Universitat Oberta de Catalunya

uoc.edu

Pumpkin Hero

Genís García Pedrosa
Grado Multimedia
Trabajo Final de Grado - Videojuegos

Joel Servitja Feu
Joan Arnedo Moreno

Enero 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Pumpkin Hero*
Nombre del autor:	*Genís García Pedrosa*
Nombre del consultor/a:	*Joel Servitja Feu*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	*01/2021*
Titulación:	*Grado Multimedia*
Área del Trabajo Final:	*Videojuegos*
Idioma del trabajo:	*Español*
Palabras Clave	*Platformer, Videogame, Unity*

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

Se ha desarrollado un videojuego de plataformas 2D usando Unity, con tal de aprender y mejorar los conocimientos en dicha herramienta. Así pues, la idea principal ha consistido en la creación de un nivel capaz de ser superado por el jugador de inicio a fin.

Pumpkin Hero es el título del producto, influenciado por la ambientación de terror que se presenta y, sobre todo, por el protagonista el cual es un hombre con cabeza de calabaza. A pesar de haber mencionado la palabra "terror", la realidad es que el videojuego no tiene ninguna de las intenciones propias de dicho género, como pueden ser la de asustar o hacer pasar miedo. Mas bien, como la mayoría de videojuegos de plataformas, se pretende medir la agilidad y los reflejos del jugador para poder superar el nivel con éxito.

Para la realización del proyecto se ha establecido una planificación inicial, determinando las fechas para las distintas tareas a desenvolver. Si bien muchas de ellas se han cumplido, hay otras que tuvieron que ser modificadas o, incluso, ha habido casos donde se añadieron nuevas subtareas.

Finalmente, como resultado se ha obtenido un nivel completo de un videojuego para PC, junto a su pantalla principal y los menús correspondientes. De esta manera, se puede concluir que este proyecto ha supuesto una mejora personal en lo referente al desarrollo de videojuegos gracias a los conocimientos y experiencia adquirida.

Abstract (in English, 250 words or less):

A 2D platform video game has been developed using Unity, in order to learn and improve knowledge of that tool. Thus, the main idea has been to create a level capable of being overcome by the player from start to finish.

Pumpkin Hero is the title of the product, influenced by the horror setting that is presented and, above all, by the protagonist who is a man with a pumpkin head. Despite having mentioned the word "horror", the reality is that the video game does not have any of the intentions of this genre, such as to scare or make people look afraid. Rather, like most platform video games, it is intended to measure the agility and reflexes of the player in order to successfully overcome the level.

An initial planning has been established to carry out the project, establishing the dates for the different required tasks. Although many of them have been fulfilled, there are others that had to be modified or, there have even been cases where new subtasks were added.

Finally, as a result, a complete level of a PC video game has been obtained, along with its main screen and the corresponding menus. In this way, it can be concluded that this project has meant a personal improvement in terms of video game development, thanks to the knowledge and experience acquired.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Des de la antigüedad, el ser humano ha tenido la necesidad de jugar y entretenerse. Por ese motivo, durante el transcurso del tiempo se fueron inventando multitud de juegos, a la vez que se crearon también juguetes de todo tipo para los más pequeños. A día se pueden distinguir diferentes tipos de juegos, como los juegos de mesa (parchís, la oca, el uno,…) los juegos colectivos (el pilla-pilla, el escondite, el cuarto oscuro,…) e incluso los deportes (futbol, baloncesto, tenis,…).

Dicho esto, hoy en día existen juegos adaptados en el ámbito informático, los conocidos videojuegos. El mundo de los videojuegos ha estado en constante evolución al largo de los últimos años. Des de las antiguas máquinas recreativas y consolas que corrían los míticos títulos, como los son *Pong, Space Invaders* y *Tetris,* hasta lo que podemos encontrar hoy en día.

Si bien es cierto que antiguamente los videojuegos significaban únicamente un pasatiempo, lo cierto es que en la época actual pueden ser utilizados en muchos ámbitos, tales como la educación, la sanidad o en la competición deportiva. Por todo esto y gracias a la constante innovación tecnológica, el público abarcado es mucho más amplio. Un claro ejemplo de esto son los *smartphone,* donde los videojuegos han sido introducidos llegando al alcance de la mayoría de personas.

Así pues, con este contexto presente se ha pretendido desarrollar un videojuego con la finalidad de entretener al jugador. De esta manera, el usuario podrá evadirse, olvidar sus problemas por unos momentos y pasar un buen rato jugando.

1.2 Objetivos del Trabajo

Uno de los objetivos que tiene este Trabajo es la realización de un nivel completo del videojuego, el cual pueda ser completamente jugable. Para ello se requiere de un menú principal de juego y el nivel de juego en sí, con su correspondiente menú de pausa e interfaz. El producto final tiene que ofrecer una experiencia satisfactoria para el jugador, de manera que tiene que tener unas mecánicas de juego bien definidas e intuitivas, a la vez que una jugabilidad fluida. Añadido a esto es necesario contar con una parte gráfica bonita y agradable a la vista, a la vez que una música y efectos de sonido adecuados. La finalidad con el cumplimiento de estos objetivos es hacer que el usuario juegue varias veces e incluso que se quede con la necesidad de jugar más niveles distintos.

Por otro lado, en el ámbito personal se tiene un objetivo muy importante, el cual es aprender sobre el desarrollo de videojuegos y adquirir experiencia, además de obtener un producto para poder mostrar.

1.3 Enfoque i método seguido

Actualmente existen muchos videojuegos completamente diferentes, con muchísimas mecánicas ya creadas y con una serie de géneros ya definidos. Por esta razón, es difícil crear un producto des de cero propio sin dejarse influenciar por aquellos que ya han tenido éxito en tiempos pasados. Si bien se podría desarrollar un juego con unas mecánicas idénticas a uno ya existente, en este caso no se ha optado por ese camino debido a la intención de crear un videojuego propio sin ser copia de otro.

De este modo, con la mentalidad de crear un producto completamente nuevo, cabe decir que, igualmente, la influencia de otros videojuegos existentes se encuentra presente en este Trabajo. Así pues, la estrategia consiste en recolectar ideas provenientes de diferentes videojuegos, adaptarlas y combinarlas entre sí para poder obtener el resultado final.

Esta estrategia es la más adecuada para este tipo de trabajo ya que no requiere tanto esfuerzo en el diseño de juego y, por ende, se puede destinar más tiempo en otros aspectos como la programación y el desarrollo del videojuego en sí. Además, debido a la complejidad en la creación de los elementos gráficos que comprenden un videojuego 2D, se ha optado por utilizar *sprites* e imágenes de terceros, al igual que con la música y efectos de sonido.

1.4 Planificación del Trabajo

Para planificar los objetivos se ha tenido en cuenta los tiempos de entrega de las distintas PACs que componen el proyecto. No obstante, hay otros factores que influyen en cuanto la planificación y abasto del trabajo. Uno de ellos es el tiempo diario que se dispone para dedicarle, ya que en este caso se podría considerar una hora de media diaria. Otro punto importante es la experiencia que se pueda tener de trabajos anteriores, estudios, dedicación previa,… . Particularmente se tienen unos conocimientos básicos sobre el desarrollo de videojuegos, de manera que pueden surgir algunas dificultades concretas que puedan alterar mínimamente la planificación.

Así pues, es importante hacer una planificación realista, no muy ambiciosa, pero sin quedarse corta. Por eso, se ha elaborado un diagrama que muestra las tareas a realizar y sus tiempos de dedicación.

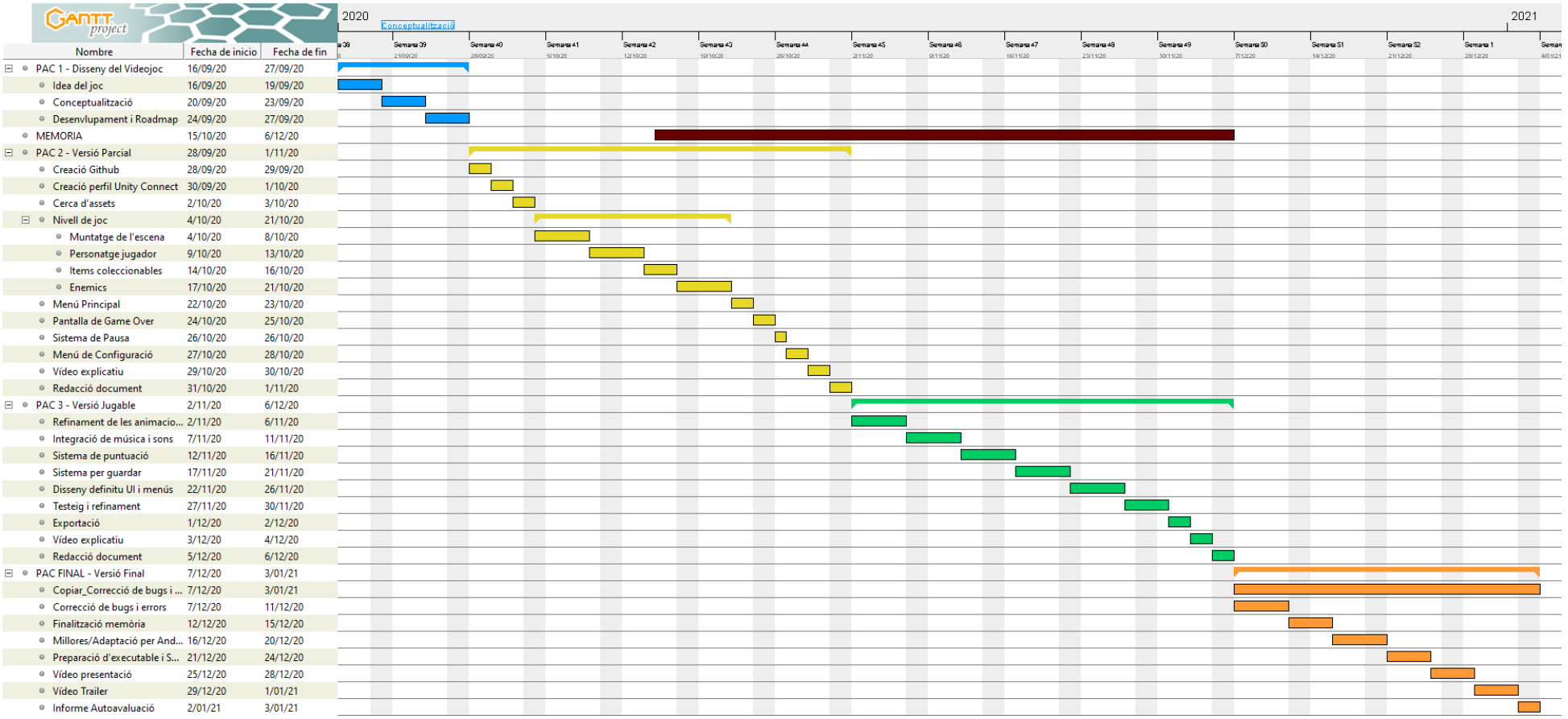


Figura 1. Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

Los productos obtenidos como resultado del Trabajo son:

- El videojuego *Pumpkin Hero* para PC.
- Documentos explicativos sobre la realización del proyecto, referente a las diferentes entregas de las PAC.
- Vídeos explicativos sobre la realización del proyecto, referente a las diferentes entregas de las PAC.
- Vídeo sobre las mecánicas y funcionamiento del resultado final.
- Tráiler del videojuego.
- Memoria del Trabajo con una explicación detallada del Trabajo.
- Repositorio Github con el Proyecto.

1.6 Breve descripción de los otros capítulos de la memoria

A continuación, en el resto de la memoria se contemplan los siguientes capítulos:

- **Estado del arte.** Se analizan los diferentes tipos de videojuegos de plataformas y algunos ejemplos existentes de juegos de plataformas 2D. También se estudian diversas tecnologías óptimas para el desarrollo de videojuegos de plataformas.
- **Definición del juego.** Conceptualización del Trabajo, explicando diferentes aspectos como la historia del juego, la ambientación, los personajes y otros elementos, la interacción con el jugador y los objetivos que se le proponen.
- **Diseño técnico.** Profundización sobre el motor de juego elegido, listado de los distintos *assets* usados, ya sean provenientes de fuentes externas o propios, definición de la arquitectura del proyecto y explicación del comportamiento de la IA de los enemigos.
- **Diseño de niveles.** Mapa del nivel con aclaraciones para lograr superarse y justificación de su diseño.
- **Manual de usuario.** Requisitos para poder jugar a *Pumpkin Hero* e instrucciones de juego.
- **Conclusiones.** Análisis del resultado obtenido del proyecto.

2. Estado del arte

2.1 Revisión sobre el género del videojuego

El Trabajo consiste en la creación de un videojuego de aventuras perteneciente al género de plataformas, caracterizados por tener que caminar, correr, saltar o escalar por una serie de plataformas o acantilados mientras que, a su vez, se recogen objetos para completar el nivel. Durante el camino aparecen enemigos que dificultan el avance del jugador e, incluso, pudiéndole ocasionar el hecho de perder el juego. Este tipo de videojuegos suelen usar vistas de desplazamiento horizontal hacia la izquierda o hacia la derecha.

Dentro del género de plataformas se pueden encontrar varios subgéneros como, por ejemplo [1]:

- **Plataformas de exploración**: tienen un estilo de juego diferente a los de plataformas tradicionales, predominando el uso de escenarios más grandes, que se han de recorrer varias veces y en varias direcciones, una evolución continua de las habilidades del personaje y generalmente con un argumento y ambientación mucho más desarrolladas.
- **Plataformas isométricos**: Los isométricos usan este tipo de proyección para darle un aspecto en 3D con gráficos en 2D. Los juegos de plataforma isométrica utilizan una representación gráfica que al proyectarse se forman ángulos de 120º.
- **Plataformas 2D**: Los videojuegos de plataformas 2D se caracterizan por utilizar gráficos de apariencia plana y por usar *scroll* de la pantalla, tanto en horizontal como en vertical. Es el estilo de plataformas más clásico y es el más característico de las videoconsolas de 8 bits y 16 bits. Actualmente el desarrollo de videojuegos 2D sigue siendo muy popular en juegos creados por compañías independientes.
- **Plataformas 2.5D**: Son Videojuegos en 2 dimensiones que, mediante ciertas técnicas como la perspectiva, sombras y demás, aparentan ser en 3 dimensiones.
- **Plataformas 3D**: Los videojuegos en 3D utilizan gráficos con modelos en tres dimensiones, lo que permite que el juego tenga libertad de movimiento en el espacio y no solo en un plano, gracias a los avances en las tecnologías 3D.

Para este proyecto, se ha desarrollado un videojuego de plataformas 2D. El ejemplo clásico de este género recae en la saga de *Super Mario Bros,* el cual ha sido muy popular al largo de los años y donde, hoy en día, sigue sacando nuevos títulos. Así pues, esta saga ha tenido bastante influencia en el videojuego realizado para el Trabajo.

Figura 2. Super Mario Bros, Nintendo, 1985 [2]

Figura 3. Super Mario Bros. U, Nintendo, 2019 [3]

Además, otros videojuegos se han usado como referencia para desarrollo de *Pumpkin Hero* han sido *Rayman Legends* y *Oddmar*, siendo este último un ejemplo de videojuego de plataformas para Android.

Figura 4. Rayman Legends, Ubisoft, 2013 [4]

Figura 5. Oddmar, Mobge Ltd, 2019 [5]

2.2 Revisión sobre la tecnología del videojuego

Para el desarrollo de muchos de los títulos más famosos del mundo de los videojuegos, las grandes empresas han optado por utilizar un motor propio interno de la compañía. Aun así, existen varias herramientas para el desarrollo videojuegos las cuales no tienen nada que envidiar a los motores de videojuegos mencionados previamente y que, además, se encuentran al alcance de pequeñas empresas y de desarrolladores independientes.

Por ese motivo, es necesario analizar bien las diferentes opciones que se tienen disponibles para la realización del proyecto y elegir, así, la más óptima. Algunas de las herramientas más interesantes per el desarrollo de videojuegos que pueden ser útiles a la hora de crear un videojuego de plataformas 2D son [10]:

- **GameMaker:** Es un motor enfocado al 2D aunque también dispone de funcionalidades para el desarrollo 3D. Ofrece un sistema muy sencillo de *drag and drop*, con objetos, eventos y acciones. Además, dispone de su propio lenguaje de programación GML, con una sintaxis sencilla. Permite exportar a muchos sistemas gracias a los módulos de exportación, monetización y análisis de estadísticas. Originalmente fue creado para ayudar a los estudiantes, motivo por el cual, hoy en día, es uno d ellos mejores programas para aprender a crear videojuegos, entender la lógica y su funcionamiento. No obstante, uno de los problemas es que dispone de una versión gratuita para hacerse una idea de su potencial per que, a la hora de la verdad, toca pasar por caja.

Figura 6. Ventana de trabajo de GameMaker [6]

- **Stencyl:** Es una herramienta que permite crear juegos 2D sin programar, utilizando un sistema *drag and drop* de eventos y acciones. Por lo tanto, es una opción muy fácil debido a su funcionamiento: Es necesario ir aplicando la lógica y uniendo las piezas como si fuera un puzle, aunque también se pueden descargar módulos de plataformas gratuitamente. Además, se puede comercializar el videojuego para Windows y navegador de forma gratuita.

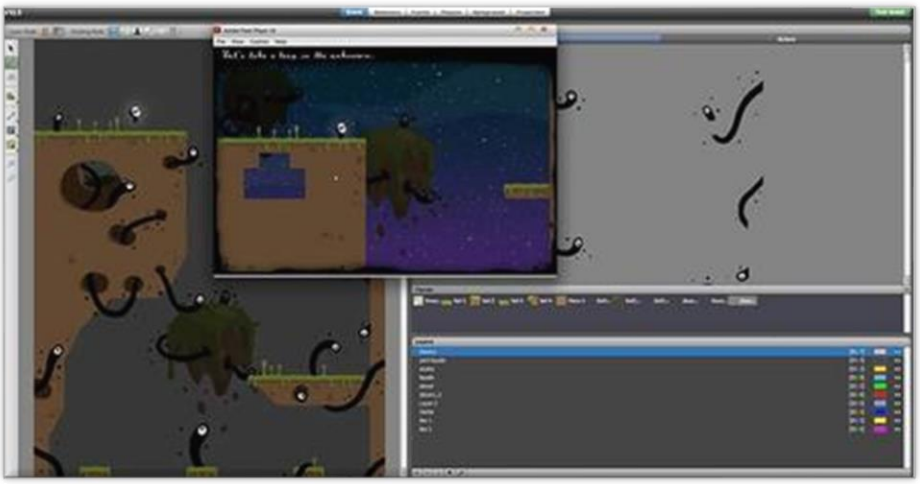


Figura 7. Ventana de trabajo de Stencyl [7]

- **Unity:** Actualmente es uno de los mejores motores 2D y 3D y, por este motivo, uno de los más utilizados. Esto es debido, principalmente, a que ofrece respuesta a las necesidades de los desarrolladores independientes, con licencia gratuita i con precios asequibles, además de permitir la exportación para un montón de plataformas. Es muy recomendable para la gente iniciada en el sector, ya que es muy intuitivo, dispone de muchísima información y tiene mucha versatilidad. Es tanto su potencial que una gran cantidad de empresas de videojuegos importantes ya trabajan con este motor.

Figura 8. Ventana de trabajo de Unity [8]

- **Unreal Engine:** Otro clásico del sector, proporcionado por *Epic Games* consigue unos acabados sorprendentes con poco esfuerzo gracias a la iluminación dinámica. No obstante, suele estar muy relacionado con los videojuegos del estilo FPS o con un estilo visual marcadamente realista. Dispone de herramientas como un avanzado editor de materiales, un creador de cinemáticas o la programación visual mediante los *Blueprints* para exprimir el hardware al máximo. Es totalmente gratuito, aunque *Epic Games* se queda con el 5% de los ingresos generados por el título.

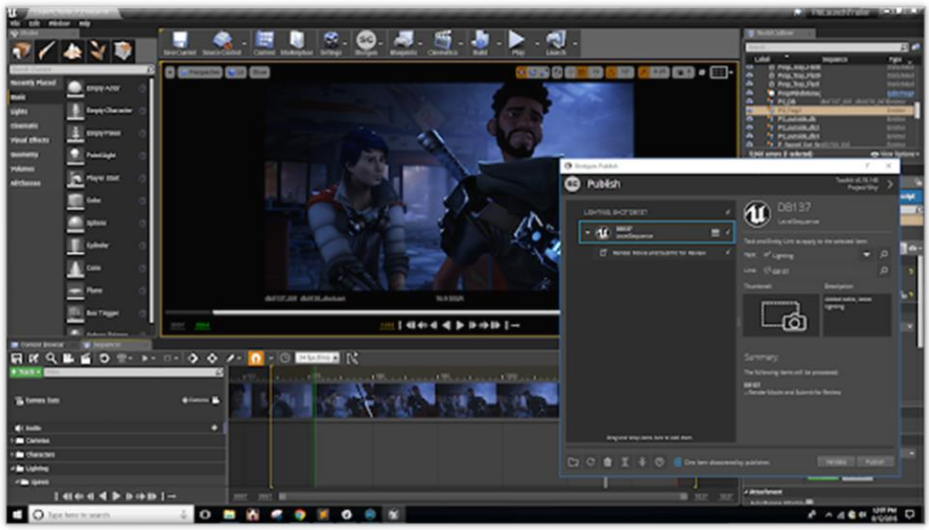


Figura 9. Ventana de trabajo de Unreal Engine [9]

Después de este análisis sobre algunos de los motores de videojuegos más interesantes que hay actualmente, se ha decidido que la opción más adecuada es Unity. Aunque objetivamente Unity sea una de las mejores opciones (si no la mejor) debido al cumulo de ventajas que ofrece, cabe decir que también influye, sobre todo, el hecho de la experiencia personal relacionada con este motor y las ganas de aprender. Por eso, queda decidido que Unity es el *engine* utilizado para este proyecto.

3. Definición del juego

3.1 Descripción del juego

Pumpkin Hero es un videojuego de plataformas 2D, formado de distintos niveles que el jugador deberá de ir superando progresivamente para poder seguir avanzando. Es necesario destacar que este Trabajo tan solo dispone de un nivel de juego, mostrando las características más importantes que comprenderían el videojuego completo.

Así pues, el objetivo principal del jugador en cada uno de los niveles consiste en llegar des del punto inicial al punto final, superando todos los obstáculos con los que se vaya encontrando. Estos obstáculos son una manera de impedir y dificultar el avance del jugador. Algunos ejemplos son los distintos tipos de plataformas que pueden provocar la caída del jugador al vacío, ocasionando el fin del juego, o los enemigos que intentaran acabar con la vida del protagonista.

No obstante, durante el recorrido del nivel aparecen objetos coleccionables que tienen una funcionalidad específica dentro o fuera del nivel, como por ejemplo monedas, armas o ítems clave. De esta manera, el jugador tiene otros objetivos secundarios, aparte de completar el nivel.

3.2 Tipo de interacción juego-jugador

En *Pumpkin Hero,* el jugador se pone en la piel del personaje protagonista, el cual deberá controlar mediante los inputs correspondientes, concretamente teclado y ratón en PC. Estos controles ofrecen la posibilidad de hacer correr al personaje, saltar y atacar cuando sea necesario.

Durante el nivel de juego, el jugador se encuentra encima del terreno y deberá evitar caer al vacío. Así pues, será necesario una interacción con todos aquellos elementos del juego que permitan el avance del personaje, como por ejemplo todo el conjunto de plataformas que se encuentran en el nivel. Además, existen enemigos que atacan al protagonista, y por eso, el jugador puede atacarles, saltarles encima para eliminarlos o intentar esquivarlos. Otros elementos interactuarles muy importantes son los elementos coleccionables, que serán recolectados una vez el personaje mantenga un contacto directo con ellos.

Algunas de las capacidades requeridas dl jugador pueden ser la precisión, es decir, tener la capacidad de controlar al personaje de forma precisa evitando que caiga en agujeros y siendo capaz de esquivar enemigos. También, relacionado con este último punto, es necesario disponer de un mínimo de reflejos para realizar acciones en el momento adecuado.

3.3 Plataforma de destino

Pumpkin Hero es un videojuego para PC, concretamente para Windows. Aun así, no debería ser problema el hecho de llevarlo a otro sistema operativo distinto. A pesar de esto, en la planificación del proyecto se contempló la posibilidad de crear una versión del juego para Android, aunque debido al tiempo eso no ha sido posible.

El motivo principal por el cual se eligió esta plataforma de destino se basa fundamentalmente en la jugabilidad. Así pues, se considera mucho más cómodo y satisfactorio para el jugador el uso del teclado y ratón para un videojuego de plataformas. A pesar de todo, un videojuego de plataformas para móvil es completamente viable. Además, los problemas de rendimiento suelen ser menores en desktop que en cualquier dispositivo móvil, el cual requiere de una mayor optimización y limitación a la hora de desarrollar el juego. Por último, otro motivo que también ha descartado la opción de destinar el juego a consolas, es el grado de facilidad a la hora de hacer pruebas en pleno desarrollo. Al destinar el videojuego a PC, las pruebas pueden ser directas sobre el programa de desarrollo, mientras que, para el caso de cualquier consola se requiere de tecnología externa, por ejemplo, mandos y, por qué no, la misma consola para realizar el *testeo* del producto final.

3.4 Historia, ambientación y trama

El videojuego de *Pumpkin Hero* está ambientado sobre una temática de terror, aunque su intención no sea la de asustar o hacer pasar miedo al jugador.

La historia tiene lugar en un antiguo pueblo, el cual ha quedado bajo los efectos de una temida maldición, atrapando la vida y alterando la forma física de sus habitantes. Desgraciadamente, no se conoce apenas nada sobre lo que está pasando. Lo único que se ha notado es que en el cementerio es donde se concentra la mayor parte de energía maligna, pudiendo ser el foco principal del problema. No obstante, por ese motivo es un lugar peligroso, repleto de criaturas extravagantes y fenómenos paranormales. Sin embargo, es necesario adentrarse en el cementerio y poder llegar al fondo del asunto, donde se encuentra el principal responsable de esto: un ser maligno proveniente de una dimensión oscura.

3.5 Definición de personajes y elementos. Interacción entre los agentes del juego

En el videojuego, el personaje principal es el protagonista el cual es controlado por el jugador. No obstante, también habrá un tipo de criatura que actuará como enemigo suyo y otros personajes que serán parte del entorno y no tendrán efecto alguno durante el avance del nivel más que decorar y ambientar la escena.

Por otro lado, está el escenario formado por un conjunto de elementos de entre los cuales se encuentran los suelos, las plataformas y los objetos coleccionables.

- ***Pumpkin Hero / Sam*:** Es el protagonista de la historia y, por lo tanto, el jugador es capaz de controlarlo integrándose, así, en el mundo del videojuego. Sam era un agente de seguridad que sufrió los efectos de la maldición, provocando un cambio peculiar en su cabeza la cual fue transformada en calabaza. Aun así, no ha perdido sus dotes de combate y, por ese motivo, es uno de los más indicados para realizar una expedición en el cementerio. El jugador, mediante este personaje, es capaz de correr, saltar por las diversas plataformas y atacar con los puños o con el objeto de ataque que haya encontrado.

Figura 10. Sam

- ***Criatura del cementerio, araña*:** Es el enemigo del protagonista en el nivel actual. Hay varios de ellos y se encuentran siempre patrullando una zona concreta, moviéndose de un lado a otro. Si el jugador se acerca demasiado, no dudarán en atacarle, así que es necesario mantener las distancias. Aun así, si su punto débil es su parte superior, por lo que al saltarles encima se acabara con sus vidas.

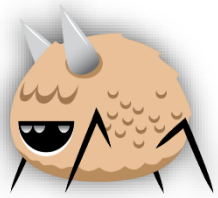


Figura 11. Araña

- **Personajes decorativos:** Estos personajes no influyen en el juego, sino que más bien forman parte de la decoración del escenario. Se tratan criaturas habitantes del cementerio, como zombis, brujas, fantasmas y demonios, además de inocentes búhos.

Figura 12. Personajes decorativos

- **Plataformas:** Con el fin de alcanzar el final del nivel, el jugador deberá pasar por diversas plataformas con distintos comportamientos. Existen plataformas en movimiento, otras que aparecen y desaparecen y otras que actúan como un pistón aplastante.

Figura 13. Plataformas

- **Coleccionables:** Actualmente, en el videojuego existen dos tipos de coleccionables. Uno de ellos son monedas que se encuentran situadas por todo el escenario. Al tocarlas, el jugador aumentará su contador de monedas. A pesar de no tener ninguna funcionalidad extra implementada, la intención es usarlas para poder comprar ciertos objetos fuera del nivel (cosméticos, niveles extra, funcionalidades adicionales, …).

Figura 14. Moneda coleccionable

Por otro lado, hay tres objetos coleccionables que son: un pergamino, un libro, y una olla. Estos tres objetos se encuentran de forma separada por el nivel y están situados en zonas más complejas para su obtención. El nivel puede ser completado sin conseguir estos ítems, pero sí que será indispensable obtenerlos para poder desbloquear futuros niveles.

Figura 15. Ítems coleccionables

- **Portal mágico:** Para pasar el nivel, el jugador debe interactuar con el portal mágico que se encuentra al final. Solo hace falta pasar por él y el nivel será completado.

Figura 16. Portal mágico

- **Bate de béisbol:** El bate de béisbol es un equipamiento que se encuentra por el mapa y que el jugador puede recoger. Al equipárselo, el personaje coge el bate con la mano aumentando su daño de ataque y alcance.

Figura 17. Bate de béisbol

3.6 Objetivos planteados al jugador

A medida que se vayan introduciendo nuevos niveles en el juego, el objetivo general del jugador será completarlos todos. Además, para los más coleccionistas, puede ser interesante el hecho de disponer de todos los objetos cosméticos que pueda haber en una futura tienda.

Partiendo de esta base y con lo que se dispone actualmente, se pueden definir tres objetivos para cada nivel.

- El primero de todos consiste en superar el nivel de inicio a fin, llegando al portal final con vida. Este objetivo será también parte de los demás objetivos definidos a continuación.

- Este objetivo consiste en conseguir los tres ítems coleccionables (pergamino, libro, olla) para completar el nivel 100%. No obstante, en versiones futuras del videojuego será indispensable su obtención para poder desbloquear niveles posteriores en ciertos puntos del juego. Es importante destacar que, para dar por conseguido un ítem, se requiere completar el nivel hasta al final con ese o esos ítems.
- El último objetivo hace referencia al segundo objetivo general mencionado al inicio de la sección, el cual iba destinado a desbloquear ciertos objetos o funcionalidades que no son obligatorias para el avance en el juego. Así pues, las monedas que aparecen en los niveles son la clave para esto, convirtiéndose así en un objetivo a conseguir para todos aquellos que quieran usar la tienda.

3.7 Concept Art

Al inicio del proyecto se hicieron unos concept para enfocar la idea. Se utilizaron imágenes de internet para crear composiciones con la idea que se tenía en mente, así que estas no son imágenes definitivas del juego.

Figura 18. Concept Art. Plataformas móviles

Figura 19. Concept Art. Objetos coleccionables

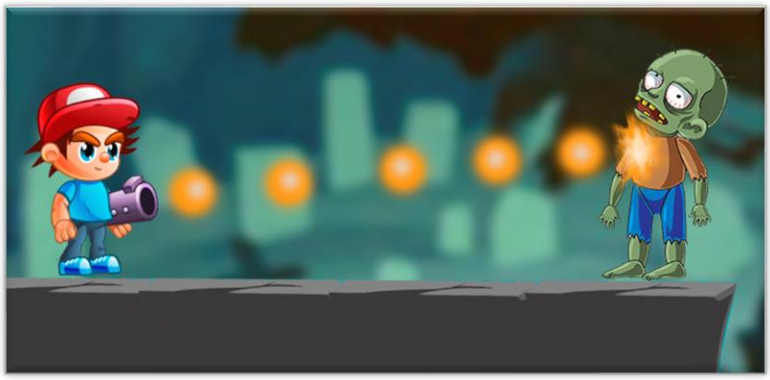


Figura 20. Concept Art. Ataque

3.7 Arte y diseño final

Una vez visto los concept, a continuación, se muestran unas pocas imágenes sobre el arte y diseño del nivel de juego.

Figura 21. Imágenes reales del juego

4. Diseño Técnico

4.1 Entorno de desarrollo elegido

En el capítulo "2.2 Revisión sobre la tecnología del videojuego" de esta memoria ya se analizaron algunos de los entornos que podrían ser aptos para este proyecto. Se describió cada uno de ellos y finalmente se decidió que la mejor opción para este caso era Unity. La conclusión es que Unity es uno de los motores más populares y potentes actualmente debido al cúmulo de posibilidades que ofrece y a su curva de aprendizaje. Además, la básica experiencia personal en este entorno y las ganas de aprenderlo y profundizarlo son un punto muy importante que fortalecen todavía más esta decisión.

Así pues, Unity dispone de varias características, como las listadas a continuación [11]:

- **Exportación a multitud de plataformas.** Una de las características más importantes y cómodas de Unity es que soporta la exportación para una cantidad enorme de plataformas. No solo se puede elegir la plataforma con la se va a trabajar creando y editando el juego, cuyo editor en este momento soporta Windows y MacOS, además de Linux de forma experimental, sino que se puede crear el juego para más de 25 plataformas.
- **Documentación.** Unity dispone de una excelente documentación en línea. En esta documentación se encuentra el manual de Unity, donde se puede consultar cada uno de sus apartados, desde cómo actualizar a una versión concreta, hasta guías de expertos para realizar tareas bastante avanzadas y que no todo el mundo necesita. Dentro de la documentación aparece un historial de versiones anteriores para consultar aquella con la que se esté trabajando. También se dispone de la documentación de la API de programación o scripting con todos los *namespaces*, con sus clases, métodos y ejemplos de uso de cada uno de ellos.
- **Inmensa comunidad de usuarios.** Otra de las grandes ventajas de Unity es la gran comunidad de usuarios que tiene, no solamente dentro de los propios foros de Unity sino por todo Internet. Es muy extraño que surja una pregunta, una duda o algún problema durante el desarrollo del juego y que no haya sido ya respondido o se esté discutiendo en los foros de Unity o alguna otra plataforma.
- **Licencia gratuita.** Dispone de una modalidad de licencia gratuita que permite desarrollar juegos des del primer momento sin coste alguno, con acceso a recursos para facilitar esta tarea.
- **Unity Asset Store.** Se trata de un espacio desarrollado para Unity, el cual contiene multitud de *assets* creados por la propia compañía y por terceros. Des el mismo editor de Unity se puede acceder fácilmente e importar los *assets* una vez se haya realizado su descarga, ya sea de forma gratuita o de pago.
- ***GameObjects* y componentes.** Todo lo que haya en la escena de Unity se trata de un *GameObject*. Así pues, de por sí solo un *GameObject* no tienen ninguna funcionalidad. Es por eso que

existen los componentes, los cuales se añaden a los distintos *GameObjects* para dotarlos de un comportamiento propio. Además, es posible crear y modificar los componentes para lograr el comportamiento adecuado.

- **Motor de físicas.** Gracias al motor de físicas del que dispone Unity se pueden añadir físicas al juego de manera sencilla, a la vez de ofrecer la capacidad de detectar colisiones y realizar simulaciones realistas.

4.2 Requisitos técnicos del entorno de desarrollo

Los requisitos técnicos del editor de Unity son [12]:

Requisitos Mínimos	Windows	macOS	Linux
Sistema Operativo	Windows 7 (SP1+) y Windows 10, 64-bit	Sierra 10.12.6+	Ubuntu 16.04, Ubuntu 18.04, y CentOS 7
CPU	Arquitectura x64 con set de instrucciones SSE2	Arquitectura x64 con set de instrucciones SSE2	Arquitectura x64 con set de instrucciones SSE2
GPU	DX10, DX11 y DX12	Intel y AMD con Metal	Nvidia y AMD con OpenGL 3.2+ o Vulkan

4.3 Herramientas usadas

Para el desarrollo del proyecto se han usado las siguientes herramientas:

- **Unity 2019.3.11f1.** Es el motor de juego usado.
  - Unity Asset Store. Plataforma de *assets* en línea.
  - Cinemachine. Creación de la cámara del juego.
  - TextMeshPro. Elaboración de textos sofiticados.
- **Visual Studio 2019.** Editor de programación ala hora de crear los *scripts*.
- **GitHub.** Repositorio git y para control de versiones.
- **OpenGameart.** Plataforma de *assets* en línea con licencia libre.
- **FreeSound.** Plataforma de sonidos y músicas en línea con licencia libre.
- **Adobe Premiere Pro CC 2019.** Editor de vídeo.
- **Audacity.** Editor de audio.
- **Microsoft Word 2019.** Procesador de textos para la documentación del proyecto.

4.4 *Assets* y recursos del juego

Tanto los *assets* que conforman la parte gráfica del juego como la música y los efectos de sonido han sido sacados de fuentes externas. Esto es debido a la cantidad de tiempo que requiere su creación y a la poca práctica que se dispone para la elaboración de recursos 2D y sonido. Sin embargo, algunos sonidos sí que han sido modificados, ya sea mediante

efectos o cortando su duración. Así pues, en el desarrollo del Trabajo se han integrado estos recursos y se ha elaborado una serie de scripts propios. Los siguientes recursos han sido adquiridos y realizados por terceros:

- **Halloween 2D asset Pack:** Obtenido a partir de la Asset Store de Unity, este Pack contiene varios *sprites*, tanto como de personajes, de entorno y hasta de interfaz. Todo el conjunto de menús ha estado elaborado gracias a estos *assets*. Esto incluye el menú principal, el menú de pausa y los menús de victoria y derrota.

Figura 22. Ejemplo de sprites UI

Por supuesto, el escenario también ha estado creado usando este pack. Desde las plataformas, los personajes animados los cuales venían con la secuencia de imágenes para la animación, y el suelo gracias a una paleta de Tiles con la que venía.

Figura 23. Ejemplo de Tiles, plataformas y personajes

Por último, los *sprites* de los tres ítems coleccionables, es decir, el pergamino, el libro y la olla, además de las fuentes de texto, también se incluyen en este pack.
Disponible en la *Asset Store* de Unity.
https://assetstore.unity.com/packages/2d/gui/icons/dark-themed-2d-pack-126098

- **Spine 2D Character Asset.** Este pack incluye el personaje principal, concretamente el protagonista del juego el cual es controlado por el jugador. Incluye dos versiones: una que utiliza un sistema de *spine* o esqueleto para animación y otra con las animaciones como secuencias de imágenes, el cual fue el utilizado para este proyecto. Lo importante se encuentra en la carpeta "Character", ya que contiene las imágenes para el personaje.

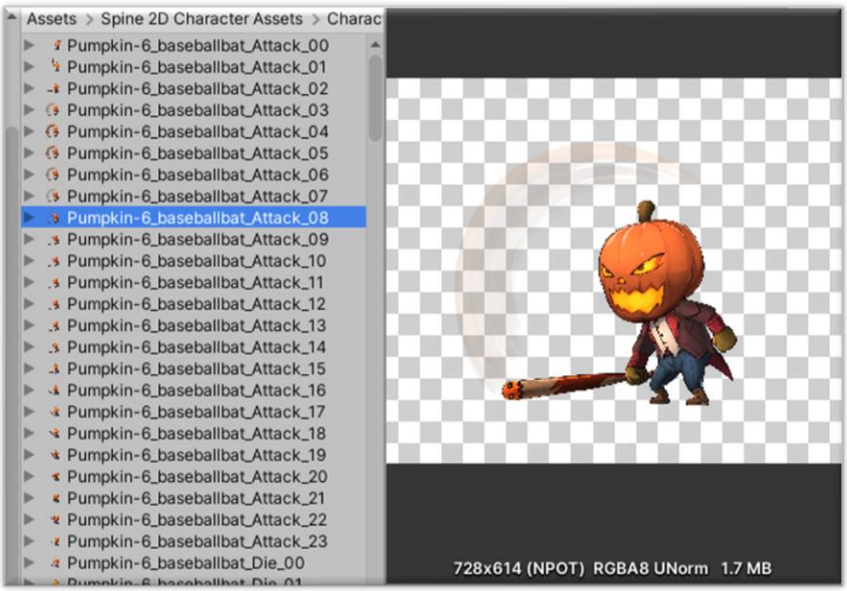


Figura 24. Protagonista, secuencias de imágenes para animación

Disponible en la *Asset Store* de Unity.
https://assetstore.unity.com/packages/2d/characters/2d-chibi-pumpkin-agent-character-spine-151077

- **Araña enemiga.** Para las arañas enemigas que aparecen durante el nivel de juego se han utilizado *sprites* con las animaciones como secuencia de imágenes.

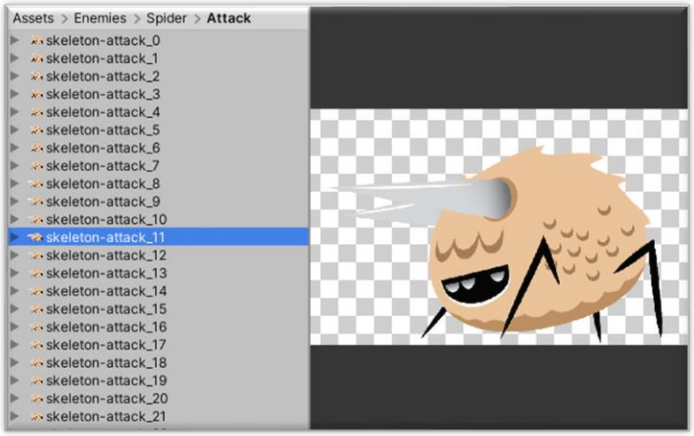


Figura 25. Araña, secuencias de imágenes para animación de ataque

Disponible en:
https://bevouliin.com/furry-monster-free-2d-game-asset/

- **Monedas.** Las monedas coleccionables del nivel y su animación de rotación han sido adquiridas a través la web de OpenGameart. https://opengameart.org/

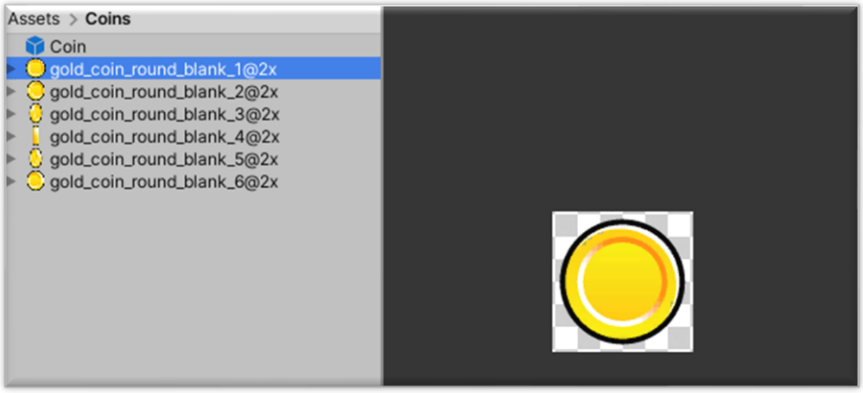


Figura 26. Moneda, secuencias de imágenes para animación de rotación

- **Sistemas de partículas.** Los sistemas de partículas que se ven, por ejemplo, al morir los enemigos, al recibir daño o en el portal forman parte de un *asset* externo llamado "Cartoon FX".

Figura 27. Sistema de partículas del portal

Disponible en la *Asset Store* de Unity.
https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-109565

- **Música y sonidos.** Todos los efectos de sonido y la música de fondo del menú principal provienen de:
Fresound: https://freesound.org/
OpenGameart: https://opengameart.org/

- **Scripts:** Todo el contenido de *scripts* dentro de la carpeta "Scripts" del proyecto es propio.
    - AssetAudio: Gracias a este *script* es posible añadir los sonidos a los elementos del juego. Además, dispone de

funciones que facilitan el control a la hora de reproducir sonidos.

- AudioManager: Los sonidos con carácter 2D, es decir, los que siempre suenan igual independientemente de la posición de la cámara, se encuentran en un *GameObject* con este *script*. Tiene un comportamiento muy similar al de *AssetAudio*.
- AudioSlider: Ha sido elaborado gracias a los foros de la comunidad. Es el principal encargado de regular el volumen del juego, a partir de los diversos canales del *AudioMixer* y de los *sliders* del menú de configuración de sonido.
- BaseballBat: Equipa el bate al personaje cuando el jugador colisiona con el bate en el nivel.
- BottomLimit: Se trata del detector que identifica al jugador cuando cae al vacío y activa la función de perder el juego del *GameManager*.
- Coin: Para detectar cuando el jugador obtiene una moneda.
- Collectables: Referente a los tres ítems coleccionables. Detecta cuando el jugador obtiene uno de ellos e identifica de cuál se trata.
- EndLevel: Básicamente se encarga del comportamiento del portal final y de llamar a la función de victoria del juego.
- Enemy: Es el comportamiento de las arañas enemigas, es decir, el movimiento, el ataque, su muerte,… .
- EnemyTop: Detecta cuando el jugador colisiona con la parte superior de las arañas y aplica las instrucciones oportunas.
- GameManager: Encargado de controlar el estado del juego.
- GameMenuUI: Contiene los elementos de interfaz y realiza funciones sobre ellos.
- LevelManager: Parecido al *GameManager* pero enfocándose únicamente al nivel jugado.
- ParticlePlatformGhost: Desactiva la partícula de destello que activan a ciertas plataformas fantasma, cuando estas ya están activas.
- PlatformGhostTriggerFadeOut: Es el detector del jugador para iniciar la aparición de una plataforma fantasma en concreto.
- PlatformFalling: Relacionado con las plataformas aplastantes que actúan como un pistón. Contiene su comportamiento, como el movimiento de subir y bajar y su colisionador.
- PlatformGhost: Común para todo tipo de plataformas fantasmas. Tiene una variable para indicar si la plataforma será visible o no al inicio. Además, contiene las funciones para hacerlas aparecer o desvanecerlas.
- PlatformMoving: Para las plataformas en movimiento. Realiza el movimiento a partir de dos puntos y detecta el

contacto con el jugador para añadirlo como hijo con tal de que este se mueva con la plataforma en contacto.

- PlatformMovingOnPlayer: Referente a plataformas como la del final del nivel, que inician su movimiento cuando el jugador entra en contacto. A partir de aquí inician un movimiento único hacia su punto de destino. Muy similar al *script* de "PlatformMoving".
- PlayerCombat: Se usa para que el jugador ataque. Se crea una zona circular de impacto para detectar lo que golpea y, si es un enemigo, aplicar el daño.
- PlayerController: Controla el movimiento del jugador, el salto y si está tocando el suelo. También tiene funciones como la de girar el personaje según la dirección a donde se dirija.
- PlayerManager: Es el estado del jugador. Se encarga de equipar el bate, cambiando todas las animaciones, controla cuando el jugador es inmune al ser golpeado e incluye las funciones de tomar daño o morir en función de su vida.
- SceneController: Principalmente para usar funciones como la de carga de escena, salir o pausar.
- Sound: Clase para los sonidos usada en los *scripts* de audio para introducir los distintos audios del juego identificando su nombre, clip, volumen, si se reproduce en bucle, grupo del *AudioMixer* y el *AudioSource*.

4.5 Esquema de arquitectura del juego y componentes

Actualmente, el videojuego está compuesto de dos escenas: el menú principal y el nivel.

- **Menú Principal.** Es la escena de "MainMenu" y tiene la siguiente estructura:

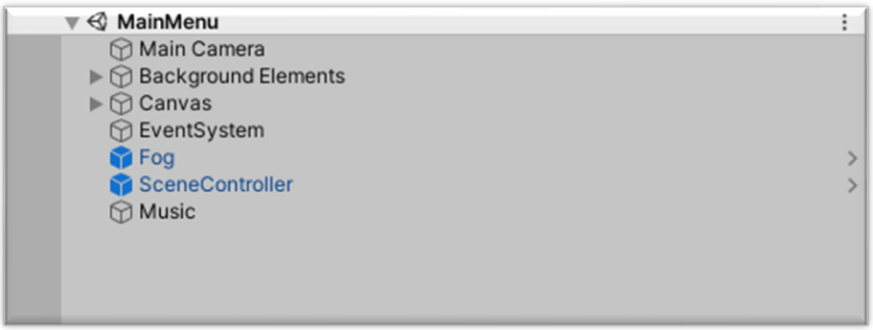


Figura 28. Jerarquía del Menú Principal

- MainCamera: Se trata de la cámara principal, la cual renderiza la escena.
- Background Elements: Son todos los elementos decorativos de fondo. Las imágenes del cielo, los árboles, la casa, las tumbas,… .
- Canvas: Incluye el panel de los menús y todos sus elementos como botones y otros submenús. Contiene tres bloques

principales que son el panel del menú principal con sus tres botones, el menú de confirmación al pulsar el botón de salir del juego y el menú de configuración de audio con los *sliders* y textos correspondientes.

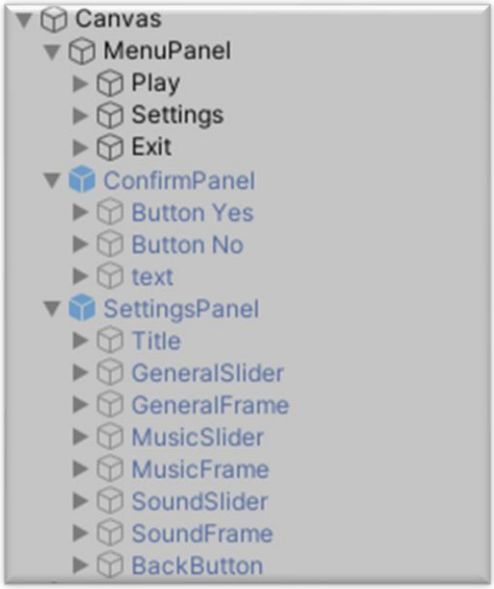


Figura 29. Jerarquía del Canvas del Menú Principal

- EventSystem: Creado con el Canvas para el control de eventos.
- Fog: Sistema de partículas que generan una niebla verdosa en la parte inferior de la escena.
- SceneController: Se usa para llamar funciones des de los botones, como para cargar la escena del nivel o salir del juego.
- Music: Contiene un *AudioSource* para la música de fondo.

- **Nivel de juego.** Es la escena de "Level" y tiene la siguiente estructura:

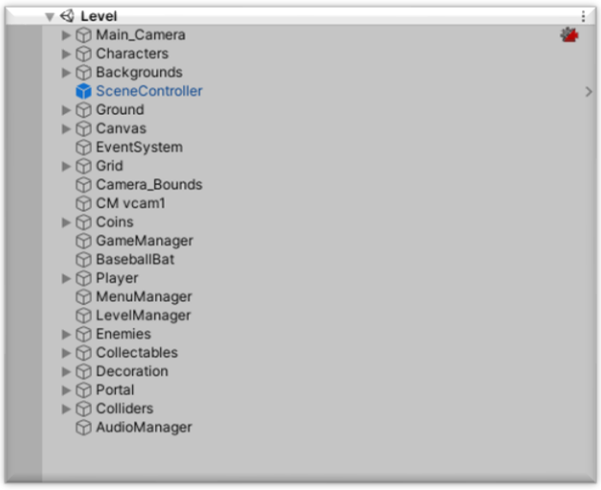


Figura 30. Jerarquía de la escena del nivel de juego

- MainCamera: Se trata de la cámara principal, la cual renderiza la escena. En su interior hay otra cámara específica para renderizar los elementos de la capa de fondo. Está vinculada a las funcionalidades de *Cinemachine* para seguir al personaje y limitar su movimiento.
- Characters: Son todos los personajes de decoración que se encuentran en la escena, como zombis, fantasmas y brujas.
- Backgrounds: Objetos decorativos de fondo, es decir, el cielo, nubes, tumbas y todo aquello que forma parte de la decoración de atrás.
- SceneController: Contiene funcionalidades de la escena, como pausar, cargar o salir del nivel.
- Ground: Es el conjunto de plataformas con las que el jugador puede interactuar. Así pues, se encuentras las plataformas en movimiento, las plataformas fantasmas y las plataformas aplastantes.
- Canvas: Son todos los elementos de interfaz y menús.

Figura 31. Jerarquía del Canvas del nivel de juego

Contiene la barra de vida, la puntuación de monedas y objetos coleccionables, el menú de pausa y los paneles de derrota y victoria.

- EventSystem: Creado con el Canvas para el control de eventos.
- Grid: Gracias a este *GameObject* se genera una cuadrícula para pintar el escenario a partir del *tilemap*.

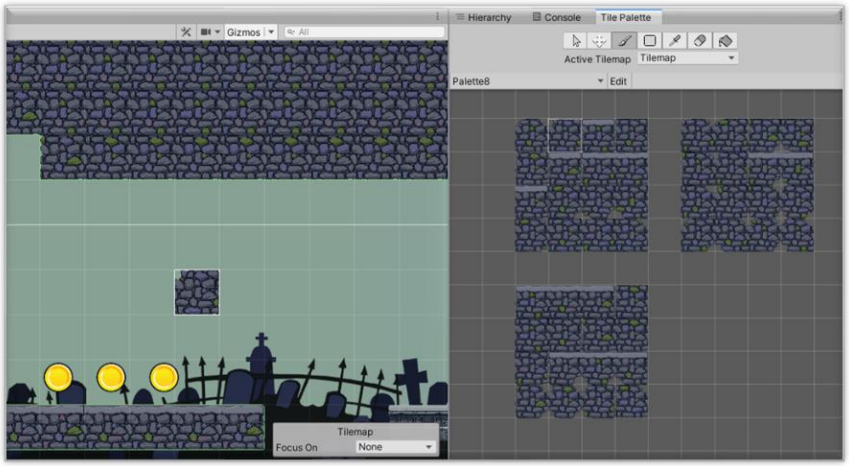


Figura 32. Tilemap del nivel

Se usa un *Composite Collider 2D* para generar el colisionador automáticamente para todas las *tiles*. Gracias a esta herramienta se ha creado el nivel de piedra y de tierra.

- CameraBounds: Se trata de un *Collider 2D* que limita el movimiento de la cámara. De esta manera, si el jugador cae al vacío la cámara dejará de seguirlo si sobre pasa estos límites.
- CM vcam1: La cámara virtual de *Cinemachine*. Aquí se indica que la cámara a la cual está asociada siga al jugador y se configuran otros parámetros. También tiene el componente *Confiner* para asignarle el "CameraBounds" explicado anteriormente.
- Coins: Todas las monedas del nivel.
- GameManager: Contiene el script del *GameManager* para controlar el juego.
- Baseballbat: Es el bate de béisbol.
- Player: *GameObject* del jugador, con todos sus componentes y que incluye los detectores del suelo, el punto de ataque y partículas para cuando es golpeado.
- MenuManager: Contiene referencias a los elementos de interfaz y realiza funciones sobre ellos.
- LevelManager: Se usa básicamente para el control del nivel.
- Enemies: Agrupación de todos los enemigos de la escena, es decir, las arañas del nivel.
- Collectables: Son los tres ítems coleccionables: pergamino, libro y olla.
- Decoration: Los objetos decorativos del nivel. Son los que se encuentran por el suelo como plantas, raíces, árboles, tumbas,… .
- Portal: El portal del final del nivel junto a sus sistemas de partículas.
- Colliders: Son los *colliders* para que el jugador no pueda desplazarse a la izquierda del nivel al inicio, otros para corregir errores del *collider* del *tilemap* y otro que es el límite de abajo para detectar al jugador cuando cae al vacío.
- AudioManager: Contiene los sonidos generales, como la música de fondo o los sonidos del personaje.

Estas dos escenas se encuentran en la carpeta "Scenes" dentro de "Assets". Además, dentro de "Assets" se tiene la siguiente estructura de carpetas:

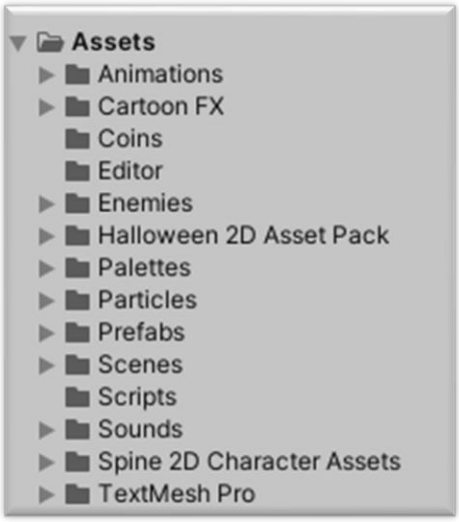


Figura 33. Estructura carpeta "Assets"

- **Animations.** Contiene las animaciones y controles de animación del personaje principal y de las monedas.
- **Cartoon FX.** Asset con los sistemas de partículas.
- **Coins.** Imágenes de las animaciones de las monedas y su *prefab* animado.
- **Enemies:** Contiene todo lo referente a los enemigos, en este caso la araña.
- **Halloween 2D Asset Pack.** Asset usado para la creación gráfica del nivel, con *sprites*, fuentes, animaciones y paletas de *tiles*.
- **Palettes.** Algunas paletas de tiles personalizadas.
- **Prefabs.** Objetos prefabricados de enemigos, plataformas e interfaz. Los *prefabs* son un método eficaz para poder reutilizar los *assets* y tener mayor organización.
- **Scenes.** Contiene las escenas explicadas previamente.
- **Scripts.** Son el conjunto de *scripts* usados en el proyecto.
- **Sounds.** Todos los archivos de música y efectos de sonidos, además del *Audio Mixer*.
- **Spine 2D Character Assets.** Asset del personaje protagonista que incluye todas las animaciones para las diferentes armas.
- **TextMesh Pro.** Referente a la herramienta de TextMesh Pro, para textos sofisticados.

4.6 IA de los enemigos

Los enemigos de este nivel son una especie de arañas que se mueven andando por el terreno de un punto a otro. Su comportamiento está definido en el script "Enemy.cs". El movimiento nunca se detiene si el jugador no interfiere, de manera que, si la araña llega al punto A, se girará dirección al punto B. Estos dos puntos son *GameObjects* vacíos, cuyas posiciones son leídas por código.

Figura 34. Puntos y recorrido del enemigo

Sus condiciones de movimiento son dos: el enemigo se moverá si no está atacando y si está recuperado al haber recibido un ataque del jugador. Si una de estas condiciones no se cumple, el enemigo no se moverá.

Para detectar al jugador se le ha creado un *Box Collider 2D*. Así pues, si el jugador colisiona con el enemigo, este se encarará hacia el jugador, aunque esté de espaldas y, si está recuperado de algún ataque previo, le atacará. Mientras el jugador esté en modo invulnerable tras el ataque, el enemigo seguirá andando sin atacarle.

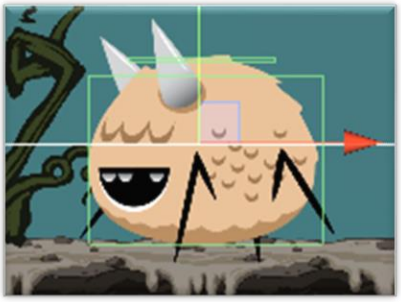


Figura 35. Colliders del enemigo

Las arañas también tienen un segundo colisionador en su parte superior, para detectar cuando el jugador les salta encima y morir. Cuando son atacadas, se cambia el color de su *Sprite Renderer* a uno más rojizo para mostrar que están dañadas. También tienen sistemas de partículas que se reproducen cuando sufren daño o al morir.

Por último, cabe destacar que hay funciones que se usan en eventos de animación. Así, mediante la animación, es posible saber cuándo el enemigo se tambalea o cuando inicia y finaliza el ataque.

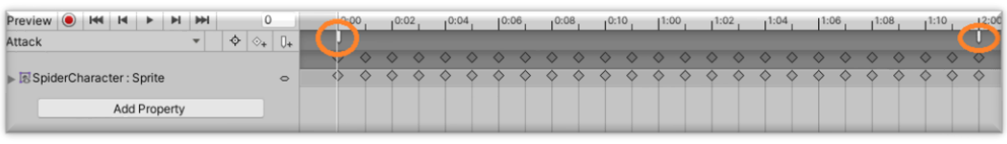
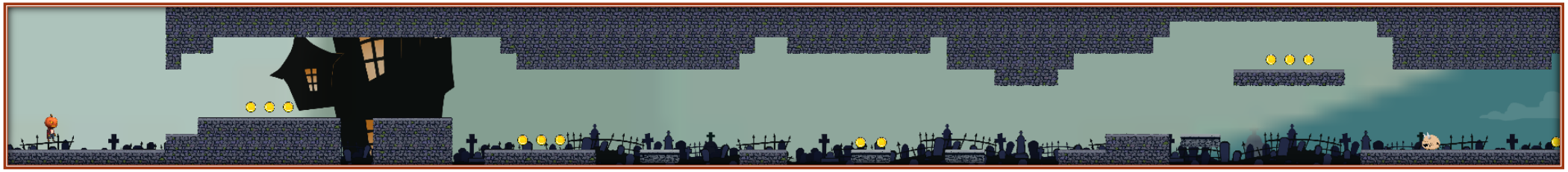


Figura 36. Eventos de animación para el ataque del enemigo

5. Diseño de niveles

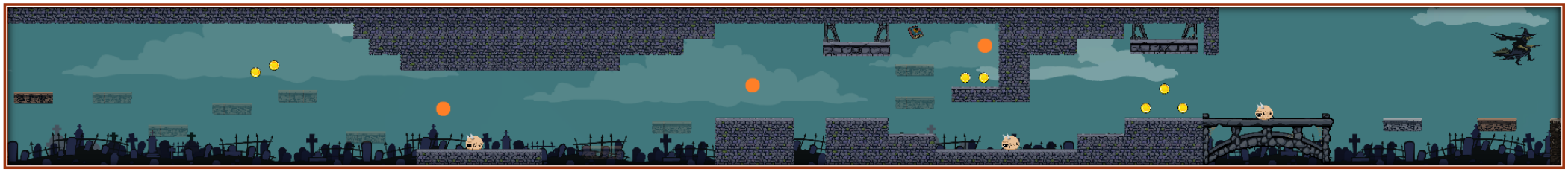
5.1 Mapa del nivel



El avance del nivel es hacia la derecha. Al principio aparecen monedas para coleccionar, plataformas móviles y el primer enemigo.

Empieza una parte del nivel de tierra y aparece una plataforma que se moverá cuando el jugador se le suba encima. También hay varios enemigos, seguidos de una plataforma fantasma que se tendrá que tocar para que desaparezca y obtener el pergamino.

Después de saltar por varias plataformas fantasma y otras en movimiento, el jugador se encontrará una plataforma aplastante. Deberá de calcular para pasar por debajo suyo sin ser aplastado. También podrá adquirir el bate de béisbol para aumentar su capacidad de ataque.



Al principio de la sección las plataformas fantasmas irán apareciendo conforme el jugador pise la plataforma anterior. Los puntos naranjas son activadores de plataformas fantasmas. Solo es necesario que el jugador toque ese punto representado por una luz blanca. De esta manera, se podran hacer aparecer las plataformas que conducen al segundo ítem coleccionable del nivel: el libro.

En esta parte del nivel el jugador deberá de ser rápido. Aquí se encuentra una plataforma que iniciará su movimiento una vez el jugador se suba encima de ella. La plataforma no se detendrá, de manera que el jugador deberá de ir cruzando por las porciones de terreno y saltar de nuevo encima de la plataforma hasta la siguiente porción de terreno.

Una vez se suba en el terreno, si se toca la zona marcada con un punto naranja, aparecerá una plataforma fantasma a la izquierda que permitirá llegar al tercer coleccionable: la olla. Luego el jugador, rápidamente, deberá cruzar el terreno hacía la derecha y tomar de nuevo la plataforma, la cual se dentendrá definitivamente al avanzar un poco más. Para pasar a la siguiente parte, tan solo hará falta subir en la plataforma en movimiento vertical.

Finalmente, al cruzar por dos plataformas aplastantes, un par de otras en movimiento y algún que otro enemigo, se llegará al portal final. Al tocarlo, el nivel será superado.

5.1 Criterios de diseño

El nivel se ha generado teniendo un cuenta una curva de dificultad que fuese de menor a mayor. De esta manera, al principio aparecen solo plataformas en movimiento sencillas.

Figura 37. Criterios de diseño. Plataformas en movimiento

El primer enemigo se presenta aislado para que el jugador pueda interactuar con él sin tener en cuenta otros factores.

Figura 38. Criterios de diseño. Primer enemigo aislado

Luego aparece una plataforma que se mueve solo si el jugador se le sube encima. Al ser lo unico que tiene en frente, el jugador se le subirá sí o sí y verá que la plataforma solo se mueve una vez hacía delante.

Figura 39. Criterios de diseño. Plataforma que se mueve con el jugador

Al avanzar, aparece una plataforma fantasma, la cual no presenta ningún riesgo para el jugador ya que al desaparecer, an solo le abre camino para el pergamino. Este objeto también está encima de una plataforma de ese tipo, por lo que el jugador al haber visto el comportamiento de la anterior podrá tener en cuenta que estas plataformas desaparecen.

Figura 40. Criterios de diseño. Primeras plataformas fantasmas

Más tarde el juegador se encuentra con una plataforma aplastante. Su movimiento y el sonido ya dejan claro cuál es su peligro. Seguidamente, el bate de beisbol se encuetra en medio del camino, casi obligandole al jugador a cogerlo. El bate se encuentra en este punto, ya que el jugador ha tenido oportunidades de atacar con los puños y ahora notará una mejora de ataque. Además esto permite poder añadir mas enemigos.

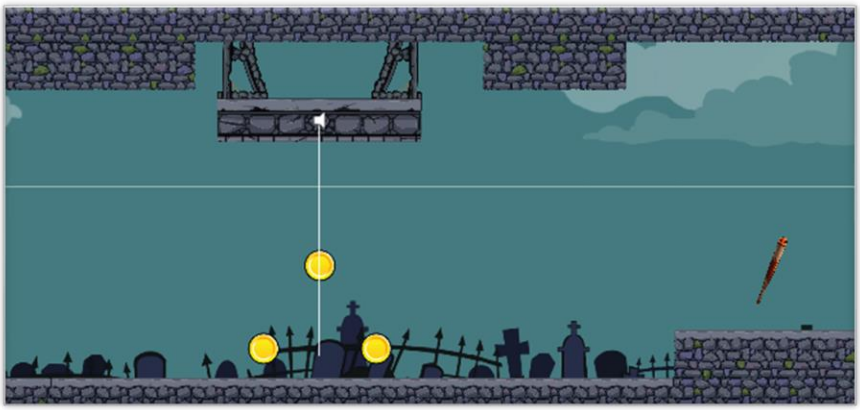


Figura 41. Criterios de diseño. Plataforma aplastante y bate de béisbol

La dificultad aumenta poniendo a prueba los reflejos del jugador. Se encuentran las plataformas fantasmas que son invisibles al inicio y que se irán volviendo visibles a medida que el jugador avanza por las plataformas anteriores. Luego hay un terreno con un enemigo fácil de eliminar. Aquí el jugador no puede avanzar, salvo que se fije en la luz que ilumina el activador de la plataforma fantasma que le permita seguir por el camino. De esta manera está obligado a interactuar con la luz.

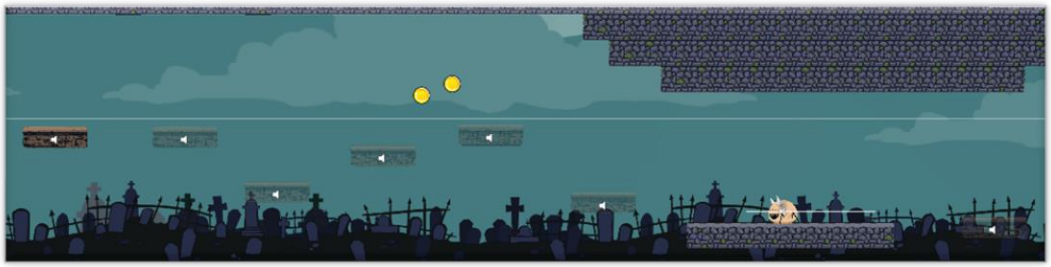


Figura 42. Criterios de diseño. Plataformas fantasmas y zona del destello de activación

En el siguiente punto, aparece el segundo ítem coleccionable flotando en el aire. Sin embargo, previamente el jugador habrá aprendido que las luces parpadeantes hacen aparecer plataformas. Así pues, tocando un par de ellas se logran activar las plataformas para alcanzar el libro.

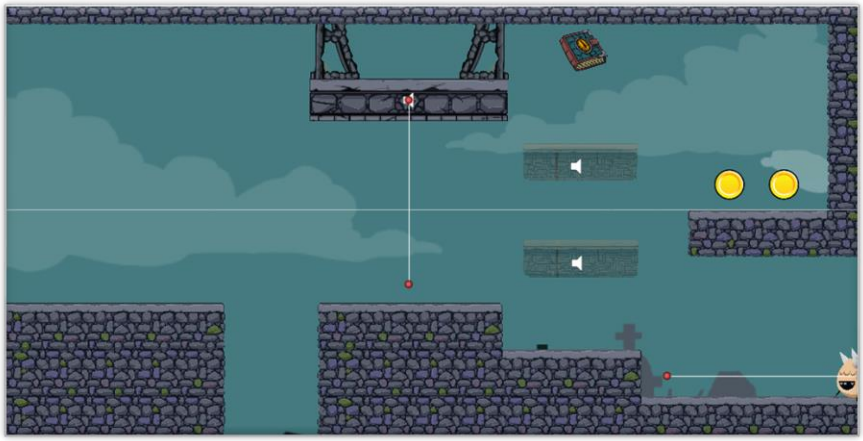


Figura 43. Criterios de diseño. Plataformas fantasmas y libro coleccionable

En esta parte del nivel aparece una plataforma como la que apareció anteriormente. Aquí se pone a prueba la velocidad del jugador y aparece el tercer ítem coleccionable que solo podrá conseguirse si se es rápido.

Figura 44. Criterios de diseño. Última plataforma móvil con el jugador

En este punto, el jugador deberá ver la olla para tomar la dirección correcta. La plataforma fantasma que le permite de acceder es muy fácil de activar. Si este trayecto se hace sin pausas, el ítem puede ser obtenido regresando a la plataforma en movimiento.

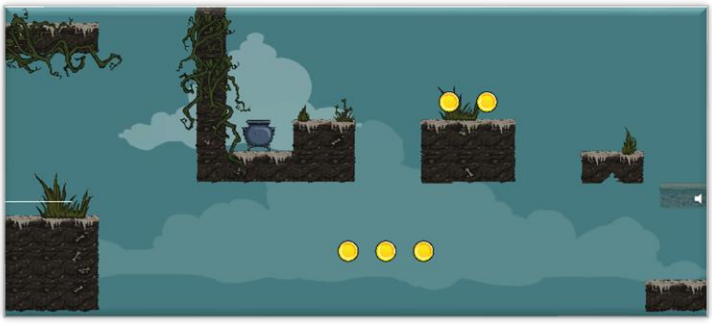


Figura 45. Criterios de diseño. Tercer Ítem coleccionable

Después de este acelerón llega el final del nivel. Todo lo que aparece ya se vio antes. Enemigos, plataformas aplastantes y otras en movimiento. Al final, el camino está bloqueado y solo está el portal. Así pues, es una señal obvia que indica al jugador que debe interactuar con él.

Figura 46. Criterios de diseño. Portal Final

6. Manual de usuario

6.1 Requerimientos técnicos de hardware

Los requerimientos técnicos son los mismos que ofrece Unity. Como el videojuego está destinado a PC, estos se indicarán para esta plataforma.

Requisitos Mínimos	Windows	macOS	Linux
Sistema Operativo	Windows 7 (SP1+) y Windows 10, 64-bit	Sierra 10.12.6+	Ubuntu 16.04, Ubuntu 18.04, y CentOS 7
CPU	Arquitectura x64 con set de instrucciones SSE2	Arquitectura x64 con set de instrucciones SSE2	Arquitectura x64 con set de instrucciones SSE2
GPU	DX10, DX11 y DX12	Intel y AMD con Metal	Nvidia y AMD con OpenGL 3.2+ o Vulkan

- Almacenamiento en disco duro: 2,5GB.

6.2 Instrucciones de juego

El videojuego se abre mediante el archivo ejecutable "Pumpkin Hero", haciendo doble clic sobre él. Luego de una pequeña transición con el logo de Unity, se accede al menú principal de juego.

Figura 47. Menú principal

El menú tiene tres opciones:

- **Play:** Es el botón para iniciar el juego. Al hacer clic sobre él se inicia automáticamente el nivel de juego actual.
- **Settings:** Se utiliza para abrir el menú de configuraciones de sonido. Este menú está compuesto por tres secciones con *sliders* que sirven

para aumentar o disminuir el volumen del juego. La primera sección es para el volumen general, de manera que, si se desliza el botón de la barra manteniendo el clic izquierdo del ratón presionado, se modificará el volumen global. La segunda sección regula el volumen de la música, por ejemplo, la del menú principal o la del nivel de juego. Por último, la tercera sección modifica el volumen de los efectos de sonido, como los sonidos del personaje en el nivel, los enemigos o las plataformas. Finalmente, el botón *Back* sirve para regresar al menú principal aplicando los cambios.

Figura 48. Menú de sonido

- **Exit:** Botón para salir y cerrar el juego.

Al entrar al nivel mediante la opción de *Play*, el nivel cargará y el jugador ya será capaz de controlar al personaje e iniciar el recorrido. Además, dentro del nivel aparece una interfaz con tres bloques clave:

Figura 49. Interfaz del nivel

- **Vida del jugador:** Se trata de una barra de color verde que se va vaciando a medida que el jugador pierde vida. Si la barra verde está completa significa que el jugador tiene toda la vida. Contrariamente, si la barra verde se vacía del todo, el jugador pierde el juego.
- **Monedas adquiridas**: Un contador que marca el número de monedas que el jugador ha logrado conseguir. Estas monedas se encuentran por el nivel y basta con tocarlas para aumentar el contador en 1 por cada moneda obtenida.
- **Ítems adquiridos**: Indica los ítems coleccionables que el jugador ha sido capaz de recolectar. Por defecto, aparecen tres imágenes, una para cada objeto. Una vez el jugador colisione con un ítem en concreto, la imagen de la interfaz correspondiente se iluminará indicando así que el ítem ha sido obtenido.

Los controles del videojuego son bastante sencillos:

- **Movimiento del jugador:** Las teclas "A" y "D" son las que se utilizan para mover el personaje en las distintas direcciones. Si se pulsa "A" el personaje se mueve hacia la izquierda y si se pulsa "D", hacia la derecha.
- **Salto del personaje:** Al pulsar la barra espaciadora del teclado el personaje salta.
- **Ataque:** Clic izquierdo del ratón para realizar el ataque, ya sea con los puños o con el bate de beisbol.
- **Activar/Desactivar menú de pausa:** Mediante la tecla "Escape" se entra en el menú de pausa y una vez allí, con la misma tecla se sale de la pausa.

Al abrir el menú de pausa, el flujo del juego se detiene. También aparece el menú de pausa centrado en pantalla con las siguientes opciones:

Figura 50. Menú de pausa

- **Resume:** Del mismo modo que al pulsar la tecla "Escape", este botón permite reanudar el juego y seguir des del punto des de donde había sido pausado.

- **Settings:** Igual que en la opción del menú principal, sirve para configurar el audio mediante los *sliders* de audio general, música y sonido.
- **Exit:** Para salir del nivel y regresar al menú principal. Al darle a esta opción se abre un nuevo menú preguntando si de verdad se desea salir del nivel. Si así es, se debe pulsar el botón de *tic* verde abajo a la izquierda. En caso contrario es necesario pulsar el botón de cruz roja abajo a la derecha.

Figura 51. Menú de pausa, confirmación de Exit

Una vez el jugador pierde ya sea porque su vida llega a 0 o porque cae al vació, aparece el menú de derrota.

Figura 52. Menú de derrota

En este menú aparecen los siguientes elementos a destacar:

- **Número de monedas:** Es el número de monedas que el jugador ha logrado conseguir hasta el punto en el que perdió el juego.
- **Ítems coleccionados:** Tres imágenes correspondientes a los tres objetos coleccionables dentro del nivel. Por defecto solo se

muestran las siluetas en gris, queriendo decir que el ítem con esta apariencia no ha sido adquirido. Si el ítem ha sido coleccionado en el momento de perder el juego, se muestra la imagen en color de este objeto en vez de su silueta en gris.

- **Replay:** Botón que permite reiniciar el juego des del inicio, reiniciando también monedas y objetos coleccionados.
- **Exit:** Botón para salir al menú principal del juego.

Por otro lado, existe un menú de victoria al llegar al final del nivel y entrar al portal. Este menú tiene los mismos elementos que el menú de derrota, pero tiene una apariencia distinta.

Figura 53. Menú de victoria

7. Conclusiones

7.1 Lecciones aprendidas

Este proyecto supone mi segundo videojuego creado con Unity y el primero desarrollado en 2D. Por lo tanto, tenía un poco de experiencia con la herramienta, aunque sé que me queda mucho por aprender. Este Trabajo me ha permitido adquirir nuevos conocimientos en el desarrollo de videojuegos en Unity en el ámbito general y particularmente he podido estudiar más a fondo su metodología para el 2D. Por ejemplo, he trabajado con *tiles* para la generación del nivel, por lo que aprendí a generar distintas paletas y pintar sobre un sistema de rejillas o *grid* y un *tilemap*. Otra cosa que me gustó, fueron los eventos en las animaciones para llamar funciones de los *scripts* en un momento concreto de la animación. El sistema *parallax* de la cámara es otro aspecto que nunca toqué antes y que me sorprendió gratamente. Para detectar si el personaje está en el suelo se usaron *Raycast,* ya que es una forma efectiva de lograrlo y que tendré en cuenta para futuros proyectos. El audio es otro punto que creo que me salió bastante bien, de manera que me quedaré con este sistema de trabajo utilizado. Así pues, estos son algunos ejemplos de conocimientos que he adquirido a parte de algo muy importante como es la práctica desarrollando un videojuego. De esta manera se obtiene experiencia que te permite evitar ciertos errores en un futuro.

7.2 Análisis de objetivos planteados inicialmente

Al inicio del proyecto se establecieron ciertos objetivos. El principal, fue el hecho de tener un nivel de juego con sus menús, tanto el menú principal como el de pausa. En este sentido se puede decir que el objetivo se ha cumplido. Sin embargo, también se planificaron algunas características para el videojuego que no se han podido introducir, como es el caso de más equipamiento para el personaje o nuevos enemigos. Ciertamente, los objetivos iniciales fueron bastante optimistas, aunque en varios de ellos ya se contempló la idea de que sería difícil realizarlos. Por ejemplo, se contempló la idea de hacer una versión para móvil del juego si daba tiempo, ya que se trataba de un objetivo secundario. Así pues, la falta de experiencia o de algunos conocimientos provocaron algún que otro retraso en la planificación impidiendo cumplir algunos de los objetivos. Referente al tema de enemigos, ha resultado difícil obtener unos *sprites* animados gratuitos (ya que el presupuesto ya se alcanzó) con ciertas características, además de tener el tiempo en contra. Por último, existe un objetivo a nivel personal que recae en tener un proyecto hecho para añadir a mi portfolio de trabajos y poder enseñarlo a la hora de buscar faena.

7.3 Análisis de la metodología y seguimiento de la planificación

La planificación se cumplió bastante bien des de un inicio, pero al final tuvieron que haber modificaciones. Debido a la falta de experiencia, surgieron problemas técnicos o dudas que tomaron más tiempo del

deseado en resolverse. Aun así, este tipo de tropiezos hacen aprender para poder dar soluciones en un futuro. Por otro lado, el Trabajo se ha realizado en época de *covid19,* lo significa que en mi trabajo laboral he estado remotamente des de mi casa, aunque durante un tiempo he tenido que ir presencialmente. Si bien es cierto que mis tareas allí se basan en programación y termino bastante cansado, uno de los problemas que tuve era el no haber tenido en cuenta que trabajaría presencialmente, ya que el viaje (y más en transporte público) consume mucho tiempo. Además, también se tuvo una mudanza entre medio y falta de internet. No obstante, por suerte siempre se ha tenido un hueco para dedicarle al proyecto ya que personalmente es algo que me interesa, sobre todo por lo que me puede aportar.

7.4 Temas pendientes y trabajo futuro

En el apartado de objetivos se comentaron ejemplos de alguno que no podrieron se cumplidos. Obviamente, un tema pendiente es el cumplimiento de dichos objetivos. Se considera que el primero que debe de tratarse es el tema de los enemigos, es decir, añadir al menos una nueva especie de enemigo y, a poder ser, con ataque a distancia. Otro aspecto donde se le dedicaría un tiempo, es en la creación de un *checkpoint* o un punto a medio nivel de reaparición en caso del que el jugador pierda. Por último, otro elemento destacable es el tema de la iluminación, donde se probaría de añadir luces para crear zonas oscuras, brillos o luces tenebrosas para favorecer la ambientación. Una vez hecho esto, lo siguiente podría ser una adaptación para dispositivos móviles. Y por si faltara faena… siempre se pueden crear nuevos niveles de juego y un sistema de tienda.

7.5 Pruebas con usuarios

Se han realizado pruebas con usurarios, lo que ha permitido identificar, sobre todo, algunas partes del nivel que eran algo complicadas o no quedaban del todo claras. Un punto clave es cuando el jugador tiene que activar plataformas tocando un destello parpadeante. Aquí se ha notado una dificultad, ya que el jugador, o bien no se percataba del destello o bien se pensaba que formaba parte de la decoración. Como consecuencia, el destello se ha hecho más visible, ya que parpadea más veces y, por lo tanto, se muestra más tiempo. También se le ha dado un toque rojizo para que no se confunda con las estrellas blancas del fondo. Otra zona difícil del nivel es a la hora de conseguir el tercer ítem coleccionable, es decir, la olla. En el primer intento, el jugador es bastante incapaz de obtener el ítem ya que no sabe cómo reaccionar. Sin embargo, posteriormente sí es capaz de obtenerlo. De hecho, la intención en el diseño era esa, la de obligar al jugador a intentar varias veces el nivel para conseguir los ítems como en *Súper Mario Bros.*

8. Glosario

- **Asset:** Un asset de Unity es un elemento que puedes usar en tu juego o Proyecto. Un asset puede provenir de un archivo creado fuera de Unity, como un modelo 3D, un archivo de audio, una imagen o cualquiera de los otros tipos de archivo compatibles con Unity.
- **Canvas:** En Unity, es el área donde todos los elementos UI deben estar.
- **CheckPoint:** Punto del nivel que, al alcanzarlo, permite al jugador reaparecer en él si pierde el juego.
- **Clase:** En programación orientada a objetos, se trata de una plantilla para la creación de objetos de datos según un modelo predefinido.
- **Collider:** Componente de Unity que define un área con la que se pueden detectar colisiones.
- **CPU:** En inglés, "Central Processing Unit". Es la parte de la computadora en la que se encuentran los elementos que sirven para procesar datos.
- **Drag and Drop:** Significa arrastrar y soltar.
- **Función.** En programación, es un conjunto de líneas de código que realizan una tarea específica y puede retornar un valor.
- **Game Engine:** Significa "Motor de videojuegos", el cual hace referencia a una serie de rutinas de programación que permiten el diseño, la creación y funcionamiento de un videojuego.
- **GameObject:** Objeto base de Unity al cual se le pueden añadir componentes para dotarlo de ciertas funcionalidades.
- **Git:** Sistema de control de versiones open source.
- **GPU:** En inglés, "Graphics Processing Unit". Es la parte de la computadora que se encarga de procesar todos los gráficos.
- **Prefab:** Objeto reutilizable en Unity creado a partir de un *GameObject* cualquiera.
- **Script:** En Unity se trata de un pequeño programa que se puede asociar a un *GameObject* como si se tratara de un componente más, proporcionándole una funcionalidad concreta.
- **Slider:** Es un deslizador que permite al usuario establecer un valor numérico de un rango predeterminado, arrastrando el ratón.
- **Sprite:** Se tratan de objetos 2D, como imágenes de elementos o texturas.
- **Sprite Renderer:** Componente en Unity que permite mostrar imágenes como *Sprites* para su uso en escenas 2D y 3D.
- **Tiles:** Assets organizados en *Tilemaps* para construir un escenario 2D. Cada *Tile* hace referencia a un *Sprite* seleccionado, el cual es renderizado en una localización de la rejilla del *Tilemap*.
- **Tilemap:** Componente que contiene y almacena *Tiles* para crear niveles 2D.
- **UI:** En inglés, "User Interface" o en español, "Interfaz de Usuario", es la vista que permite a un usuario interactuar de manera efectiva con el sistema.

9. Bibliografía

1. Videojuegos de plataformas (23/12/2020): https://es.wikipedia.org/wiki/Videojuego_de_plataformas

2. Super Mario Bros (27/09/2020): https://sites.google.com/site/bananagamereviews/home/super-mario-bros-nintendo

3. Super Mario Bros. U (27/09/2020): https://www.vsgamers.es/en/product/new-super-mario-bros-u-deluxe-nintendo-swith-game

4. Rayman Legends (27/09/2020): https://www.gamersyde.com/video_rayman_legends_ps4_gameplay_2-31538_en.html

5. Oddmar (27/09/2020): https://oddmar.uptodown.com/android

6. GameMaker (27/09/2020): https://www.mcvuk.com/business-news/yoyo-games-launches-major-gamemaker-studio-2-update-allowing-for-a-more-visual-approach/

7. Stencyl (27/09/2020): https://www.vidaextra.com/listas/si-quieres-hacer-tus-propios-juegos-estos-son-los-mejores-motores-que-vas-a-encontrar

8. Unity (27/09/2020): https://unity.com/es/how-to/beginner/unity-good-2d-development

9. Unreal Engine (27/09/2020): http://www.geekpro.es/unreal-engine-4-y-el-fotorrealismo-que-puede-lograr/

10. Motores de videojuegos (27/09/2020):
    - https://docs.hektorprofe.net/escueladevideojuegos/articulos/engines-motores-recopilacion-programas/
    - https://www.vidaextra.com/listas/si-quieres-hacer-tus-propios-juegos-estos-son-los-mejores-motores-que-vas-a-encontrar

11. Características de Unity (25/12/2020):
    - https://academiaandroid.com/motor-de-juegos-unity-3d/
    - https://www.google.com/amp/s/openwebinars.net/amp/blog/que-es-unity/

12. Requisitos técnicos de Unity (28/12/2020):
    - https://docs.unity3d.com/es/current/Manual/system-requirements.html