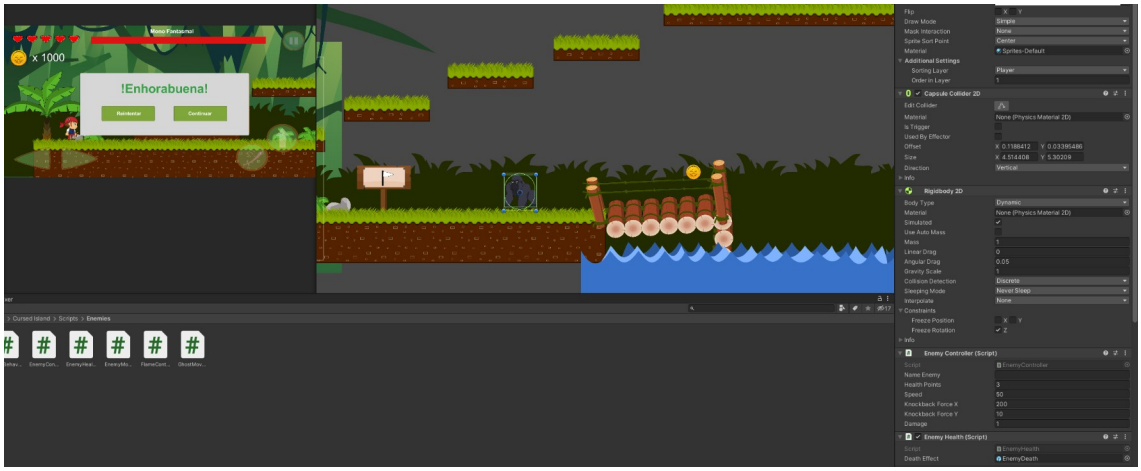

Figura 47: Captura de Unity con las propiedades de los enemigos

Hay distintos enemigos en el juego con diferentes comportamientos. Sus lógicas se encuentras en la carpeta *scripts* → *enemies* Comencemos con los 4 tipos de enemigos que hay:

- Gorilas: estos son los enemigos fáciles de derrotar para el jugador. Se aprecian dos tipos de movimientos, el cual el código lo distingue si es "patrol". El primer movimiento es el gorila moviéndose por el suelo hasta que se encuentra una pared o un sitio sin suelo y se da la vuelta para continuar. Después el segundo movimiento es el "patrol", el cual el enemigo patrulla de un punto A a un punto B.

- Fantasma normal: este fantasma tiene la misma lógica que el movimiento del gorila que patrulla, moviéndose de un punta A a un punto B. A diferencia de los gorilas estos no se pueden matar con el arma.

- Fantasma eater: Este fantasma no se puede matar al igual que los otros, solo se destruye si llegas a un cartel de "prohibido fantasma". Este a diferencia de los demás enemigos, persigue al jugador una vez que este haya entrado en su radio de detección, a partir de entonces te sigue a cualquier sitio, modificando la velocidad según si lo miras o no (como los fantasmas de *Mario Bros*).

- Mono fantasmal (jerry): este es el boss final de nivel, su movimiento se basa en teletransportase a un lugar aleatorio y lanzar bolas de fuego que hiere al jugador. Los lugares aleatorios son coordenadas que se guardan en un *array* y que con un número aleatorio elige la posición con la que cargará las coordenadas. Las bolas de fuego en el momento de lanzarse,

se dirigen a la posición en coordenadas en la que está el jugador. Este enemigo tiene más vida que los gorilas y se le puede matar con el arma.

Cada uno de los enemigos tiene una animación de una explosión una vez desaparecen, a diferencia del jefe final, que dispone de un sistema de partículas propias de Unity para cuando este es derrotado.

4.3 Elementos interactivos

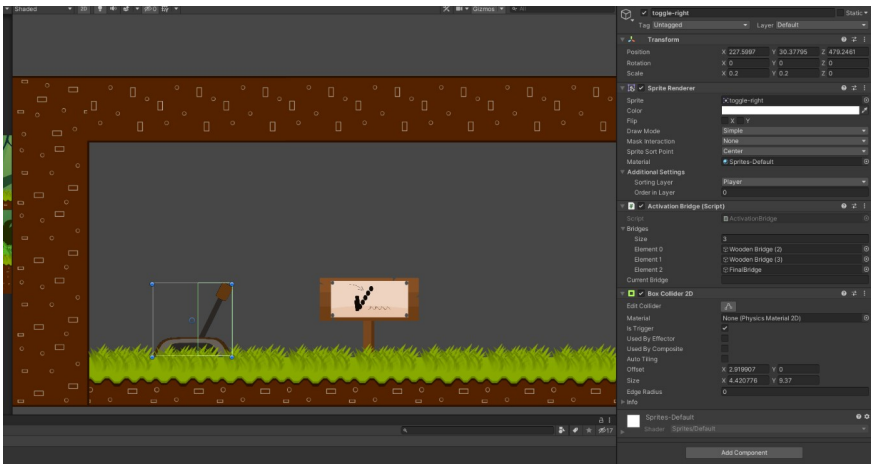

Figura 48: Captura de interfaz de Unity mostrando una palanca

El juego dispone de diversos elementos, los cuales el jugador puede interactuar. Entre ellos están:

- Carteles de guardado: estos carteles son los que harán guardar la partida o si el jugador muere aparecerán en estos puntos, su código se pueden encontrar en el apartado *Scripts* → *Scenescripts* → *checkpoint*. Básicamente este código tiene la clase *DataManager* que se encargará de guardar la posición del personaje, el número de vidas que dispone, las monedas, si se ha bajado o no el puente, etc. Una vez se cargue partida, el *DataManager* mandará los datos guardados de los carteles.

- Monedas: son un elemento de recompensa al jugador por explorar. Se puede encontrar la lógica de las monedas en *Scripts* → *Items.* Se hará una distinción de las dos clases que hay, una es para el recuento interno y la otra para mostrar al jugador el número en la interfaz.

- Plátanos: los plátanos son elementos que nos recupera la vida, añadiendo uno al contador de vidas. Se puede ver su código en *Scripts* → *Items.*

- Palanca: esta palanca no es más que un *bool* que le permitirá saber al juego qué puente cargar, si el que está arriba o abajo. Se puede encontrar su lógica en *Scripts → Items.* Se ha creado dos versiones de este puente y en función del *bool* sabemos si se tiene que activar y desactivar uno u otro.

- Activación del boss: cuando nos acercamos a la mitad de la habitación del jefe hay un *trigger* que hace que la música del jefe suene,se bajen las columnas para que no puedas salir hasta que lo derrotes y aparezca el mono fantasmal. Se puede ver su código en *Scripts → SceneScripst → boss.*

- Diálogo: Cuando derrotas al jefe final, puedes encontrar con un *NPC* el cual inicia un diálogo contigo cuando entras en su rango. Para darle utilidad a las monedas en esta demo, se ha optado por que el *NPC* te diga un diálogo u otro si tienes las 6 monedas que hay. Se puede encontrar su código en *scripts → dialogue*.

- Bloques que se rompen: en esta demo de un nivel solo existe un bloque se rompe cuando le atacas con el arma. Simplemente en un objeto que se destruye una vez colisiona con el arma del personaje. Se puede ver su lógica en *Scripts → SceneScripst → platforms.*

- Respawn: cuando el jugador salta al vacío, aparece en un *checkpoint*, pero el juego tiene que saber que el jugador ha salido del nivel. Se ha colocado un *collider* de tipo *trigger* en toda la parte de abajo del nivel y cuando el jugador choca con ese collider, se lanza el evento de quitarle un vida y aparecer en el punto de control. Se puede ver el código en *scripts → player.*

- Día y noche: en un punto del juego se vuelve de noche. Usamos un boleean para saber si el jugador a llegado a cierto punto del juego y que color dar al fondo. Se puede ver su código en *Scripts → Scenescripts → Effects.* Como al final se ha optado por crear un solo nivel, se ha decidido mostrar al jugador de cuando se hace de noche a cuando es de día. Por la noche estarán los gorilas y por la noche están los fantasmas, algo que iba a ser 3 niveles se ha decido poner en uno. Así al nivel, se le ofrece más dinamismo.

4.3 Cámara del juego

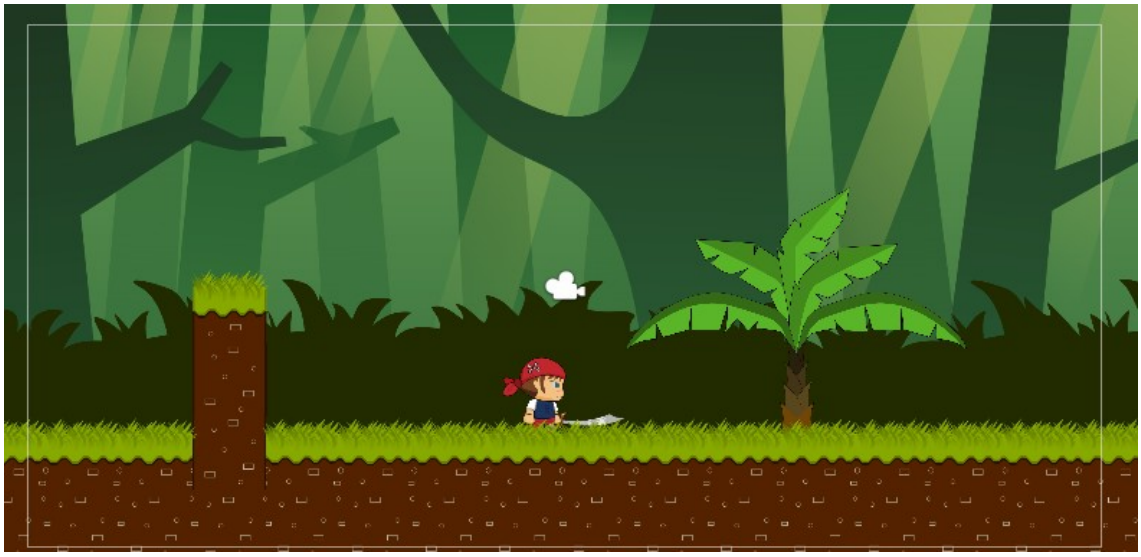

Figura 49: Captura de Unity mostrando la cámara

La cámara del juego tiene tres características importantes a destacar:

- Es una cámara que persigue al jugador por el mapa. Aunque saltemos o nos movamos hacía los lados nos perseguirá, estableciendo al jugador como su centro y añadiendo el fondo de los árboles.

- La cámara tiene un límite (el objeto *level* es que lo determina). Con esto se quiere decir que, por ejemplo, si el jugador esta al comienzo del nivel no se ve lo que hay a la izquierda, lo mismo si estamos al final del nivel que no se ve que hay a la derecha porque la cámara tiene un tope en cada una de sus dimensiones, esto de delimita en código también. Pasa lo mismo si el jugador se cae al vacío, la cámara no le sigue, permanece estática al ras del suelo.

- Dispone del efecto *parallax*, esto se trata de que los objetos del fondo se muevan más lentos que los de delante. Se puede ver más claro cuando están las lianas delante de la cámara y los arbustos que están más alejados.

El código de la cámara se puede ver en *Scripts → Scenescripts (camera controller y Parallax).*

4.4 Sonido del juego

Figura 50: Pistas de audio

El juego dispone de 22 pistas de audio. Todas ellas sacadas de páginas con sonidos libres de derecho y de uso no comercial.

Los sonidos se han dividido en dos grupos, los que son de música y los que son de efectos. De hecho, en la figura 48, se puede observar que hay dos pistas maestras, que serán las que nos clasifique los sonidos en esos grupos. Con esto, ganamos que podamos modificar la música o los efectos, por ejemplo, si queremos bajar el volumen de música y subir el de efectos en las opciones del juego.

Para incorporar los sonidos se ha usado una clase única *AudioManager* que será la que gestione los sonidos a nivel de programación. Se puede encontrar su código en *Scripts → Music.* Así con un *Singletone* (una única instancia) podemos invocar este *AudioManager* desde cualquier punto del juego.

5. Diseño del nivel

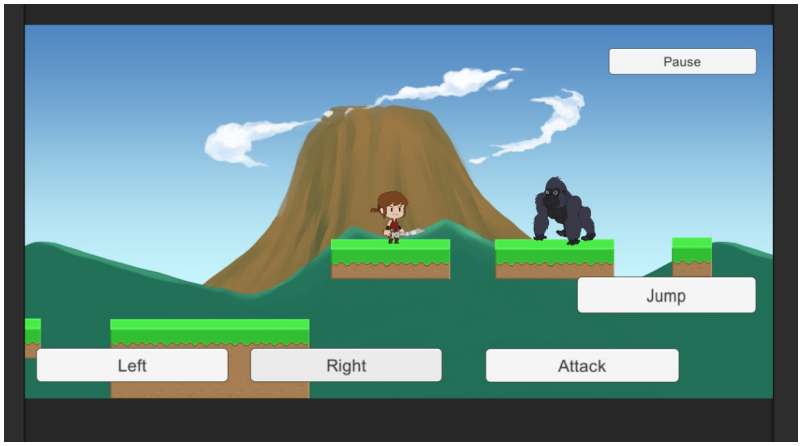

Figura 51: Nivel de la versión parcial

Al comienzo del semestre, se realizó un nivel (el nivel 1) el cual iba a ser definitivo solo añadiendo los plátanos y las monedas. Sin embargo, gracias a la reseña del profesor y de opiniones de otros usuarios , se decidió cambiar el concepto del nivel. También cabe destacar que se aprendió más de Unity.

Quitando las diferencia visuales, los problemas que tenía el nivel inicial son:

- Una distribución simple, sin dar al jugador un reto satisfactorio. Se basaba solamente en dar saltos de una plataforma a otra.

- La disposición de los enemigos no era acertada y en ocasiones podía frustrar cuando morías.

- No se sabía cuando habías llegado al final del nivel (hasta que salía la ventana emergente). Tampoco había diferencia con el principio de nivel.

- Algunas plataformas no estaban adaptadas al movimiento del personaje, generando que en ocasiones no se pudiera alcanzar alguna y obligando al jugado a explorar otras rutas.

El resultado final en un vistazo general, ha sido el siguiente:

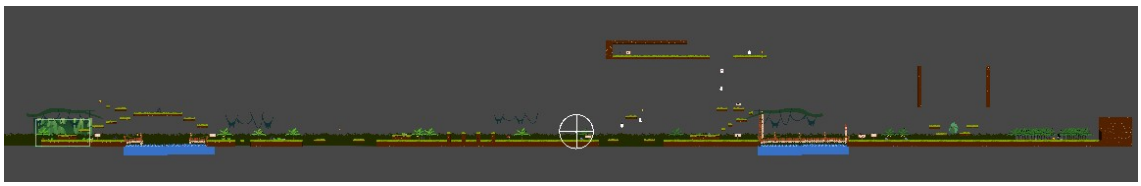

Figura 52: Nivel en la versión final

El nivel ha sido más meditado y dividido en secciones, los cuales da una jugabilidad distinta. Las secciones en las que se divide son son:

- Puente roto: de esta forma obligas al jugador a tomarla otra ruta y se familiariza con los saltos, también le muestras que con la exploración puedes conseguir monedas.

- Plataformas móviles: el jugador sabrá que habrá plataformas móviles en el nivel y que tendrá que lidiar con ello.

- Torres: se le muestra al jugador que hay elementos interactivos con el botón de atacar.

- Fantasmas: en este punto tiene que poner en práctica los saltos que ha aprendido, además de *timear* los saltos correctamente para alcanzar la moneda o la plataforma. Los fantasmas actúan de elemento que obliga al jugador a pensar un poco más.

- Palanca y puente: El jugador ahora sabe que puede interactuar con otro elemento, cuando le de a la palanca, al igual que la torre rota, sabrá que la tiene que golpear para activar.

- Jefe final: En esta parte se le ha querido dar al jugador la oportunidad de poner a prueba todo lo que ha aprendido en el nivel. Saltar o esquivar en el tiempo requerido y golpear.

- Final de nivel: Aquí se ha querido dejar claro que es el final de nivel gracias al diálogo con el *NPC*. Incluso se puede ver una cueva de “salida” que da a entender que has terminado.

Gracias a las etapas que tiene el nivel, hace que de manera subconsciente el jugador aprenda a controlar el personaje, aprender las mecánicas que le ofrece el juego y que sienta satisfacción cuando termina de jugar.

También se ha tenido en cuenta, que el juego es para móvil y como se ha comentado anteriormente se ha pretendido dar al nivel un punto equilibrado de dificultad.

6. Manual de usuario

6.1 Requisitos del juego

Los requerimientos básicos para usar el juego son los siguientes:

- **SO**: Android
- **Versión**: 4.4 (API 19)+
- **CPU**: ARMv7 with Neon Support (32-bit) or ARM64
- **Gráfica**: OpenGL ES 2.0+, OpenGL ES 3.0+, Vulkan
- **Ram**: 1GB

6.2 Uso del juego

Figura 53: Imagen de la interfaz del juego

En esta imagen nos encontramos con los siguientes botones:

- Flechas: en la parte de abajo izquierda tenemos flechas con las que nos desplazaremos lateralmente.
- Botón de pausa: arriba a la derecha disponemos del botón de pausa para pausar el juego o salir al menú principal.
- Botón de saltar: abajo a la derecha representado con una flecha tenemos el botón de saltar.
- Botón de atacar: abajo a la derecha junto al botón de saltar representado por una espada, tenemos el botón de atacar.

Junto a los botones en la parte superior izquierda tenemos corazones que representan la vida del jugador y las monedas representan el número de monedas recogidas a lo largo del nivel.

7. Pruebas de usuarios

A continuación se comenta las experiencias de los usuarios a nivel general en distintos apartados y los móviles que se han utilizado para las distintas pruebas.

Gráficos:
Gráficos coloridos que se integran perfectamente con lo que se quiere representar. Es visualmente atractivo para los usuarios y las animaciones permiten al jugador un feedback constante de lo que esta ocurriendo en el juego.

Jugabilidad:
Mecánicamente correcto, las acciones de saltar y golpear responden al instante. Problemas a la hora de esquivar los enemigos porque siempre se acaba recibiendo daño. La inclusión elementos interactivos para completar el nivel les gusta y así como secretos en los que se esconde monedas.

Rendimiento:
El juego ha rendido bien en todos los tipos de móviles que se han probado. La resolución la han visto correcta en sus correspondientes móviles.

Experiencia en general:
En general ha gustado el juego y se han divertido con él. No obstante, a los usuarios en ocasiones se han encontrado problemas a la hora de derrotar sobretodo al jefe final por un *bug* en concreto que hay.

Lista de teléfonos móviles:

1. Asus Rog Phone II
2. Huawei P20
3. Oppo reno Z
4. Samsung Galaxy S20
5. Xiaomi Redmi 9
6. Oppo find X2 Pro
7. Samsung Galaxy S10
8. OnePlus 6

8. Conclusiones

8.1 Conclusiones del trabajo

La experiencia de realizar este trabajo ha sido muy enriquecedora a pesar de los momentos de frustración debido a la falta de conocimiento del entorno.

Me gusta el mundo de los videojuegos y realizar mi propio proyecto aunque sea algo pequeño realmente es muy satisfactorio. He aprendido mucho sobre como crear un videojuego, desde el diseño, animaciones, cómo implementarlas, físicas, etc. De hecho pienso que ha sido un acierto utilizar Unity al ser una herramienta estandarizada y que mucha gente ha usado, porque en los momentos de dudas he podido encontrar fácilmente información.

Por otro lado, he tenido que utilizar en este proyecto todo mi conocimiento adquirido hasta ahora a lo largo del grado, como por ejemplo estimar los tiempos que tardaría en hacer cada parte, el diseño del juego o saber programar. Me hubiera gustado que en el grado se hubiera dado una iniciación a Unity o a otra herramienta para ahorrar tiempo de aprendizaje y de bloqueos que se podrían haber evitado con un conocimiento básico, así no hubiera hecho falta aprender deprisa y corriendo para llegar a las fechas.

Aún así, me ha gustado desarrollar mi propio proyecto, el aprender a usar Unity ha sido muy satisfactorio, incluso poner en práctica ideas que se me han ocurrido para mejorar la experiencia del juego.

8.2 Objetivos planteados y reflexión crítica

Realmente se han llegado a muchos de los objetivos plateados. Quizás no de la manera que se quería pero que de alguna manera se ha podido llevar a cabo.

Un ejemplo de ello era el querer hacer tres niveles diferentes, uno sería de día, otro de tarde y otro de noche. En cada nivel iba a haber diferentes enemigos, para resaltar las desemejanzas que habría de dificultad entre nivel y nivel siendo el de la noche más difícil. En su lugar, en el mismo nivel se ha hecho un transcurso de día y noche mostrando también de alguna manera que el nivel iba a ser distinto (como por ejemplo la aparición de los fantasmas).

Por otro lado, me hubiera gustado añadir un mapa de selección de nivel, pero como finalmente se ha realizado solo un nivel, se ha decidido prescindir de ello.

El motivo por el que no se ha realizado lo esperado ha sido realmente por la falta de tiempo. El aprender a usar Unity también ha sido algo que ha quitado tiempo y se ha tenido que deshacer todo lo que se hizo en la PEC2 para rediseñar y hacer mejores diseños y mecánicas.

8.3 Seguimiento de la planificación y metodología

Para la entrega de la PEC2 si que se siguió la planificación. Se entregó un nivel con enemigos y el personaje, pero era algo ineficiente ya que el nivel no proporcionaba diversión y el personaje tenía movimientos muy toscos. Tener que realizar otros dos niveles dejaría unos niveles simples porque no tendría tiempo para realizar nuevas mecánicas. Asique se decidió centrarse tan solo en un nivel y hacerlo más atractivo para el usuario. Por lo que, en el mes de abril no se trabajaron en otros niveles sino que los esfuerzos se centraron en realizar mecánicas interactivas para el jugador, enemigos como el fantasma que te persigue o el jefe final para que suponga un reto al jugador.

Para la entrega de junio se han añadido los sonidos y arreglado bugs, permitiendo que el tiempo hubiera estado haciendo el mapa de los niveles, lo he invertido en arreglos de bug, realizar la documentación y los vídeos.

Para el resto del proyecto se ha seguido el Roadmap, dentro de lo que cabe no he tenido que hacer un cambio añadido y he podido trabajar bien con lo planteado inicialmente. El tiempo para la PEC3 ha sido bien invertido en crear una buena experiencia de juego, aunque reconozco que debido a todos los cambios ha habido mucho trabajo para la PEC3 que igual se podría haber repartido mejor.

8.4 Líneas de trabajo futuro

Cursed Island es una base de aprendizaje. De hecho, ahora que no habrá fechas de entregas se puede trabajar en mejoras e incluso hacer la idea más exacta de lo que se quería. Aún así, ya que he dado mi primer paso con este proyecto, me gustaría realizar otros géneros para ampliar mi conocimiento y quizás dar el salto a Unreal Engine.

Me llevo un buen sabor de boca en el trabajo que he realizado y que me gustaría repetir con más calma.

9. Glosario

- SNES: Son las siglas de Super Nintendo Entertainment System, una videoconsola sacada en 1990.

- NES: Son las siglas de Nintendo Entertainment System, una consola lanzada en 1983.

- Gold Master: Es la versión finalizada de un videojuego.

- Parallax: es un efecto que se usa para crear una ilusión espacial con medios de representación bidimensionales.

- Inkscape: es un software libre de vectores gráficos (como Adobe Illustrator).

- Unity assets: es una página con recursos que te proporciona Unity para tus juegos, como por ejemplo imágenes, sonidos, etc.

- Tiles: son pequeñas dimensiones de una imagen que se utilizan para crear suelos. Por ejemplo, si juntas muchos tiles que tengan dibujados una porción de arena puede simular el suelo de un desierto.

- Prefab: son elementos reutilizables y creados con una serie de características que serán llamados en un proyecto de juego.

- Checkpoint: traducido es punto de control. Es un punto de guardado en el progreso del juego.

- Flip: en Unity sirve para girar una imagen hacía el otro lado como si fuera un espejo.

- Respawn: es una palabra del inglés que significa regenerar. Es la acción de que un jugador o un enemigo aparezca de manera espontánea.

- Rigidbody 2D: es un término de Unity que determina que un objeto que le pongas esta propiedad va a disponer de una físicas, como la gravedad o la masa.

- Collider: es una propiedad de Unity que permite a los objetos colisionar entre si.

- Bool: es un tipo de dato lógico usado en programación que solo permite dos estados, true o false.

- Array: es un tipo de dato lógico usado en programación que guarda distintas celdas de información de un tipo de dato.

- Singletone: terminología usada en programación que crea una llamada de un objeto en su propia clase y desde cualquier parte de la aplicación puede ser invocada.

- NPC: son siglas en inglés de Non playable character. Es un personaje controlado por el juego pero no por el jugador.

- Trigger: en el contexto de esta palabra, existen colliders de tipo trigger que permiten detectar una colisión pero el jugador no la percibe.

- Timear: es una palabra españolizada de la palabra inglesa timer, la cual dicta el momento preciso que se realiza una acción.

- Bug: es un error que se encuentra en una aplicación informática.

10. Bibliografía

Colaborador de Wikipedia. Videojuegos de plataformas. *Wikipedia* [En línea]. <https://es.wikipedia.org/wiki/Videojuego_de_plataformas> [Consulta en Mayo de 2021].

Miguel "Starty!" Garay. [G2D] La historia de los juegos de plataformas (Donkey Kong, Pitfall y mas) (Parte 1). *Masgamers* [En línea]. <*https://www.masgamers.com/juegos-de-plataformas-generacion-2d*> [Consulta en Mayo de 2021].

José David Muñoz. Tres mil millones de personas en todo el mundo juegan ahora a videojuegos, según informa un estudio. *Hobbyconsolas* [En línea]. <https://www.hobbyconsolas.com/noticias/tres-mil-millones-personas-todo-mundo-juegan-ahora-videojuegos-informa-estudio-698121> [Consulta en Mayo de 2021].

Colaborador de documentación de Unity. System requirements for Unity 2020 LTS. *Unity Documetation* [En línea]. <https://docs.unity3d.com/Manual/system-requirements.html> [Consulta en Mayo de 2021].

Colaborador de documentación de Unity. Unity User Manual. *Unity Documetation* [En línea]. <Unity - Manual: Unity User Manual 2020.3 (LTS) (unity3d.com) > [Consulta en Mayo de 2021].