Universitat Oberta de Catalunya

uoc.edu

Ninja Adventure | Creación de un juego de plataformas 2D

Aitor Tena Torrecilla
Máster en Diseño y Desarrollo de Videojuegos
Diseño, creación y multimedia

Helio Tejedor Navarro
Joan Arnedo Moreno

Junio 2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Ninja Adventure | Creación de un juego de plataformas 2D*
Nombre del autor:	*Aitor Tena Torrecilla*
Nombre del consultor/a:	*Helio Tejedor Navarro*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (MM/AAAA):	06/2020
Titulación:	*Máster en Diseño y Desarrollo de Videojuegos*
Área del Trabajo Final:	*Diseño, creación y multimedia*
Idioma del trabajo:	*Castellano*
Palabras clave:	*Videojuego, Plataformas, 2D*

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

Este proyecto se ha llevado a cabo dentro de la asignatura final del máster como proyecto para superar la docencia. La finalidad de este proyecto es la creación de un videojuego de plataformas 2D ambientado en el mundo ninja y con los elementos distintivos de este género de juegos (saltos, desafíos, exploración, enemigos, pruebas, recolección de ítems, …).

Haciendo uso del motor de desarrollo Unity, a lo largo del proyecto se han implementado los diferentes elementos que conforman este videojuego ninja de aventuras en 2D como es un personaje principal, enemigos, niveles, ítems para recolectar, desafíos, etc. así como las mecánicas e interacciones entre los elementos que constituyen el juego.

El resultado del proyecto es una primera versión de este videojuego de plataformas 2D donde se puede controlar a un ninja que debe superar cuatro niveles repletos de desafíos, saltos, peligros y enemigos. Además, el cuarto nivel implementa un *boss* con diferentes ataques y fases que no se lo pondrá nada fácil al jugador para derrotarlo.

Al final, este videojuego sigue la línea clásica de los juegos del mismo género, implementando un personaje principal y un conjunto de niveles, con sus respectivos elementos y dificultades, que el jugador debe superar y desbloquear. Por otro lado, este videojuego puede enfocarse al modelo comercial *freemium*, dando al jugador contenido básico para jugar y contenido extra para conseguir ventajas, *skins*, vidas extra, etc. Actualizaciones en el videojuego pueden ir incorporando nuevos contenidos para mantener al usuario jugando a menudo.

Abstract (in English, 250 words or less):

This project is the outcome of the work done during the final semester in the master's degree. The aim of the project is to create a 2D platform videogame set in a world of ninjas that has several distinctive elements of this gaming genre (jumping, challenges, exploration, enemies, trials, collectable items, …)

Using Unity engine as the main development tool, during the project many different game elements have been implemented altogether in order to build the player, the levels, the enemies, the items and ultimately the videogame itself.

The result of the project is a first version of the 2D platform videogame including four levels where you play as a female ninja that has to clear levels in order to unlock more levels and go forward in the videogame. The fourth level implements a boss fight where the player will have to defeat a powerful enemy that has many attacks and behaviors. It will not be an easy challenge.

Finally, this videogame represents a classic version in its genre, implementing a main character, a set of different levels and challenges that the player must overcome in order to go forward. An option for this videogame comes in the commercial area as a freemium videogame, where the player gets basic content and extra content like benefits, skins, extra lives, etc. Updates can keep more content coming in as to get the user playing more often.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Como parte final del máster en diseño y desarrollos de videojuegos es necesario crear un juego para demostrar los conocimientos adquiridos y asimilados durante los cursos y asignaturas del máster. El reto que se plantea es el de poder crear un videojuego de cualquier género dentro del tiempo disponible en el semestre.

El proyecto culminará en la creación de un juego que debe ser un producto lo más acabo posible, teniendo en cuenta las limitaciones temporales y de recursos. El videojuego deberá contar con los elementos necesarios para su funcionamiento, experiencia y jugabilidad y como mínimo un nivel lo más detallado posible que incorpore y demuestre todo el trabajo realizado.

1.2 Objetivos del Trabajo

- Definición del género del videojuego.
- Definición de los elementos más relevantes que constituirán el videojuego (personaje principal, enemigos, ítems, interacciones, ...).
- Definición de las mecánicas a implementar.
- Elección de las herramientas más adecuadas para llevar a cabo la creación del videojuego.
- Elección de las plataformas donde irá destinado el videojuego.
- Creación o adaptación de los recursos necesarios para el videojuego como pueden recursos gráficos, de sonido, *scripts*, etc.
- Implementación de las mecánicas.
- Implementación de los elementos del juego.
- Implementación de menús.
- Implementación de la UI.
- Implementación de escenas y niveles del videojuego.
- Pruebas unitarias y globales, corrección de errores.
- Creación de una versión alfa del videojuego.
- Versión final del videojuego como producto ejecutable en al menos una plataforma de destino.

1.3 Enfoque y método seguido

Para poder cumplir con los requisitos del proyecto, es indispensable elaborar un producto nuevo, es decir, crear un videojuego nuevo. A partir de aquí hay bastante libertad a la hora de construir el juego ya sea utilizando motores de desarrollo creados por terceros o hacerlo uno mismo, crear recursos o buscar recursos ya existentes (gratuitos o de pago *royalty-free*), implementar la lógica a partir de ejemplos y adaptarla

al videojuego o crearla desde cero si procede y así con todos los aspectos del proyecto.

Se ha decidido usar un motor de terceros, Unity, el cual es libre de uso y provee de múltiples herramientas para crear e implementar todos los elementos que conforman este videojuego. En el apartado gráfico y sonoro se ha optado por hacer uso de recursos gratuitos y disponibles *online* que sean de uso libre (no comercial). Para implementar las mecánicas, lógica e interacciones de los elementos del juego se ha hecho uso del conocimiento y experiencia tanto de lo aprendido en el máster como en el ámbito personal.

Al final, un videojuego es un producto complejo en el cual se unen diferentes conocimientos y tipos de recursos, los cuales todos combinados crean el producto y experiencia final. Dependiendo del tipo de juego a crear, es muy difícil que una persona pueda crear todos los recursos desde cero, es por eso que se ha optado por hacer uso de recursos libres en el apartado del motor de desarrollo, gráfico y sonoro y construir todos los elementos de código, lógica y mecánicas uno mismo. Estos últimos recursos son el "pegamento" que hace que todos los demás elementos funcionen combinados y den como resultado un producto donde todas las partes están integradas y funcionan en conjunto para crear la experiencia del videojuego.

1.4 Planificación del Trabajo

El proyecto se ha planificado en base a las tareas principales y el tiempo disponible en el semestre. Para acometer la creación del juego el único recurso disponible es el autor del mismo, por lo tanto, solo una persona trabajará en el proyecto y se encargará de todas las tareas y planificaciones.

En los próximos capítulos se detallarán con más precisión las tareas y sus planificaciones dentro del proyecto. Por el momento a modo global, aquí se detallan las PEC y los hitos que marcan:

- PEC1 – 3 semanas – Diseño del videojuego
- PEC2 – 5 semanas – Versión parcial
- PEC3 – 7 semanas – Versión jugable
- PEC Final – 2 semanas – Versión final

1.5 Breve sumario de productos obtenidos

A la finalización del proyecto, se ha obtenido principalmente el videojuego en su versión final ejecutable (en al menos una plataforma).

Por un lado, a lo largo de la elaboración del proyecto se han obtenido diferentes versiones parciales del juego, así como una versión alfa previa a la versión final del videojuego.

Por otro lado, también se han generado recursos propios, sobre todo *scripts* que ejecutan la lógica del comportamiento del juego y que describen las uniones e interacciones de los elementos que lo conforman.

1.6 Breve descripción de los otros capítulos de la memoria

Capitulo 1. Introducción

Comentarios previos a temas generales del proyecto. Contexto y justificación del trabajo, los objetivos planteados, metodología de trabajo, planificación general y pequeña descripción de los productos obtenidos del proyecto.

Capítulo 2. Estado del arte

Breve resumen a la historia y evolución de los juegos de plataformas para dar un poco de contexto al tipo de juego creado en el proyecto.

Capítulo 3. Definición del juego

Características y temas más importantes en la definición del videojuego a desarrollar. Descripción, ambientación, interacciones y mecánicas, plataformas de destino, diseños y planificación de tareas. Este capítulo sienta las bases en las que se desarrollará el juego.

Capítulo 4. Diseño técnico

En este capítulo se entra en más detalle técnico de como se ha diseñado y construido el juego. Se selecciona la plataforma de desarrollo, las herramientas utilizadas, los recursos que se usan en el videojuego, la arquitectura y la construcción de los componentes y sus relaciones para lograr crear el juego.

Capítulo 5. Diseño de niveles

Guía de niveles. Criterios de diseño y construcción, así como los elementos que los forman.

Capítulo 6. Manual de usuario

Información de los requerimientos técnicos e instrucciones para jugar. Descripción de los menús del juego y controles.

Capítulo 7. Conclusiones

Conclusiones del proyecto. Crítica de los objetivos, planificación y metodología. Comentarios sobre posibles futuros desarrollos o evoluciones del videojuego que no se hayan podido trabajar en el proyecto actual.

Capítulo 8. Glosario

Definición de los términos y acrónimos más relevantes utilizados dentro de la memoria.

Capítulo 9. Bibliografía

Lista con los recursos bibliográficos empleados en la elaboración de la memoria.

2. Estado del arte

2.1 Historia y evolución de los juegos de plataformas

Si se analiza la cantidad de géneros de juegos que existen actualmente probablemente se obtendría una lista enorme que crece con cada generación de juegos que se lanzan al mercado. Entre todos los géneros, hay uno que desde sus inicios ha marcado un estilo propio que ha llegado a muchas generaciones de jugadores y que ha sabido reinventarse a lo largo del tiempo para seguir ofreciendo entretenimiento y diversión: los juegos de plataformas.

Prácticamente, todas las plataformas y generaciones de consolas han podido disfrutar de títulos de este género y algunos han marcado un antes y un después en la historia de los videojuegos por su fama y por como los jugadores han disfrutado jugándolos. Los juegos de plataformas han sufrido una evolución a la par que los tiempos y la tecnología y han sabido proporcionar nuevas formas de jugarlos, incorporar nuevos elementos y adaptarse a todo tipo de dispositivos.

En los siguientes párrafos se va a resumir el camino y la evolución que ha visto este género de videojuegos, junto con las plataformas donde han sido jugados, desde sus inicios hasta la época actual para poder dar una visión del estado del arte de este tipo de juegos.

Los videojuegos de plataformas han evolucionado a través de diferentes “eras” donde cada una ha ido incorporando nuevos elementos al género y ha hecho crecer tanto su popularidad como la cantidad de títulos disponibles. Empezando en la década de los 80, juegos como Space Panic o Donkey Kong contaban con pantallas estáticas donde el jugador solo podía moverse (de forma vertical normalmente) dentro del nivel renderizado por pantalla. Al llegar a los límites de la pantalla no se avanzaba a otras zonas del nivel ya que el nivel era únicamente la pantalla. Más títulos fueron llegando al mercado como Mario Bros de Nintendo o Pitfall! de Activision, este último siendo bastante innovador al introducir varias pantallas conectadas unas con otras para lo que sería el principio del desplazamiento dentro de los niveles (aunque debido aun al límite de potencia de los dispositivos disponibles la innovación no resaltó mucho en aquellos tiempos).

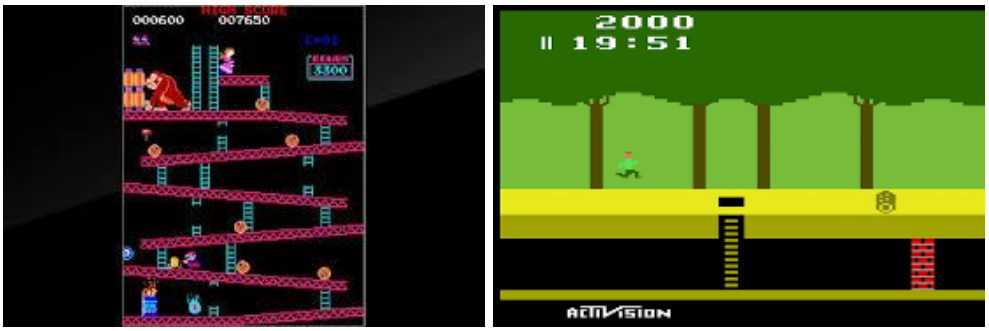

Figura 1 – Izquierda imagen de Donkey Kong (raspberrypi.org) | Derecha imagen de Pitfall! (pinterest.com)

No se tardó mucho tiempo en superar esta limitación y títulos como Jump Bug y Quest for Tires (ColecoVision) empezaron a incluir el desplazamiento horizontal en el género de plataformas. Otros títulos introdujeron efectos visuales como el *parallax* (Pac-Land en 1984), que añadían profundidad a los niveles, y que se extenderían a los demás juegos del mismo género. Esta era será recordada por el nacimiento de uno de los personajes y videojuegos más famosos que jamás hayan existido, Super Mario Bros para la consola de sobremesa NES. Además de niveles con desplazamiento horizontal, este juego introducía muchos otros elementos que se convertirían en iconos del género.

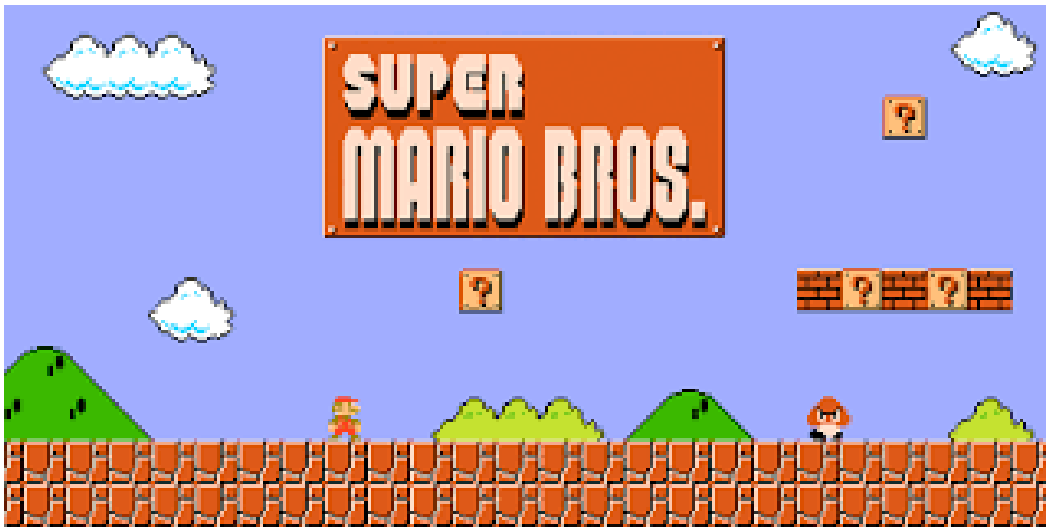

Figura 2 – Super Mario Bros en 1985 (nintendo.es)

A medida que los dispositivos sobre los que se ejecutaban los juegos de plataformas se volvían más potentes (se llega ya en este punto a los 16-bits), los desarrolladores empezaron a explorar y explotar estas nuevas características para introducir mejoras en los juegos. El éxito de Super Mario despertó a otra grande en aquel entonces de la industria de los videojuegos, Sega. Consciente de que necesitaba un estandarte en el género, Sega creó otro de los personajes y juegos más famosos de la historia como es Sonic.

Los juegos empezaron a introducir niveles cada vez más largos y elaborados, efectos visuales mejorados y *sprites* más detallados alcanzando un nuevo nivel en el género. Por otro lado, en esta época

empiezan a aparecer títulos para otra plataforma no nacida para la ejecución de juegos como son los PC. La compañía id Software lanzó al mercado el juego llamado Commander Keen, uno de los primeros juegos para esta plataforma que introducía desplazamiento lateral en los niveles.

La era de los 16-bits ha dejado joyas del género de plataformas como Mega Man X (Capcom), Donkey Kong Country (Nintendo), Castelvania Symphony of the Night, Super Metroid, Rayman, etc. y más secuelas de los populares Super Mario y Sonic.

Figura 3 – Izquierda Mega Man X (nintendo.es) | Derecha Castelvania Symphony of the Night (mobygames.com)

Llegados a este punto un nuevo salto gráfico dentro de los videojuegos estaba irrumpiendo de forma rápida en todos los géneros. El 3D estaba empezando a aparecer en escena aprovechándose de las capacidades mejoradas de las plataformas de videojuegos.

Aun así, algunos títulos intentaron un acercamiento intermedio, el 2.5D, para no perder la esencia de lo que hasta ahora habían significado los juegos de plataformas y "engañar" astutamente a los jugadores. Dando la sensación de juegos más modernos con gráficos 3D este tipo de videojuegos seguían desarrollándose en una vista 2D y al introducir gráficos con una dimensión extra se acuño el término 2.5D. Aquí se encuentran títulos como Clockwork Night, Pandemonium y Klonoa.

Por el lado contrario al acercamiento anterior se encuentran títulos con una perspectiva 3D pero con gráficos 2D. Entre los representantes de este tipo de juegos se encuentran títulos como Antartic Adventure (Konami) y Congo Bongo (Sega).

La década de los 90 supuso el avance del 3D en los videojuegos donde comenzó a ganar más y más terreno y títulos inesperados como Crash Bandicoot fueron auténticos éxitos. Este último, sin ser gráficamente sorprendente, introdujo nuevos elementos, mecánicas y formas de juego que fueron una revolución en aquel momento.

Figura 4 – Portada del juego de PS1 de Crash Bandicoot (pinterest.com)

Entrados ya en los juegos plenamente con gráficos 3D y consolas que llegaban hasta los 64-bits, en esta década se encuentran juegos como Super Mario 64, Banjo-Kazooie, Donkey Kong 64 y Tomb Raider, siendo este último un juego innovador al introducir elementos de *shooters* con resolución de puzles, todos estos elementos integrados con el género de plataformas.

Estrenando nuevo milenio, una nueva generación de juegos de plataformas 3D vieron la luz. Títulos como Sly Cooper, Jak and Daxter o Ratchet and Clank inauguraron una nueva de los videojuegos de plataformas no solo llevando el género a un nuevo nivel sino creando o diversificando este género en nuevos subgéneros dentro de las plataformas. Estos juegos introdujeron nuevas mecánicas y enfoques en la experiencia de juego tal que, aunque las plataformas seguían siendo el género principal, ya no se los podía considerar solo pertenecientes a este género, sería simplificar demasiado. Esta época en los videojuegos no podría haber sido posible sin las consolas como Playstation 2, DreamCast, Sega Saturn o GameCube que acogieron cada una grandes títulos del género.

Figura 5 – Izquierda Ratchet andClank (heypoorplayer.com) | Derecha Jak and Daxter (forbes.com)

Avanzando en la primera década del segundo milenio y con la llegada de consolas de nueva generación como PlayStation 3, los juegos de plataformas siguieron reinventándose y mezclándose con otros géneros para dejar títulos como Mirror's Edge (apostando por la primera persona), Super Mario Galaxy con todos sus niveles y jugabilidad, Uncharted de Naughty Dog, que tras dejar atrás el estilo de Crash

Bandicoot, creó una nueva franquicia de videojuegos donde mezcló las plataformas con la acción y una narrativa cinemática o Little Big Planet que daba la posibilidad al jugador de crear sus propias experiencias. En esta era, el término juego triple A o AAA se empezaba a poner de moda para los juegos con grandes presupuesto y muy buenos acabados tanto en gráficos como en jugabilidad y experiencia. Los juegos de plataformas no podían ser menos, y, aunque mezclados con otros géneros, consiguieron reinventarse de nuevo y seguir siendo un género atractivo, rompedor y con muchas ventas.

Figura 6 – Juego Uncharted para PS3 (hobbyconsolas.com)

Mezclados en la era de los juegos AAA, una nueva generación de desarrolladores *indie* empezó a resucitar el espíritu inicial de los juegos de plataformas creando juegos nuevamente en 2D. Títulos como Braid, Fez, Super Meat Boy y Limbo consiguieron popularizar los juegos *indie*.

Llegando ya a los últimos años hasta la época actual, algunos grandes como Nintendo han intentado reinventarse volviendo a sus orígenes lanzando juegos de sus famosas franquicias con gráficos 3D pero con estilo 2D. Los desarrolladores *indie* han seguido generando títulos como Owlboy, Hollow Knight, Axiom Verge, Giana Sisters, Inside, Spelunky o Cuphead que han ganado mucha popularidad por su reinvención del género y cierta vuelta a sus orígenes.

Figura 7 – Hollow Knight (kickstarter.com)

Por otro lado, la incursión de los dispositivos móviles desde los primeros *smartphones* hasta los dispositivos actuales con grandes prestaciones, la expansión y apertura de motores de desarrollo de videojuegos y la creciente industria y demanda de contenido relacionado con los videojuegos ha desembocado en un sinfín de títulos desde simples hasta más elaborados para este tipo de plataformas.

A partir de aquí es difícil poder predecir como seguirán evolucionando este tipo de juegos, pero una cosa es segura, si algo han demostrado los juegos de plataformas es su versatilidad, jugabilidad, innovación, reinvención y mezcla con otros tipos de géneros de videojuegos así que es fácil de suponer que sea en 2D, 2.5D, 3D o 4D, volviendo a sus orígenes o evolucionando con nuevos elementos, los juegos de plataformas van a seguir siendo un género que se podrá disfrutar por mucho tiempo y seguirá trayendo diversión a todos los que quieran jugar y embarcarse en sus aventuras [1] [2] [3] [4] [5] [6] [7].

2.2 Plataformas de desarrollo

Como se ha mencionado en el anterior apartado sobre la historia y evolución de los juegos de plataformas, este género ha estado presente prácticamente desde las primeras consolas y PCs que salieron al mercado.

Conforme la tecnología ha ido evolucionando también lo ha hecho la forma en la que los videojuegos se crean, desde los inicios donde todo el juego iba codificado en el mismo código fuente hasta las herramientas más modernas utilizadas hoy en día donde existen motores de desarrollo que se nutren de muchas otras herramientas de diseño de personajes, efectos, niveles, etc.

Los juegos de plataformas han ido adaptándose a los tiempos y todo tipo de herramientas de diseño de videojuegos han sido utilizadas para crearlos. Desde herramientas específicas para cada plataforma hasta otras que ofrecen más variedad, el género de plataformas ha podido crearse prácticamente con cualquier opción disponible.

En la época actual con la generalización y apertura de ciertas herramientas de diseño como Unity o Unreal Engine entre muchos otros softwares, prácticamente cualquier persona puede crear un juego de plataformas desde lo más simple y básico hasta toda la complejidad que se le quiera dar. La tecnología actual permite que la mayoría de plataformas puedan convertirse en un lugar de desarrollo de videojuegos y poder lanzar las creaciones a la gran variedad de dispositivos que hay actualmente en el mercado.

Se puede observar que las plataformas de desarrollo han ido pasando de ser algo muy específico para dispositivos en concreto hasta ser más generales y cambiar totalmente la forma de diseñar videojuegos. Todos

los géneros se han adaptado de una forma u otra a las tecnologías disponibles y el de plataformas no se ha quedado al margen. Este género ha sabido adaptarse a la evolución de las plataformas de desarrollo y ha ido adaptando todos los elementos del desarrollo a las nuevas tecnologías [9] [10].

3. Definición del juego

3.1 Breve descripción del juego

Este es un juego de plataformas 2D ambientado en un mundo ninja y oriental en el que el jugador controla una aprendiz de ninja que desea convertirse en maestro ninja y proteger su clan. Para ello, el jugador deberá ir superando nivel tras nivel para ir desbloqueando nuevas fases del juego haciendo uso de las habilidades y técnicas ninja con las que cuenta el personaje principal. En su aventura el jugador encontrará retos, peligros y enemigos que no le pondrán las cosas fáciles, pero le ayudarán en su deseo de convertirse en el maestro ninja más habilidoso de su clan.

3.2 Subgénero y referencias a juegos existentes

Dentro del género de “Plataformas” este juego se emplaza en los “Plataformas 2D” clásicos, desarrollándose únicamente en dos dimensiones.

Las referencias e inspiraciones para este proyecto surgen de juegos como: *Crash Bandicoot*, *Super Mario Bros*, *Metal Slug*, *OddWorld*, *Medievil*, *Wild 9* o *Ratchet & Clank* entre muchos otros. Estos juegos clásicos, aunque no todos 2D, reúnen lo mejor de una época en el terreno de los juegos de plataformas donde los jugadores debían poner a prueba sus habilidades saltando por precipicios, eliminando enemigos, aprendiendo nuevas habilidades, tanto físicas como mágicas o realizando carreras contrarreloj entre muchas otras situaciones dentro de los juegos y dentro de la experiencia que querían ofrecer.

Los juegos de plataformas tienen varias características en común:

- El juego se estructura por niveles bien diferenciados, con principio y fin únicos.
- Existe normalmente un personaje principal que es el centro de la historia, aunque a veces pueden desbloquearse nuevos personajes.
- El personaje principal debe superar los niveles para avanzar en la historia, sea cual sea el motivo ulterior para llegar hasta el final.
- Suele haber un antagonista o razón superior por la que el personaje principal se ve obligado a superar los niveles (derrotar al antagonista, salvar el mundo, …)
- El personaje principal cuenta con diferentes habilidades que se derivan de las mecánicas del juego y a en algunos videojuegos puede aprender nuevas o mejorar las que ya tiene.

- Los niveles presentan retos que se deben superar con las habilidades. Los más comunes son salvar distancias, derrotar enemigos o esquivar trampas.
- Los niveles cuentan con ítems que el personaje principal puede recolectar y que le pueden otorgar alguna ventaja, como vidas extra, inmortalidad momentánea, …
- Puede haber “jefes” intermedios en la historia que el personaje debe derrotar para avanzar.
- En ocasiones, este tipo de juegos incorporan algún modo extra o misiones secundarias dentro de los niveles.

Estos juegos y su género son los que inspiran el desarrollo de este proyecto donde se espera que el jugador pueda sentir la emoción de la aventura y disfrutar de un rato entretenido.

3.3 Tipo de interacción juego jugador

El jugador dispone de diversas maneras para poder interaccionar con el personaje y superar los retos que se le presentan en los diferentes niveles.

Las interacciones vienen definidas por las mecánicas principales del juego y las básicas que se quieren implementar son:

- **Salto simple**: esta mecánica permite al jugador salvar distancias cortas entre huecos, subir pequeños desniveles o saltar peligros.
- **Salto doble**: es una evolución directa de la habilidad anterior. Incluida en muchos juegos de plataformas, el doble salto aporta un extra en la experiencia de juego con la cual el jugador puede tener un mejor control de los saltos ya sea para superar espacios entre plataformas, luchar contra enemigos o llegar a diversos lugares en el nivel. Esta mecánica solo es accesible una vez se ha iniciado el salto simple, el jugador podrá decidir si ejecutar el doble salto o no.
- **Planear**: esta habilidad permite al jugador planear cuando no está en contacto con el suelo, de forma que puede controlar mejor el salto o salvar un poco más de distancia que solo con un salto simple. Además, el jugador puede usar el planeo para interactuar con las zonas de viento. Esta mecánica añade jugabilidad en situaciones y retos con las que el jugador se puede encontrar, ya sea poniéndolo a prueba en situaciones aéreas o para esquivar peligros con dificultad añadida.
- **Deslizamiento**: con esta habilidad el jugador puede hacer que el personaje pase por zonas estrechas, ataques enemigos o coja velocidad para realizar un salto. Esta mecánica proporciona al jugador más habilidades para combinar unas con otras.
- **Trepar**: la habilidad para trepar puede servir para poder conectar diversas partes del nivel entre sí. Esta mecánica favorece la transición entre diferentes partes en los niveles y puede dar juego para llegar a zonas difíciles con recompensas.

- **Atacar**: para poder derrotar a los enemigos que se interpongan en el camino, el jugador necesitará esta habilidad para poder atacarlos. Básicamente se trata de un ataque a corta distancia con la *katana* del personaje para herir a los enemigos.
- **Lanzamiento de kunai**: con esta habilidad el personaje puede lanzar un kunai (arma tipo cuchillo japonesa) para poder matar a los enemigos. Para poder lanzar estos cuchillos habrá que recogerlos a lo largo del nivel y se lanzarán en la dirección en la que mire el personaje. Cuando el cuchillo impacte con un enemigo, el enemigo morirá y el cuchillo desaparecerá. Si el cuchillo impacta contra una superficie también desaparecerá.

Con estas interacciones se asegura que el juego tenga un mínimo de mecánicas que suelen incluir los juegos de plataformas 2D. Esto requiere como mínimo poder desplazarse en alguna dirección y saltar. Además, si existen enemigos hay que proporcionar alguna forma de eliminarlos con ataques o habilidades. Desafortunadamente, la mecánica de trepar y la habilidad de lanzar un kunai no han podido ser implementadas durante el desarrollo del videojuego. Aun así, estas mecánicas pueden quedar pendientes para futuras actualizaciones y evoluciones del videojuego ya que entonan perfectamente con el mundo del juego y las demás interacciones.

A parte de las mecánicas básicas, se ha pensado en las siguientes:

- **Deslizamiento vertical en pared**: con esta mecánica el jugador puede clavar un kunai en la pared y “resbalar” por ésta de forma que pueda controlar la caída. Además, el jugador puede saltar a otra pared desde la que se encuentra y volver a repetir la habilidad. Como con otras interacciones juego-jugador, esta habilidad puede suponer retos a la hora de superar obstáculos del nivel y añadir una experiencia más de juego poniendo por ejemplo un tiempo límite de uso para la habilidad con recarga posterior.
- **Balanceo con cuerda**: esta habilidad puede ser interesante para añadir más experiencia de juego. La cuerda permite al jugador balancearse en ciertos puntos del nivel para llegar de un punto a otro.

Por otro lado, para dar un toque extra a las habilidades del jugador y así a la interacción con el juego, pueden existir habilidades especiales que desencadenen mecánicas únicas por tiempo limitado o en zonas concretas. Un ejemplo sería un “salto trepador” donde el jugador puede dar saltos en el aire como si estuviese entre dos paredes, rebotando entre una y otra y trepando al mismo tiempo. Con esta mecánica el jugador podrá acceder a ciertas zonas para superar partes del nivel. Estas habilidades, al ser más especiales, deben poder usarse menos frecuente y con ciertas limitaciones.

Estas últimas mecánicas quedan como un “TODO” para futuras versiones del juego ya que la implementación de las mecánicas básicas

con la carga de las otras partes del videojuego no ha permitido desarrollarlas (se ha focalizado el esfuerzo principalmente ha aquellas interacciones básicas definidas anteriormente y que son las más primordiales para tener en el juego). Realmente, son mecánicas que pueden enriquecer la jugabilidad, la experiencia del jugador y la diversidad del juego en cuanto a desafíos, situaciones o enemigos a derrotar, además de que son interacciones que van en la misma línea que las mencionadas anteriormente pero que aportan diferentes maneras de interactuar en el juego.

3.4 Plataforma de destino

Inicialmente, el videojuego está pensado para jugarse en PC debido a los recursos y tiempo disponibles para el trabajo. Aun así, sería fácilmente exportable a plataformas como consolas tipo PS4 o XBOX.

Tanto para PC como para consolas, el juego está pensado para jugarse con mando tipo PS3/PS4/XBOX. Si bien se puede jugar con teclado en PC, el mando aporta una mejor experiencia de juego ya que facilita el manejo del personaje y sus habilidades.

En cualquier caso, este proyecto se puede exportar también a plataformas móviles adaptando ciertas mecánicas que tienen en cuenta las limitaciones físicas de este tipo de dispositivos, así como la interacción juego-jugador.

3.5 Historia y ambientación

El mundo en el que se desarrolla el juego está ambientado en la estética ninja y asiática con toques fantásticos. En este mundo existen clanes ninja y los miembros entrenan para poder llegar a ser maestros y elevar su clan a la gloria. Además, los ninjas también luchan contra criaturas malignas que habitan y amenazan sus vidas.

El personaje principal es una aprendiz de ninja intrépida con un objetivo claro: llegar a ser el mejor maestro ninja de su clan y de todos los clanes. Para lograr su sueño, el joven ninja emprende un viaje en solitario para mejorar y perfeccionar sus habilidades, así como aprender otras nuevas. El viaje se desarrolla a través de diferentes niveles y escenarios, cada uno con sus particularidades, peligros y enemigos que supondrán cada vez un reto mayor a superar por parte del jugador.

A medida que se avanza y se superan los niveles, el jugador va descubriendo poco a poco la historia de este mundo, sus personajes, los enemigos y la verdad detrás de la amenaza que suponen estas criaturas. No solo las criaturas suponen un peligro, sino que la competencia entre clanes de ninjas puede llegar a enfrentamientos entre ellos. No todo serán enemigos en esta aventura, también se forjan lazos de amistad con otros personajes que ayudarán al joven ninja a seguir con su viaje.

3.6 Definición de los personajes y elementos del juego

Dentro del mundo del juego conviven diferentes personajes y elementos que interactúan entre sí, siendo los más relevantes los siguientes:

- **Personaje principal**: el personaje principal del juego es el ninja protagonista de la aventura. Un personaje femenino en un mundo principalmente concebido para ninjas masculinos. Aun así, este personaje no dejará de luchar para conseguir su propósito y sobresalir entre su clan y todos los demás. De carácter intrépido, valiente y tenaz, todo lo que conoce se pondrá a prueba a medida que avance en los niveles. Aunque en un principio el juego solo cuenta con este personaje, se pueden implementar algunos personajes más que se puedan desbloquear a lo largo del juego. Estos personajes pueden ser compañeros de aventura que en un principio son NPCs y no intervienen directamente en la jugabilidad. De esta forma, se puede conseguir "enganchar" un poco más al jugador haciendo que cumpla más misiones u objetivos dentro del juego para desbloquear los personajes y que luego pueda re-jugar los niveles con estos personajes, lo cual es una característica que incorporan diversos juegos de este género.
- **Enemigos**: son la contraparte al personaje principal. Estos seres malvados pueblan el mundo del juego y por ende los niveles que debe superar el personaje principal. Son las criaturas que debe derrotar el ninja con las habilidades que posee teniendo en cuenta como se comportan y atacan. En un principio estos seres son diversos tipos de monstruitos, pero también existen más ninjas y otros personajes con poderosas habilidades que se enfrentarán al protagonista de la historia (más reservados para "jefes" de nivel). Los enemigos pueden tener diferentes comportamientos y atacar cuerpo a cuerpo, a distancia o con habilidades mágicas.
- **Plataformas y trampas**: los enemigos no son los únicos elementos que suponen un peligro para el personaje principal. Los niveles cuentan con plataformas que hay que salvar para avanzar donde caer resulta en la muerte inmediata. Desniveles, plataformas móviles, saltos y más retos componen los niveles que debe superar el ninja y así entrenar sus habilidades. Añadiendo un poco más de dificultad, los niveles poseen diferentes trampas como enormes pinchos, lanzadores de bolas de fuego o péndulos mortales entre otros. Realmente a lo largo del juego se pueden ir incorporando más tipos de trampas y sus correspondientes combinaciones para crear cada vez más dificultad.
- **Ítems**: como característica distintiva de este tipo de juegos de plataformas, a lo largo de los niveles se pueden recolectar ítems, que en este juego son gemas de diferentes tipos. Las gemas aumentan la puntuación del nivel y existe una en particular que recupera vida al personaje para seguir enfrentándose a los peligros que tiene por delante.

- **Miscelánea**: existen otros elementos que aportan más interacción y experiencia de juego. Entre ellos se encuentran trampolines para conectar zonas de los niveles o zonas de viento en las que el jugador puede planear e interactuar con ellas para moverse por los niveles. A medida que se avanza en el juego más de estos tipos de elementos se pueden introducir para añadir variedad a la forma de interactuar con el entorne dentro de los niveles y también para añadir retos que supongan usar toda la habilidad del jugador para superarlos.

3.7 Interacción entre los actores del juego

La principal interacción del personaje principal será con los elementos de los niveles y los enemigos que pueda encontrar. Además, en algunas ocasiones donde se descubran piezas de la historia el jugador puede interaccionar con otros personajes ninja del mismo o diferente clan. Estos personajes secundarios pueden ampliar la información de la historia, ser una interacción amena para el jugador o dar información sobre nuevas habilidades.

Los jefes de nivel también añaden una interacción más elaborada dentro del juego ya que no son simples enemigos fáciles de derrotar. Pueden dar información sobre la trama del juego y desvelar hechos importantes.

La muerte del jugador es otro tipo de interacción con él mismo, ya que repercutirá en el avance del juego.

3.8 Objetivos planteados al jugador

El objetivo principal que se le plantea al jugador al inicio del juego es el de superar los niveles para entrenar, dominar sus habilidades y llegar a ser el mejor maestro ninja.

Dentro de los niveles el objetivo principal es llegar hasta el final del mismo recogiendo cuantas más gemas mejor. Estas gemas pueden ser equiparables a otros elementos similares de juegos de plataformas donde el jugador va recogiendo monedas, frutas u otros ítems a lo largo de los niveles.

Derivado del objetivo anterior, el jugador debe intentar no morir ya sea a manos de un enemigo o cayendo por los huecos del nivel para conservar cuantas más vidas mejor. Este objetivo puede ser interesante ya que depende como se plantee el ciclo vida-muerte del jugador puede ser un elemento clave del juego. En este sentido la mayoría de juegos de plataformas incorporan un sistema de vidas donde el jugador puede obtener más vidas recogiendo algún tipo de ítem. Cuantas más vidas, más oportunidades seguidas dispone el jugador para superar un reto. En los últimos tiempos se han visto otros modos de gestionar este ciclo de vida-muerte dando un número limitado de vidas y un tiempo para regenerarlas después de perderlas. Este modelo puede ser interesante

en este juego ya que abre las puertas a más tipos de interacciones o monetización. Aun así, se debe intentar imponer un poco de presión en este sentido para que el jugador no se relaje del todo pensando que puede obtener vidas fácilmente y que las puede derrochar, esto hará que el jugador se lo tome más en serio.

Además, dentro de cada nivel puede haber objetivos extras como llegar a ciertas zonas difíciles u ocultas, recoger una cantidad mínima de gemas, derrotar a ciertos enemigos sin morir, etc. Además, se incluye un modo contrarreloj en el cual el objetivo para el jugador será superar el nivel lo más rápido posible para conseguir el mejor tiempo.

3.9 Concept art

Las imágenes siguientes muestran un ejemplo de los recursos gráficos usados en la elaboración del proyecto y que sirven para tener una idea de los personajes y elementos descritos en los apartados anteriores. En capítulos posteriores se detallarán todos los recursos gráficos que han sido usados para crear el videojuego.

- Personaje principal

Figura 8 – Diferentes estados del personaje principal

- Enemigos

Figura 9 – Enemigos

Figura 10 – Jefe de nivel

- Niveles

Figura 11 – Conceptualización primer nivel

Figura 12 – Conceptualización segundo nivel

Figura 13 – Conceptualización tercer nivel

- Gemas

Figura 14 – Gemas para recolectar en los niveles

3.10 Planificación de tareas y cuantificación del tiempo/recursos

A continuación, se detallan la lista de tareas para llevar a cabo el proyecto de creación de este videojuego.

- Mecánicas
  - Jugador: desplazamiento
  - Jugador: salto
  - Jugador: salto doble
  - Jugador: planear
  - Jugador: deslizamiento
  - Jugador: trepar

    - Jugador: atacar
    - Jugador: lanzar kunai
    - Jugador: muerte
    - Jugador: reaparición
    - Jugador: interacción ítems
    - Jugador: cámara
    - Enemigo: comportamiento
    - Enemigo: ataque
    - Enemigo: muerte
    - Jefe: comportamiento
    - Jefe: ataque
    - Jefe: muerte
- Gráficos
    - Jugador: animación desplazamiento
    - Jugador: animación salto
    - Jugador: animación salto doble
    - Jugador: animación planear
    - Jugador: animación deslizar
    - Jugador: animación atacar
    - Jugador: animación trepar
    - Jugador: animación lanzar kunai
    - Jugador: animación muerte
    - Jugador: animación reaparición
    - Enemigo: animación comportamiento
    - Enemigo: animación ataque
    - Enemigo: animación muerte
    - Jefe: animación comportamiento
    - Jefe: animación ataque
    - Jefe: animación muerte
    - Ítems
- Niveles
    - Configuración elementos básicos
    - Configuración plataformas
    - Checkpoints
    - Configuración ítems
    - Configuración kunais
    - Configuración enemigos
    - Configuración reaparición jugador
    - Construcción nivel 1
    - Construcción nivel 2
    - Construcción nivel 3
    - Construcción nivel 4
- UI
    - Menú principal
    - Menú niveles
    - Menú opciones
    - Menú como jugar
    - Menú pausa
    - Configuración elementos gráficos
    - Configuración elementos pantalla de juego

- Efectos de sonido
    - Configuración efectos de sonido del juego
- Pruebas
    - Mecánicas
    - Niveles
    - UI
    - Ejecutable
    - Corrección errores
- Documentación y presentación del proyecto
    - Generación memoria proyecto
    - Preparación presentación/defensa del proyecto

El diagrama de Gantt adjunto aporta una visualización en forma de gráfico de tiempo y los recursos necesarios para las tareas descritas. De esta forma se obtiene una planificación a seguir durante la elaboración del proyecto donde se puede observar el estado del mismo, así como los avances o retrasos.

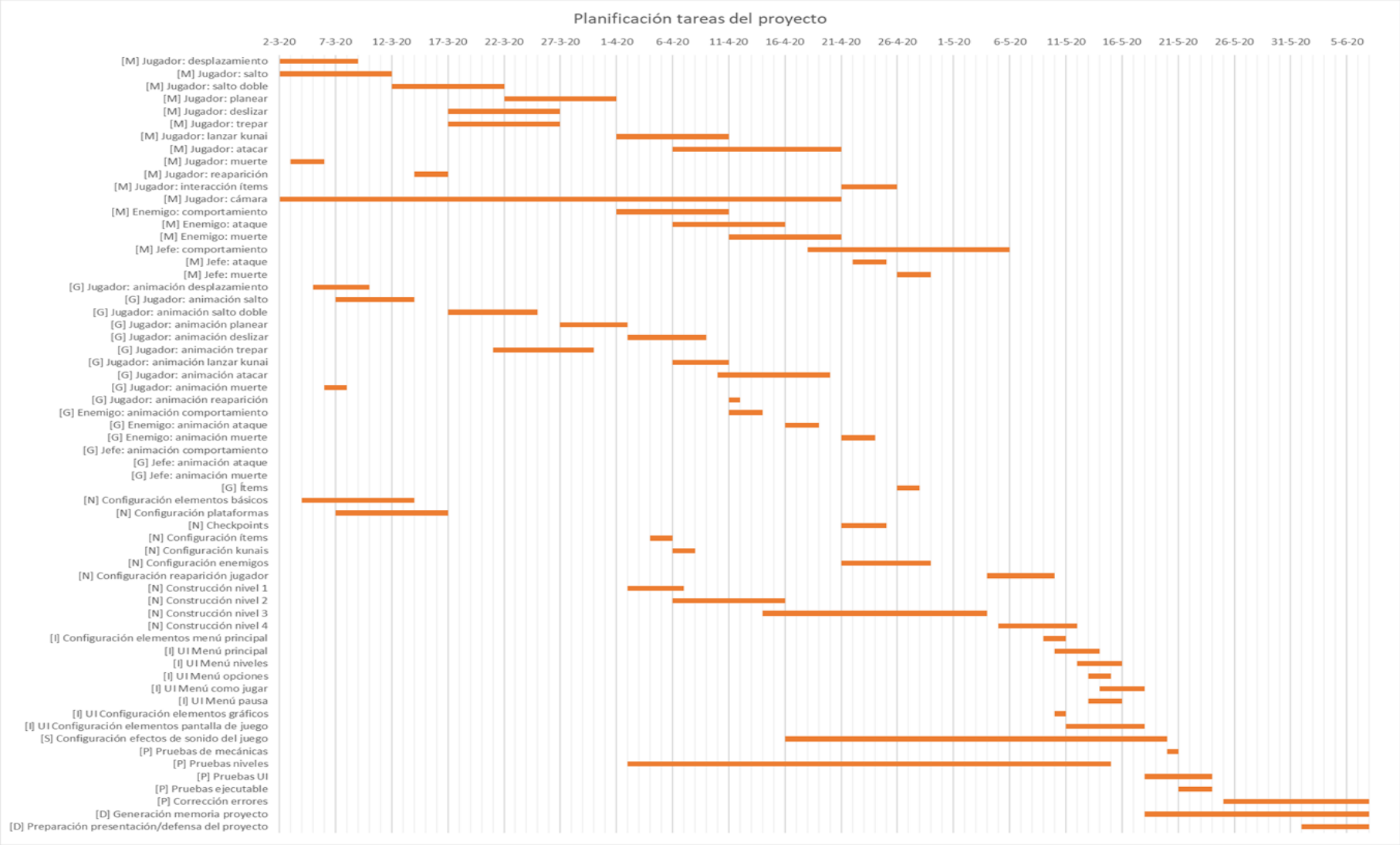

Figura 15 – Diagrama de Gantt del proyecto

El proyecto lo elabora una persona, así pues, este será el único recurso del que se dispondrá a lo largo de todo el desarrollo.

Como se puede observar, la planificación se ha basado en primero desarrollar las mecánicas básicas de desplazamiento o salto para luego pasar a las animaciones que interpretarán estas mecánicas. Al mismo tiempo, se empiezan a desarrollar diferentes elementos que conformarán los niveles.

En la fase media, se irán acabando y puliendo las mecánicas del jugador y los enemigos seguidas de sus correspondientes animaciones. Entre tanto, se iniciará la construcción de los niveles a medida que se van acabando de desarrollar los últimos aspectos de éstos. Una vez conformadas las mecánicas, animaciones y los elementos de los niveles se acabarán de construir y pulir los niveles que se presentarán en el proyecto. Al mismo tiempo se irán incorporando los efectos de sonido del juego.

Por último, se tiene una fase de pruebas y de corrección de errores que se puedan encontrar una vez se juntan todos los elementos que integran el juego. También se dispondrá de unos días para elaborar la memoria y la presentación del trabajo.

4. Diseño técnico

4.1 Entorno de desarrollo

Para poder evaluar que entorno de desarrollo se adecúa mejor a los requerimientos del proyecto, se enumeran primero las características más relevantes a tener en cuenta:

- Juego de plataformas 2D.
- Recursos visuales sencillos (texturas de calidad media), no se exige mucha carga a la renderización de cada *frame* y por consiguiente a la tarjeta gráfica del dispositivo.
- Mecánicas para juego de plataformas 2D y jugabilidad.
- Recursos limitados tanto de tiempo como de desarrollo, una persona para todas las tareas.
- Elección de un motor conocido para facilitar carga trabajo y ahorrar tiempo.
- Uso de un lenguaje de programación conocido y cómodo para el desarrollo del proyecto.
- Recursos materiales limitados para el desarrollo (limitación en el hardware).

Una vez valorados los puntos anteriores, el motor que más se ajusta a estos requerimientos es Unity. Primero, se descarta el uso de un motor de elaboración propia ya que el tiempo y recursos para poder lograr el desarrollo de este proyecto son limitados. Ciertamente, la elaboración de un motor propio sería un desarrollo muy interesante pero las limitaciones comentadas anteriormente descartan están opción. Para el desarrollo de un juego como el proyectado en este trabajo, donde se requieren mecánicas 2D y la carga de renderizado es ligera, un motor de uso libre como Unity podrá satisfacer las exigencias del proyecto. Tanto en el apartado de físicas, gráficos, elementos de sonido, IA o animaciones, este motor puede ofrecer gran variedad de herramientas y versatilidad que ayudan al desarrollo del videojuego.

Por otro lado, Unity es el motor que se ha usado durante los semestres anteriores del máster para llevar a cabo multitud de proyectos diversos, así pues, es un motor conocido y con el que se está acostumbrado a trabajar. Además, la carga que impondrá este motor a los recursos de hardware disponibles para el desarrollo del proyecto es asumible teniendo en cuenta los requisitos del juego.

Tras haber decidido cual va ser el motor de desarrollo para trabajar en el proyecto, la versión utilizada de Unity ha sido la 2017.2.2p3 siendo el entorno de desarrollo Mac OSX 10.13. A lo largo del proyecto este entorno ha sido más que suficiente para poder satisfacer los requisitos y las necesidades derivadas del uso del motor, tanto el *debug*, compilado

y ejecución de Unity y del videojuego. Por otro lado, .NET 3.5 es la versión utilizada para los *scripts* en tiempo de ejecución, así como el uso de Mono para el *backend* de los *scripts* y .NET 2.0 para la API de compatibilidad.

4.2 Otras herramientas empleadas en el desarrollo

Durante a elaboración del proyecto no se ha usado únicamente el motor Unity para construir el videojuego. La siguiente lista enumera las herramientas más relevantes que también se han utilizado para realizar el juego.

- GitHub

Para llevar a cabo el control de versiones de los diferentes desarrollos e implementaciones se ha usado git y su software de escritorio GitHub Desktop. Como con todas las herramientas e interfaces de control de versiones basadas en git, este software permite trabajar con ramas, hacer commit, pull, push, merge, etc. La metodología seguida para llevar un orden sobre las ramas del proyecto ha sido “gitflow”, con una rama principal master, una rama develop y ramas feature para implementar los desarrollos.

- Visual Studio 2019 for Mac

El IDE para la programación de scripts ha sido la versión de Microsoft para Mac OS de Visual Studio. Junto a su complemento para Unity, este IDE se integra perfectamente con el motor para poder llevar a cabo la programación de scripts en lenguaje C#.

- Gimp

Algunas imágenes o *sprites* usados en el proyecto han necesitado algún retoque básico antes de poder ser usados en el videojuego. Esta herramienta gratuita de edición de imágenes es la que se ha usado para tales propósitos.

- Herramientas online

Algunas herramientas disponibles gratuitamente online también se han utilizado para para algunas tareas del proyecto. Es el caso de audiotrimmer.com que permite editar con algunas funcionalidades básicas archivos de audio o ezgif.com y lunapic.com que también permiten la edición de imágenes con múltiples opciones a través de sus páginas web.

4.3 Inventario de recursos gráficos

En esta sección se enumeran todos los recursos gráficos que se usan en el videojuego. Todos estos recursos han sido seleccionados de la red y

son libres de uso. Únicamente algún recurso gráfico se ha retocado mínimamente con las herramientas mencionadas en la sección anterior, siendo estos retoques cambios de tamaño, recortes de secciones o borrado de fondos de la imagen.

Estos recursos gráficos se encuentran dentro de la carpeta principal del proyecto en la ruta Assets/Sprites y sus correspondientes subcarpetas para mantener un orden:

- **Animated Half Saw**: una sierra semienterrada en el rail que la contiene y que se puede desplazar de un lado a otro.
- **Animated Springboard**: trampolín animado en forma de muelle y plataforma de color rojo.
- **Animated Waterfall**: *tiles* animados para representar una cascada en movimiento.
- **Autumn Tileset**: esta carpeta contiene los *tiles* y *sprites* usados para crear el segundo nivel del juego.
- **Crystal Gems**: aquí se encuentran diferentes imágenes que representan gemas de cristal, algunas de ellas usadas para construir los ítems en forma de gema que se pueden recolectar en los niveles.
- **Enemies**: aquí se encuentran los *sprites* utilizados para crear los enemigos de los niveles (no jefes de nivel). En las diferentes subcarpetas se encuentran la serie de imágenes que se usan para crear las animaciones de los diferentes estados (correr, atacar, quieto, …).
- **Fireball**: bola de fuego que se utilizada como trampa en los niveles.
- **Graveyard Tileset**: esta carpeta contiene los *tiles* y *sprites* usados para crear el tercer nivel del juego.
- **Healthbar**: recursos usados para crear la barra de vida del personaje principal.
- **Nature Tileset**: esta carpeta contiene los *tiles* y *sprites* usados para crear el primer nivel del juego.
- **NinjaGirl**: todos los *sprites* utilizados para crear las animaciones del personaje principal.
- **Nuts**: imagen de una rosca usada en alguna de las trampas.
- **Ropes Chains**: imágenes de cuerdas y cadenas usadas en algunas trampas.
- **Sakura**: *sprite* del árbol de sakura usado para representar los puntos de control. También se encuentran imágenes de pétalos que se han usado para algún efecto de partículas.
- **SamuraiHeavy**: *sprites* utilizados para crear las animaciones del jefe del nivel cuatro.
- **Sandclock**: imágenes que contienen los *sprites* para realizar la animación del reloj de arena que aparece junto al marcador de tiempo en el modo contrarreloj de los niveles.
- **Signals**: imágenes utilizadas en algún nivel para marcar dirección o peligro.

- **Spikes**: estacas/pinchos metálicos que aparecen en los niveles sobretodo en los huecos y agujeros.
- **Temple**: imagen utilizada para representar el final del nivel.
- **TimeTrialClock**: imagen usada para representar el reloj que inicia la prueba contrarreloj en los niveles.
- **UI**: contiene todas las imágenes utilizadas en la interfaz gráfica del juego.
- **Wind Effect**: serie de *sprites* usados en la animación de una ráfaga de viento para este tipo de zonas.

Existe otra carpeta a parte llamada Backgrounds dentro de Assets que contiene los fondos que se han usado para los niveles.

La lista siguiente contiene los sitios web desde donde se han descargado los recursos gráficos mencionados arriba:

- assetstore.unity.com
- craftpix.net
- freepgn.es
- gameart2d.com
- gamedeveloperstudio.com
- itch.io
- opengameart.org
- pngfuel.com

4.4 Inventario de recursos sonoros

Seguidamente se pasan a listar los recursos de audio/sonido que se han usado en la elaboración del proyecto. Al igual que con los recursos gráficos, todos los audios o sonidos han seleccionados en diferentes webs y son libres de usar en proyectos como este. Algunos sonidos han sido recortados en su duración para encajar mejor.

Los ficheros con estos recursos se encuentran en la siguiente ruta dentro del proyecto Assets/Resources/Audio&Sounds:

- **Audio niveles**: los cuatro niveles disponibles en el juego contienen música de fondo.
- **Sonidos botones UI**: los botones que se usan en la UI del juego tienen efectos de sonido cuando se seleccionan o clican.
- **Sonido bola de fuego**: las bolas de fuego escupidas por una trampa tienen un efecto de sonido.
- **Sonido trampolín**: el trampolín tiene emite un sonido cuando el jugador salta sobre él.
- **Sonido reloj**: cuando el jugador coge el reloj de la prueba de tiempo se escucha un sonido.
- **Sonidos enemigos**: los enemigos que se encuentran en los niveles disponen de diferentes sonidos cuando atacan, son heridos o mueren.

- **Sonido gemas**: cuando el jugador coge alguna gema esta emite un sonido. Cada gema tiene su propio sonido.
- **Sonido punto de control**: cuando el jugador llega a un punto de control se emite un sonido para avisar al jugador.
- **Sonido final de nivel**: cuando se supera un nivel se emite un sonido para avisar al jugador.
- **Sonidos personaje principal**: el personaje principal emite sonidos cuando salta, ataca, se desliza, planea, es golpeado o muere.
- **Sonidos jefe de nivel**: el jefe del cuarto nivel emite sonidos al atacar, ser golpeado y morir.
- **Sonido kunai**: los kunais del cuarto nivel emiten un sonido tipo flecha cuando se lanzan.
- **Sonido cascada**: las cascadas del primer nivel emiten su sonido característico.

Los audios y los sonidos han sido obtenidos de las siguientes webs:

- findsounds.com
- freesound.org
- itch.io
- opengameart.org
- orangefreesounds.com
- zapsplat.com

4.5 Inventario de scripts

Los *scripts* albergan la lógica del comportamiento e interacciones de todos los elementos del juego. Dentro del proyecto se organizan en sus respectivas subcarpetas en la ruta Assets/Scripts. Todos los *scripts* han sido desarrollados personalmente.

Se pasan a enumerar en la siguiente lista los *scripts* del proyecto, aunque se hará referencia a ellos y se explicarán las características más relevantes de cada uno en próximas secciones:

- AnimatedSpringboard
- AudioMixerController
- AudioPauser
- BackgroundParallax
- BossDefeatedBanner
- BossEnd
- CameraShake
- CharacterAudio
- CharacterComponents
- CharacterController2D
- CharacterEvents
- CharacterFlags
- CharacterHealth
- CharacterInputController2D
- CharacterParams
- CharacterStateGrounded
- CharacterStateNotGrounded
- CheckPoint
- CheckPointBanner
- ControlFlags
- CosClampOscillatorFunction
- CosOscillatorFunction
- CosSqrtbOscillatorFunction
- CosStrikeOscillatorFunction
- CrystalGem
- DeathRespawnBlackPanel
- EBossFases
- EEnemyStates
- EGameScenes
- EnemyPointToPointAttacking
- Fireball
- FireballHole
- GameManager
- GameManagerStateHowToPlay
- GameManagerStateIntro
- GameManagerStateLevel
- GameManagerStateLevelLoad
- GameManagerStateLevelsMenu

- GameManagerStateMainMenu
- GameManagerStateSettingsMenu
- GameManagerStateTestLevel
- GlideImpulseZoneController
- HealthBar
- HowToPlayAttackEnemy
- HowToPlayEnemyAttackEnemy
- HowToPlayNinjaAttackEnemy
- ICharacterStateMachine
- IEnemy
- IGameController
- IGameManagerState
- IntroScene
- IUISelectable
- KeyboardController
- KeyboardGameController
- KunaiMainMenuFX
- LevelCompletedBanner
- LevelEnd
- LevelLoad
- LevelScoreCounter
- LoadKunaiFX
- MainMenuFXSpawner
- MovingPlatform
- NinjaGirlMainMenuFX
- Oscillator
- OscillatorFunction
- PS3Controller
- PS4Controller
- PSPController
- PS3GameController
- PS4GameController
- PSPGameController
- RotatingHalfSaw
- SamuraiHeavyBoss
- SamuraiHeavyBossKunai
- SinHalfPICosbOscillatorFunction
- SinOscillatorFunction
- StaticLandSpike
- SwingingAnimSpringboard
- SwingingSpike
- TimeTrial
- TimeTrialCompletedBanner
- UIButton
- UIHowToPlayMenu
- UILevelsMenu
- UIMainMenu
- UINavigation
- UIPauseMenu
- UISettingsMenu
- UIToggle

Todos los *scripts* están escritos en C#.

4.6 Otros recursos

Aparte de los recursos de los puntos anteriores, se han utilizado otros tipos más para el videojuego.

- Cinemachine

Recurso descargado de la tienda de Unity para un mejor control de la cámara del juego.

- Fuentes

Se ha utilizado la fuente “gangofthree” en formato .ttf para los textos del videojuego. Esta fuente se puede encontrar dafont.com.

- Materiales

Se ha generado personalmente algunos materiales (physics materials 2D) para ciertos elementos del juego.

- Prefabs

A medida que se han ido implementando diferentes elementos del juego algunos se han pasado a prefab. En la ruta Assets/Prefabs se pueden encontrar los diferentes prefabs que se han construido personalmente para usarlos dentro del juego.

A parte, Unity ha generado otros prefabs y recursos relacionados con los pinceles y *tiles* que se usan para construir los niveles (tilemaps).

4.7 Arquitectura del juego

Una vez se han descrito los principales recursos que se han usado en el proyecto, toca avanzar a otros aspectos fundamentales del juego: como está construido y que componentes forman el videojuego.

El proyecto cuenta con varias escenas entre las que se puede pasar dependiendo de la relación que hay entre ellas. El siguiente esquema muestra los flujos entre las escenas que puede recorrer el jugador:

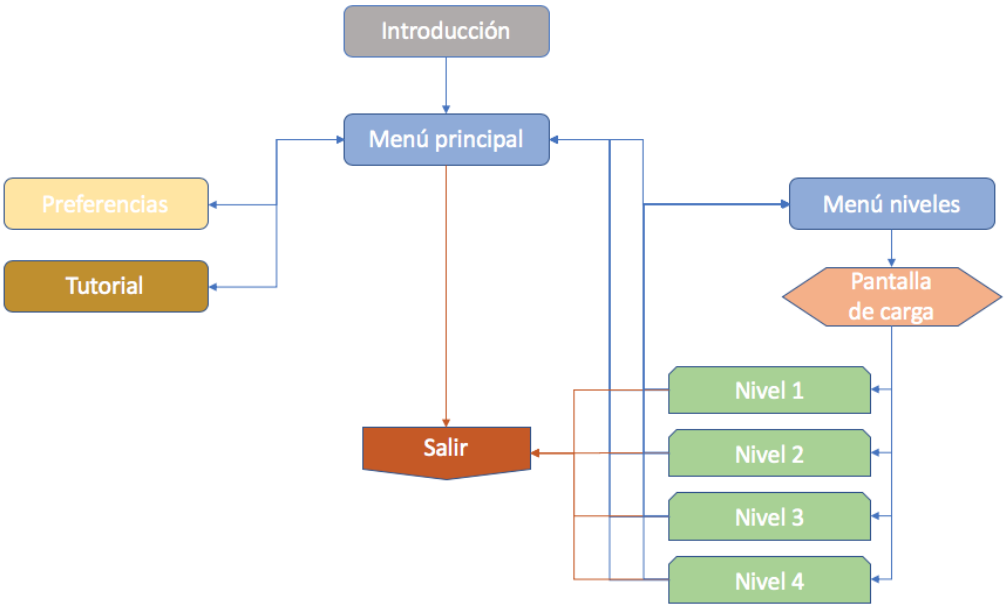

Figura 16 – Flujos de las escenas del juego

Básicamente, de la escena introductoria se pasa al menú principal desde donde se puede ir a otras escenas. En la pantalla de niveles se puede escoger que nivel jugar y acceder a la escena en concreto. Una vez en el nivel se puede: finalizarlo, pausar y volver al menú principal o salir del juego. Finalizar el nivel puede ser o bien porque el jugador llega hasta al final o porque pierde todas las vidas.

Desde el menú principal también se puede acceder a la pantalla de configuración desde la cual se puede volver al menú principal. Otra escena a la que se puede acceder desde el menú principal es la de tutorial o como jugar. Desde esta escena también se puede volver atrás hasta el menú principal.

Así pues, la escena con el menú principal es el eje vertebrador desde donde se puede acceder a las demás escenas del juego. Para conseguir que todo funcione como debe y que se pueda jugar al videojuego en los niveles, existen una serie de componentes que trabajando todos juntos logran que el juego se ejecute y proporcione al jugador la experiencia de videojuego y de producto electrónico. En las siguientes secciones se describen estos componentes, que lógica siguen, como se han

construido y como se relacionan entre todos ellos. A modo resumen, la lista de componentes es:

- GameManager – Gestión del juego y los niveles
- GameController
- Personaje principal
- Cámara y fondo
- Elementos de los niveles
- Enemigos
- Jefe de nivel
- UI – Interfaz gráfica del juego
- Modo contrarreloj

4.8 GameManager – Gestión del juego y los niveles

El GameManager (en adelante GM) es el director general de la información y de los flujos que sigue el videojuego. Creado para seguir el modelo “singleton”, en todas las escenas del juego solo existe una única instancia de este elemento y siempre es la misma.

Este componente tiene las siguientes funciones: guardar y compartir información relevante de manera trasversal, gestionar los flujos del juego entre escenas y gestionar las configuraciones de ciertos elementos del videojuego.

Primero, el flujo del juego y escenas. Como se ha visto en el diagrama de la **Figura 16** existen ciertos flujos entre las escenas del juego que crean las relaciones de conexión entre todas ellas. Para poder manejar las escenas de una forma más genérica, pero a la vez personalizada, se ha creado una máquina de estados que utiliza la interfaz IGameManagerState y que maneja el GM. Esta interfaz implementa una serie de métodos que todas las escenas deben tener: StateAwake, StateUpdate, StateChange, StateOnSceneLoaded y StateOnSceneUnLoaded. El GM llama a estos métodos en su código cuando toca y según la escena se ejecutarán unas órdenes u otras.

Las clases que utilizan esta interfaz son del estilo GameManagerStateXXX y van ligadas a cada escena. Alguna de estas clases se reutiliza para más de una escena, como es el caso de la GameManagerStateLevel que se usa para los cuatro niveles implementados en el juego. Estas clases son las encargadas de iniciar y gestionar el cambio de escena (y por lo tanto de clase) y de ejecutar todo el código que sea necesario cuando se ha cargado la escena o antes de que se elimine para dar paso a otra. Las escenas se guardan en un diccionario con su nombre en texto y una enumeración, EGameScenes, que sirve para disponer de un listado sencillo para ejecutar los cambios de escena y abstraerse del nombre en texto de la misma.

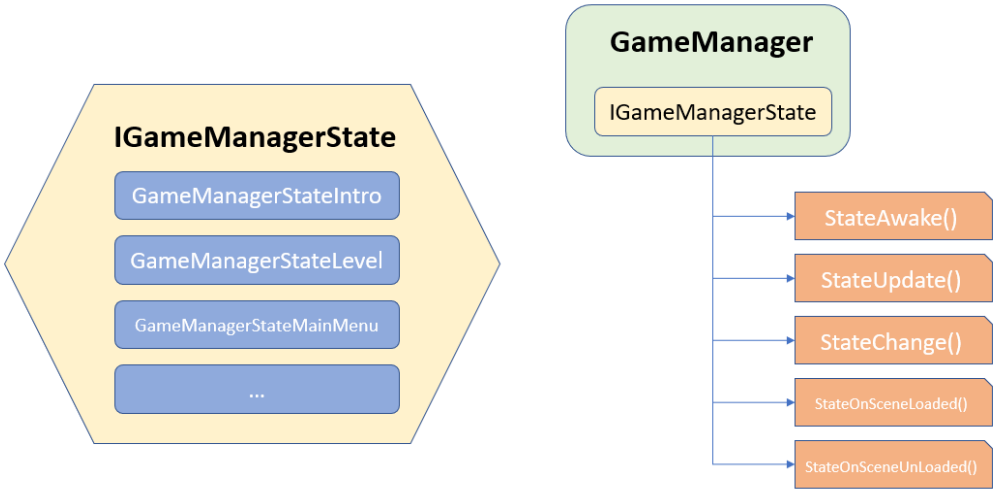

Figura 17 – Esquema GameManager y IGameManagerState

También disponen de un método Update para actualizarse en cada *frame* y un método Awake para realizar pre-configuraciones antes de que se empiece el bucle de actualizaciones de la escena. El GM contiene un miembro de tipo IGameManageState para guardar la referencia a la clase que está activa en ese momento.

De esta forma se puede tener un comportamiento homogéneo entre todas las escenas, disponer de métodos para hacer configuraciones o tareas de limpieza y gestionar el cambio de escena cuando sea necesario. A la vez, se pueden implementar métodos más particulares por escena sin tener que modificar el comportamiento de las demás escenas. Frente a otras opciones como por ejemplo gestionar todos los estados en una misma clase, se ha escogido esta manera ya que aporta relativa sencillez, orden, abstracción y homogeneización para la gestión de los flujos del juego. Implementar una escena nueva e incorporarla a los flujos del juego tiene menos dificultad de esta forma.

Segundo, métodos de configuración. Al iniciarse el juego una serie de métodos se ejecutan para configurar: el *input* del juego (detallado en la siguiente sección), el diccionario de escenas y el controlador de audio del juego (componente AudioMixer de Unity). Estos tres métodos hacen usos de ficheros de configuración xml para poder cargar la información en el GM y que luego el resto de clases/componentes de cualquier gameObject de cualquier escena pueda acceder fácilmente.

Otros métodos importantes que tienen que ver con la configuración del juego son: actualizar la puntuación del nivel, actualizar el tiempo de la prueba contrarreloj del nivel, desbloquear el siguiente nivel y desbloquear la prueba contrarreloj del nivel. En la misma línea, existen otros métodos para recuperar la información sobre puntuaciones, tiempos, niveles desbloqueados y pruebas contrarreloj desbloqueadas que algunos componentes del juego necesitan para gestionarse (por el ejemplo el menú de niveles).

Tercero, información transversal. El GM sirve como almacén de información para todas las escenas del juego. Guarda y asigna referencias de componentes o gameObjects que otros gameObjects puedan necesitar recuperar en otras escenas. Por ejemplo, al principio de cada nivel se ejecuta un método que asigna referencias a gameObjects o componentes que dentro del nivel es necesario referenciar para interactuar con ellos. El GM centraliza esta información para que no se tenga que ir a buscar cada vez por quien los necesite.

En la gestión de las escenas existe un caso muy particular en los niveles de juego que el GM también se encarga de gestionar: pausar el juego. Pausar el juego en Unity para un proyecto como este suele significar alterar la escala de tiempo que mide internamente Unity, y esto es lo que gestiona el GM. Cuando el jugador pausa el juego en un nivel, se ejecuta un método en el GM para, valga la redundancia, poner el juego en pausa. El GM dispone de un evento al que quienquiera puede suscribirse y ser notificado de cuando el juego se pausa. Este evento puede ser útil para aquellos elementos en los que la pausa instantánea no sea crucial, pero por ejemplo, el contador de tiempo de la prueba contrarreloj debe pausarse inmediatamente, no puede esperar a que el evento le llegue. Para este caso especial se gestiona desde el mismo GM. Para otros elementos, cambiar la escala de tiempo de Unity suele ser suficiente para pausarlos o reanudarlos.

Otra gestión importante que lleva a cabo el GM es la del ciclo vida-muerte del jugador dentro del flujo del juego. Para empezar, el GM asigna 5 vidas al jugador. Cuando éste muere en un nivel, se llama a un método del GM para encargarse de lo que tiene que pasar a continuación. En el modo normal si aun le quedan vidas al jugador el GM restará una vida del contador y se encargará de resucitar al jugador, recuperarle la vida y ponerlo en el punto de control (o el principio del nivel) para retomar la partida (el GM guarda la información del último punto de control alcanzado por el jugador dentro del nivel). En el caso de que no queden vidas, el GM volverá a la escena del menú principal y asignará 5 vidas nuevamente. Este punto puede evolucionar en futuras versiones del juego de tal manera que, por ejemplo, el jugador sea capaz de conseguir vidas recolectando ítems en el nivel (característica de los juegos de plataformas) o que las vidas se vayan regenerando al cabo del tiempo (aquí se puede abrir la puerta al modelo *freemium*). Como se comentará más adelante, el modo contrarreloj y el cuarto nivel se gestionan un poco diferentes. Los dos modos vuelven a cargar la escena nuevamente cuando el jugador muere y solo en el cuarto nivel se pierden vidas.

El GM también interacciona en ciertos momentos con los niveles de juego. En concreto, cuando dentro del nivel el jugador alcanza un punto de control la referencia la guarda el GM. Al morir el personaje, dependiendo de si le quedan vidas o no la transición de la escena cambia y aquí interviene el GM. Como se ha comentado anteriormente, al finalizar el nivel, el GM interviene en la actualización de estadísticas

del nivel así en como en el desbloqueo de niveles y modos contrarreloj. El GM también tiene participación en el tipo de transiciones al finalizar un nivel.

Para finalizar con el GM, otra característica que incorpora es la gestión de la música de fondo de las escenas. En el fichero “audios.xml” (Assets/Resources/ConfigFiles) se configura que archivo de sonido debe reproducirse en cada escena. Cuando se carga la escena, la clase estado llama a un método del GM para carga la música y empezarla a reproducir. Puede optarse también por no poner música en la escena.

Como se ha podido constatar, el GM es el cerebro del videojuego y el gran administrador de escenas y flujos. Aun así, sería posible fragmentar o desacoplar ciertas gestiones del GM para que otros pequeños “gestionadores” se ocuparan de otras tareas. Por ejemplo, aunque no se ha implementado en este proyecto, se podría hacer un mánager de sonido que fuera desacoplado del mánager principal del juego u otro mánager que se encargase de la carga de recursos o la gestión de ficheros. Para proyectos más grandes, esta idea seguramente puede funcionar mejor que solo disponer de un mánager central para gestionar todo.

4.9 GameController

En la definición del juego se optó por especificar que el videojuego pueda ser lanzado en diferentes plataformas como PC y consolas (PS3, PS4, Xbox, …). En este caso, de plataforma a plataforma los controles cambian y eso podría ser un problema dentro de la programación del juego para poder adaptarse a tantos controles.

Para evitar que la programación de la lógica referente a los controles del jugador se vuelva tediosa, con código repetido y redundante por cada plataforma y control o que para implementar una nueva plataforma haya que tocar el código de los controles en cualquier parte donde se usen en los *scripts*, se ha seguido una forma más genérica y por capas para poder atacar este tema. En primer lugar, se han definido una serie de métodos generales para concretar los *inputs* que puede usar el jugador cuando interacciona con el juego. Estos métodos son tales como las acciones del jugador (saltar, desplazarse, atacar, …) o métodos para navegar por los menús o pausar el juego. De esta forma, a través de todo el código que contienen los *scripts*, siempre se utilizarán los mismos métodos sea cual sea la plataforma y no habrá que particularizar código dependiendo del dispositivo. Estos *scripts* son los siguientes:

- KeyboardGameController
- PS3GameController
- PS4GameController

Seguidamente, se han definido otros *scripts* que mapean todos los botones de los controles de las plataformas de destino, aunque por el

momento solo se ha podido conseguir para el teclado de PC, mando de PS3 y mando de PS4. Se han mapeado todos los botones de estos mandos manualmente y trasladado a métodos dentro del código los diferentes estados de los botones (pulsado, mantenido, liberado). Desafortunadamente, no se disponía de ningún mando de Xbox para realizar la misma labor que con los otros mandos, aun así, tal y como se ha definido el sistema de controles incorporar un mando de otra plataforma sería tan sencillo como realizar el mapeo de botones y añadir los *scripts* correspondientes. Las clases XXXGameController incorporan un miembro referente a su tipo de controlador que a la vez mapean los botones a los métodos genéricos. Los *scripts* de mapeo de botones son los siguientes:

- KeyboardController
- PS3Controller
- PS4Controller

Finalmente, es necesario disponer de una manera para unir estas clases diferentes y que puedan utilizarse independientemente por las demás partes del código, sin que tengan que saber de que plataforma se trata. Aquí es donde entra en juego la interfaz IGameController de la cual heredan las clases XXXGameController mencionadas más arriba. De esta forma, como todos heredan de la misma interfaz cualquier parte del código sabe que hay unos métodos genéricos implementados por al interfaz y no se tienen que preocupar de que plataforma se trata.

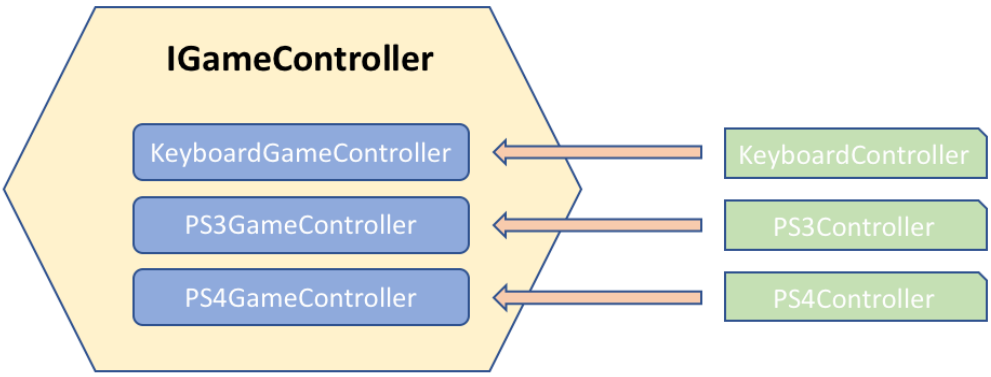

Figura 18 – Esquema de clases para el *input* del juego

Como se ha indicado en la sección del GameManager, al inicio del juego se detecta en que plataforma se está ejecutando el juego y se instancia la clase correcta de XXXGameController que se referencia a través de la interfaz IGameController, así cualquier código de otro *script* puede acceder a los controles genéricos del juego.

4.10 Personaje principal

Esta sección diseño técnico del videojuego aborda uno de los elementos fundamentales: el personaje principal. En este tipo de juego, el personaje principal es el elemento más importante para el jugador, ya

que es a través del éste elemento con el que se puede interactuar con el videojuego. El diseño del personaje explicado en anteriores capítulos debe materializarse de alguna manera en el juego, sus características, sus funcionalidades, sus interacciones, etc.

Para acometer la construcción del personaje se ha decidido separar en diferentes componentes las funcionalidades que se necesitan para implementarlo. A continuación, se describen los componentes que conforman el personaje, sus principales características y como funcionan y se comunican entre ellos:

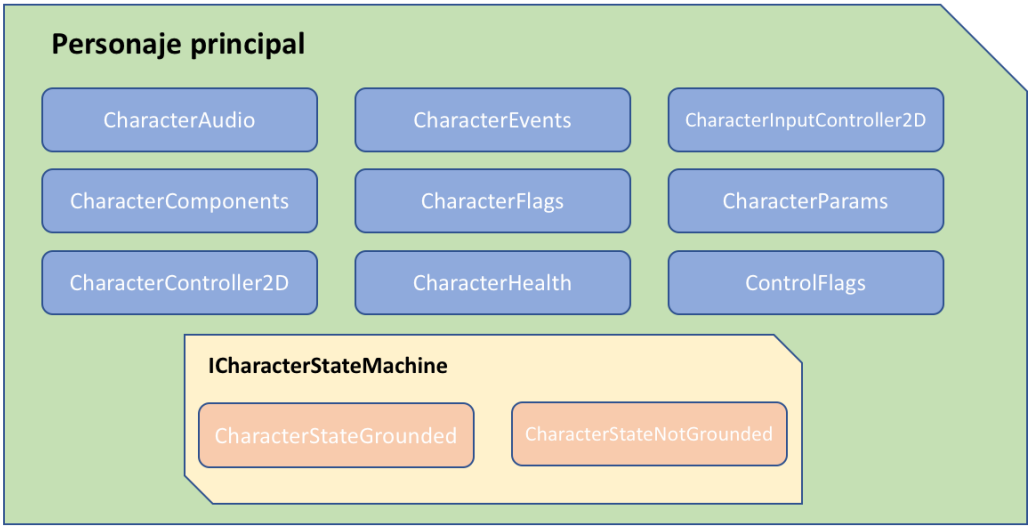

Figura 19 – Esquema de componentes del jugador principal

- Control y mecánicas del personaje

Se puede decir que los dos componentes más importantes del personaje principal son el CharacterController2D y el CharacterInputController2D. Estos dos componentes son el corazón del personaje ninja y trabajan juntos para implementar las mecánicas, el control, los estados y las animaciones.

El componente CharacterController2D reacciona a los *inputs* del jugador que le llegan a través del CharacterInputController2D y actualiza el estado y el comportamiento físico del gameObject. El personaje principal tiene diferentes Colliders según el estado en el que se encuentra (quieto, corriendo, planeando, deslizándose, etc.) que también son activados, desactivados o actualizados por este componente.

En este juego un elemento importante es saber si el jugador esta tocando el suelo o está en el aire. Esta distinción se necesita para poder acotar los estados y las transiciones entre ellos según se han definido las mecánicas (por ejemplo, si el personaje está en el aire no puede pasar a deslizarse). Para separar un poco el código del *script* y facilitar la programación, se ha implementado una pequeña máquina de estados muy simple con la interfaz ICharacterStateMachine. Actualmente, solo hay dos clases que la utilicen: CharacterStateGrounded y

CharacterStateNotGrounded, donde se puede apreciar esta separación suelo/aire. Esta interfaz cuenta con un método, StateControl, que se llama desde el método FixedUpdate del CharacterController2D para actualizar el estado y las físicas. Para pasar los posibles estados de una forma ordenada se tiene la clase ControlFlags que básicamente indica que acciones se deben ejecutar en la física del personaje según quiere el jugador.

El estado del que se habla en el anterior párrafo se entiende saber si el personaje está en el suelo o el aire y se está realizando alguna acción de tipo moverse, atacar, deslizarse o planear. Hay un último caso especial que es cuando el jugador muere en cual se deshabilita el control del jugador hasta que el juego transita a la fase correspondiente según el flujo programado. Por físicas se entienden las acciones que se llevan a cabo sobre el gameObject del personaje para que reacciones tanto al *input* del jugador como a su entorno en los niveles del juego. Para el desplazamiento se aplica una fuerza horizontal según la dirección que quiere tomar el jugador. Un material físico aplicado al gameObject hace que cueste un poco acelerarlo hasta una velocidad máxima y lo mismo para frenarlo hasta que esté quiero. Para saltar se aplica una fuerza vertical hacia arriba que combinada con la velocidad que lleve el personaje creará saltos verticales o combinados con desplazamientos horizontales (la gravedad se ha aumentado para acercarse a la jugabilidad que suelen ofrecer los juegos de plataformas donde el salto no funciona exactamente como en el mundo real). El deslizamiento funciona de forma parecida, se aplica una fuerza horizontal al personaje (si está en movimiento) en el mismo sentido del movimiento. Si está quieto no se aplica la fuerza y se entiendo como que el personaje se agacha. En el aire se puede controlar al ninja de dos maneras. Por un lado, cuando planea se aplica una fuerza en dirección vertical hacia arriba que contrarresta la fuerza de la gravedad del mundo del videojuego sin ser superior ha esta. Este hecho crea una sensación simplificado de planeo en el aire donde el jugador puede cambiar el sentido horizontal del personaje para desplazarse a la izquierda o a la derecha. Por otro lado, cuando el jugador salta se puede controlar mínimamente el personaje en el aire para hacer que aumente o disminuya su velocidad horizontal según sea el sentido de movimiento. Esta característica también se puede encontrar en algunos juegos del género de plataformas y permite dar un poco más de control al jugador sobre los saltos del personaje. Se ha pensado que esta funcionalidad puede encajar bien en ese juego ya que un ninja debería poder controlar, en cierta medida, sus movimientos en el aire. Todos los parámetros que se utilizan para las mecánicas son configurables y se encuentran en el componente CharacterParams, de esta forma es muy fácil poder cambiar la fuerza de salto o la velocidad de desplazamiento.

En el componente CharacterInputController2D, aparte de gestionar los *inputs* del jugador también se gestionan los cambios de estado del componente Animator que lleva acoplado el personaje ninja. Acorde con los cambios de estado físicos deben ir las animaciones del personaje. En

este componente también se gestiona al personaje cuando se pausa el juego o cuando se deshabilita su control, ya sea porqué ha muerto y se esta esperando a la reaparición o porque se ha llegado al final del nivel y se está reproduciendo la animación de final de nivel. Para controlar mejor las acciones que se pueden realizar existe el componente CharacterFlags que guarda información sobre el estado del personaje en la anterior actualización, así se puede saber si se estaba en el suelo o en el aire, planeando o deslizándose y gestionar mejor los cambios de estado.

- Otra opción de físicas para el personaje

En los anteriores párrafos se han descrito las físicas que sigue el personaje en respuesta al control del jugador y al entorno del nivel. Las mecánicas que se han implementado se rigen por la aplicación de fuerzas y restricciones sobre el personaje y cumplen de manera notable con la jugabilidad que se quería alcanzar.

Aun así, existe otra forma de poder aplicar las físicas e implementar las mecánicas del personaje principal. Si bien la forma actual deja que Unity calcule como debe reaccionar el gameObject del ninja a las fuerzas y contactos con otros elementos, se pueden implementar físicas propias simuladas en el código. Con la simulación de las físicas se pueden conseguir comportamientos (y por ende mecánicas) que quizás con el motor de Unity serían más difíciles de alcanzar. Una de las ideas que ha quedado pendientes es precisamente esta, implementar la lógica y simulación de las físicas propias para el personaje principal. Tal y como está estructurado en forma de componentes, el CharacterController2D y sus estados podrían sustituirse por otro componente que reaccione a los *inputs* del jugador y aplique la simulación de las físicas que correspondan. Para realizar la simulación hay que implementar ecuaciones del movimiento y actualizar el estado del personaje acorde a estas ecuaciones. De forma simplificada, Unity realiza estas simulaciones sobre un gameObject si se desea que sea el motor quien se encargue de actualizar el estado físico del mismo, perdiendo a la vez un poco de control sobre esta característica. Implementar físicas propias puede ser beneficioso para ganar más control sobre el personaje, pero también puede complicarse ya que cuanta más complejidad en las interacciones y mecánicas del personaje, más complejidad de programación habrá.

Para este proyecto se ha dejado que Unity se encargue de las físicas del personaje ya que el nivel de jugabilidad alcanzado es bastante correcto, pero para futuras versiones del juego sería interesante añadir un control de las físicas propio y mejorar aun más la experiencia de juego.

- Vida y muerte del personaje

Para gestionar la vida y muerte del personaje se usa el componente CharacterHealth. Este componente es el encargado de disminuir la vida

del personaje cuando es golpeado por un enemigo o una trampa y de reflejar los cambios en la barra de vida de la interfaz gráfica.

Cuando el personaje es herido, este componente dramatiza los daños haciendo que el *sprite* del personaje se tiña de rojo intermitentemente, de esta forma el jugador puede ver más claramente la situación. También interactúa con la cámara del juego para crear un efecto sacudida y sumarse a la dramatización de la situación (más adelante se comenta este efecto de la cámara).

Cuando la vida del ninja llega a cero, este componente gestiona el cambio de estado del Animator para reproducir la animación de muerte del personaje y también avisa al componente CharacterControllerInput2D para que deshabilite el control del jugador. Cuando la animación de muerte finaliza, el componente se comunica con el GameManager para gestionar el flujo del juego y pasar a la fase que toque según las vidas que le resten al jugador.

Como en otros componentes del personaje principal se pueden parametrizar aspectos como la vida máxima del personaje o la duración del efecto de teñir de rojo el *sprite* del jugador.

- Gestión de sonidos del personaje

El personaje ninja emite sonidos en ciertas situaciones para complementar la experiencia del juego. Por ejemplo, al saltar, deslizarse, atacar, planear o morir el personaje emite un sonido. El componente CharacterAudio se encarga de gestionar la emisión de sonidos del personaje y trabaja conjuntamente con otros componentes. Básicamente, en ciertos momentos del estado del personaje hay que emitir un sonido, así que el componente a través del CharacterAudio da la orden de reproducir el sonido que toque.

Para algunos estados se dispone de una variación de hasta tres sonidos, uno de ellos se escoge aleatoriamente y se reproduce. De esta forma se puede evitar una repetición molesta del mismo sonido, como por ejemplo si se está saltando muchas veces seguidas.

- Accesibilidad de la información entre componentes

Existe un último componente que se integra en el personaje principal que tiene como misión facilitar la accesibilidad de la información entre componentes. Este *script* es el CharacterComponents y guarda referencias a la mayoría de componentes del gameObject para que los otros componentes que interactúan entre sí puedan acceder fácilmente a la información que necesiten.

4.11 Cámara y fondo del nivel

- Cámara y Cinemachine

Un elemento primordial de todo juego son las cámaras. Estos elementos sirven para crear el enfoque y el modo en que se renderizará la imagen de cada *frame* en la pantalla del dispositivo que finalmente verá el jugador. Sin entrar en detalles de cómo funciona, lo que “ve” la cámara es lo que se acaba renderizando en la pantalla por eso es un elemento tan importante en el juego. Las cámaras permiten al desarrollador crear planos y efectos en las escenas de los juegos para que las transiciones no sean tan repentinas y tajantes.

Una mejor experiencia de usuario se consigue con una cámara que no sea rígida, entendiéndose este término como una cámara que sigue al jugador exactamente en una posición determinada definida normalmente por la posición del jugador. Para un juego de plataformas 2D se necesita una cámara capaz de se seguir al personaje y adaptarse a ciertas situaciones del nivel como puede ser que la cámara se desplace cierto margen arriba, abajo, izquierda o derecha según hacia donde mire el personaje o en situaciones concretas donde hay que ascender o descender. También que la cámara pueda seguir al personaje con cierto desfase para dar diferentes sensaciones al jugador.

Con todo esto se ha elegido Cinemachine como componente para la cámara principal de los niveles. Este recurso gratuito proporcionado por Unity trae de fábrica una serie de componentes que se adhieren al gameObject de la cámara para conferirle aptitudes como las comentadas previamente. Tras una serie de pruebas con la cámara en los niveles, se han ajustado los parámetros necesarios de los componentes de Cinemachine para lograr los comportamientos descritos anteriormente.

Desafortunadamente, la versión de Unity utilizada para el proyecto no admite la versión más reciente de Cinemachine y algunas características, como el desfase de seguimiento en el eje Y del movimiento, no se han podido incorporar. Aun así, en una futura actualización del juego esta característica puede actualizarse e incluirse perfectamente en la jugabilidad para mejorar la experiencia del jugador.

- Efecto sacudida

Otra característica implementada que tiene relación con la cámara es un efecto de sacudida cuando el jugador es herido. Aprovechando una funcionalidad que viene con Cinemachine, se ha creado el *script* CameraShake para llevar a cabo este efecto. Básicamente, Cinemachine incorpora una funcionalidad de “ruidos” que se puede utilizar para sacudir la cámara como de si un golpe se tratase. Este efecto junto a la situación cuando el personaje es herido, aporta una dramatización extra al jugador.

- Fondo del nivel

Un nivel al fin y al cabo es un trozo de mundo que se crea para que el jugador pueda jugar en él. Los juegos plataformas 2D suelen escenificar niveles, más o menos realistas, con diferentes elementos y un de ellos es el fondo del nivel.

Para conseguir un efecto de profundidad y no solo tener un fondo estático que sigue al jugador y la cámara, se ha implementado en el *script* BackgroundParallax un código para aportar un efecto de paralaje al fondo del escenario (movimiento horizontal únicamente). Este efecto visual se urde dividiendo el fondo en varias capas superpuestas y haciendo que se muevan a diferentes velocidades según el desplazamiento del jugador en una dirección u otra.

Para elaborar el efecto se han utilizado fondos de cinco capas que son gestionadas por el *script*. Repitiendo el fondo y sus capas tres veces se puede conseguir este interesante efecto moviendo los elementos de cada parte adelante o atrás según toque por su posición en el efecto. El *script* de gestión de paralaje admite diferentes velocidades para cada capa y actualmente se han ajustado con ciertos valores tras varias pruebas en los niveles.

4.12 Elementos de los niveles

Los niveles del juego incorporan una serie de elementos con los que el jugador puede interactuar ya sea para su provecho o detrimento. Estos elementos están pensados para aportar jugabilidad, interacción y para integrarse en el mundo del videojuego. Seguidamente se exponen los detalles técnicos de los principales elementos que se pueden encontrar en los niveles:

- Gemas

Estos ítems que el jugador puede recolectar son básicamente un *sprite* con la gema a mostrar y un componente que actúa como *trigger* para saber cuándo interactúan con el jugador. Poseen un componente propio para crear un movimiento de levitación hacia arriba y hacia abajo que se activa cuando el gameObject es visible por la cámara del juego (así se pueden ahorra recursos para no tener que estar actualizando en cada *frame* de las gemas que no se ven por pantalla).

Cuando el jugador entra en contacto con alguna gema esta se destruye y aumenta la puntuación del jugador en el nivel. Solo hay una excepción con las gemas de vida (forma de corazón) que al desaparecer recuperan un poco de vida al jugador. Cuando una gema desaparece crea un efecto de partículas con un color similar al de la propia gema.

Toda la lógica de las gemas está controlada por el *script* CrystalGem.

- Plataformas

Las plataformas son gameObjects que se han creado para que el jugador pueda saltar encima de ellas y poder conectar con otras partes del nivel. Están hechas a partir de un componente de Unity llamado Tilemap para poder representar adecuadamente los *tiles* con los que se ha elaborado el nivel e incorporan un componente Collider, con forma rectangular que cubre la parte superior de la plataforma, así como un componente PlatformEffector2D para que el jugador pueda saltar a través de ellas desde abajo (otra característica típica de este género).

Estos objetos pueden ser estáticos o moverse de un lado a otro con una clase propia que proporciona diferentes tipos de movimientos cíclicos y que puede aceptar parámetros como la longitud del movimiento o la frecuencia. El control del movimiento de las plataformas móviles se ejecuta desde el *script* MovingPlatform.

- Trampolín

Este elemento dentro de los niveles sirve para poder saltar con más fuerza y llegar a otras partes del nivel. Está compuesto por una animación de un muelle con una plataforma roja adosada en el extremo superior que cuando el usuario salta encima le proporciona un impulso (parametrizable) al mismo tiempo que se activa la animación del trampolín. Una flecha en la parte superior del gameObject indica la dirección hacia a que se aplica la fuerza del trampolín.

Existe otra versión de este mismo elemento en la cual la dirección de la fuerza varía entre dos ángulos límite con una frecuencia parametrizable. Con esta variación se abre un abanico de posibilidades en cuanto a situaciones dentro de un nivel para superar e interactuar con este elemento.

Las dos versiones del trampolín las controlan los *scripts* AnimatedSpringboard y SwingingAnimSpringboard.

- Zona de impulso de planeo

Este elemento tiene una finalidad muy simple que es impulsar al jugador hacia arriba cuando éste se encuentra dentro de su zona de afectación. Para que el jugador pueda ser impulsado se debe estar en modo planeo, cualquier otro estado del jugador dejará de recibir el impulso.

El gameObject cuenta con un *trigger* para saber cuando el jugador está dentro de la zona y también incorpora un componente de tipo Effector2D que cuando se activa hace que los gameObjects que haya dentro del área del *trigger* se vean afectados por el impulso. Se ha implementado una lógica para que el componente solo se active cuando el jugador está planeando.

Un componente visual en forma de viento y de efecto de partículas acompaña a este elemento que puede tener mucha versatilidad para combinarlo en diferentes situaciones y retos. Todo el comportamiento está regido por el *script* GlideImpulseZoneController.

- Trampas

Existen los siguientes tipos de trampas que se pueden encontrar a lo largo de los niveles: bola de fuego, hachas y picas oscilantes y pinchos estáticos. Aparte de los enemigos, estos elementos son los que suponen una amenaza para el jugador ya que pueden llegar a matarlo.

La bola de fuego consta de un agujero adosado a una superficie por la que se escupen bolas de fuego. La cadencia, la velocidad y el daño de las bolas de fuego es parametrizable y se destruyen en contacto con el jugador (hiriéndolo) o contra una superficie. Estas bolas siguen una trayectoria recta y los *scripts* que controlan esta trampa son Fireball y FireballHole.

Las hachas y picas oscilantes son trampas con un elemento como un hacha o una pica que oscila colgado de una cadena entre dos ángulos límite. Existen diferentes variaciones visuales de esta trampa, pero todas se comportan igual y son controladas por el *script* SwingingSpike. Cuando la trampa impacta contra el jugador este sale herido e impulsado hacia la dirección del movimiento de la trampa para así no crear una situación de daño múltiple y acabar con la vida del jugador demasiado rápido.

Los pinchos estáticos se encuentran cubriendo todos los huecos por donde puede caer el jugador y también en ciertas superficies de los niveles. Constan de un componente Collider y del *script* StaticLandSpike que simplemente cuando detecta al jugador lo mata al instante. Esta trampa se puede encontrar en gran variedad de situaciones en los niveles.

Nuevas trampas se pueden añadir sin problemas ya que para crear variedad, diversión y retos en diferentes situaciones a los jugadores es bueno implementar diferentes elementos. La combinación entre trampas u otros elementos de los niveles pueden crear múltiples y diversas situaciones para poner en un aprieto al jugador. También es cierto que no cualquier tipo de trampa es válida ya que hay que respetar la estética del juego, pero esto no limita las posibles combinaciones que se puedan crear para implementarlas en el juego. Por el momento se ha podido implementar estas tres trampas usando los recursos disponibles al alcance y que se ha pensado que pueden añadir retos y dificultades a en los niveles para crear situaciones peligrosas para el jugador respetando la estética del mundo del juego.

4.13 Enemigos

Si bien pueden existir juegos de plataformas donde el reto para el jugador se escenifique únicamente en saltos, trampas o cualquier reto físico del estilo, la inclusión de enemigos a lo largo de los niveles añade un poco más de emoción y riesgo al juego.

En este proyecto se ha implementado un tipo de enemigo con diferentes aspectos (**Figura 9**) para ser incorporados en ciertas zonas de los niveles. El aspecto de estos enemigos es de criaturas tipo “goblin” con la cabeza visiblemente más grande en proporciones que el resto del cuerpo. Si bien la estética de los enemigos en contraste con la del juego podría parecer que se ajusta, por un lado, no se disponía de más recursos gráficos que encajasen mejor y, por otro lado, al ser un mundo el del videojuego con elementos fantásticos, puede llegarse a un compromiso entre ambas estéticas. Idealmente, estaría bien poder disponer de recursos gráficos adaptados a la estética y el mundo del videojuego o poder crearlos para que se adapten a la funcionalidad que se quiere implementar en el enemigo. Al final, implementar una lógica para un enemigo es independiente de los recursos, pero para implementar el enemigo en el videojuego se necesitan unos recursos gráficos acordes a la funcionalidad programada.

La IA de estos enemigos es muy simple, atacar de lado a lado en una zona delimitada en ciclos de ataque-descanso, y es cuando el enemigo descansa el momento en que se le puede herir. El *script* que controla los ciclos del enemigo es el EnemyPointToPointAttacking y se apoya en la enumeración EEnemyStates para crear una pequeña máquina de estados dentro del propio componente y así gestionar los ciclos comentados anteriormente. Esta enumeración contiene diferentes estados en los que puede estar el enemigo como atacar, quieto, correr, correr-atacar, etc. y es usada también por el jefe de nivel. La implementación de los enemigos no tiene porqué usar todos los estados de la enumeración, simplemente los que necesite para el tipo de enemigo que se esté creando.

Para atacar al jugador, hay un punto específico de la animación de ataque en el cual se comprueba si se ha alcanzado al personaje principal, en tal caso se le resta vida y se le empuja en la misma dirección en la que se propina el ataque. Como las distancias que recorre el enemigo en sus ciclos no son muy extensas y la cadencia de golpes en el ataque no es muy elevada, empujando al personaje principal cuando recibe un ataque da tiempo para que el jugador reaccione al siguiente ataque y no sufra excesivos daños (que podría ser contraproducente para la experiencia de juego).

Las animaciones de las diferentes fases por las que pasa el enemigo se controlan a través del mismo *script* principal y gestionan los estados del componente Animator que tienen acoplado los gameObjects de los enemigos.

Para facilitar la reusabilidad de los enemigos, se ha decidido parametrizar características como la cantidad de vida, el daño de ataque, la velocidad de movimiento o los ciclos de ataque-descanso. De esta forma, con un mismo *script* pueden crearse diferentes versiones de un enemigo y reutilizar el código. Por otro lado, los gameObject de los enemigos se han guardado como prefabs para que cuando sea necesario cambiar algún parámetro, este cambio se extienda a todos los enemigos de los niveles.

Para acabar, los enemigos incorporan un elemento de UI para hacer visible la vida restante que poseen. Se trata de una barra de color rojo hecha con un deslizador que se sitúa encima de la cabeza de los enemigos y sigue el movimiento de éstos. Cuando el jugador consigue herir al enemigo, la barra se actualiza para reflejar la vida restante del enemigo.

Figura 20 – Barra de vida enemigos

4.14 Jefe de nivel

Un jefe de nivel es una implementación en la línea de este tipo de juegos, donde en ocasiones hay niveles especiales donde para superar el nivel es necesario derrotar a un enemigo “especial”. Otra variación se suele encontrar en la forma de implementar el jefe de nivel al final de la escena, otra vez teniendo que derrotarlo para poder superar el nivel.

Así pues, tras evaluar diferentes recursos gráficos (**Figura 10**), se decidió implementar un enemigo “especial” en el cuarto nivel. El diseño del enemigo también sigue la línea de muchos jefes de nivel en juegos de plataformas. Normalmente, se suele disponer de un espacio limitado donde enfrentarse al enemigo y éste suele tener varias fases con diferentes ataques y una mecánica pseudo repetitiva. Además, para atacar al enemigo suele ser en algún momento en concreto de la fase/ciclo y no en cualquier ocasión, incluso a veces siendo trabajo del jugador propiciar la situación.

El jefe del nivel cuatro cuenta con una estética samurái, en la línea de la estética del videojuego. Está programado para atacar en tres fases diferentes con tres tipos de ataques distintos y ciclos de recarga de ataque y aperturas para que el jugador pueda herirlo. Toda la implementación se ha llevado a cabo en el *script* SamuraiHeavyBoss y haciendo uso de la interfaz IEnemy y las enumeraciones EBossFases y EEnemyStates. Dentro del *script* principal se ha dividido el código por fases, estados y se ha hecho uso de corrutinas para las repeticiones de ciertos ciclos. A parte de gestionar toda la lógica, los estados y sus transiciones, este componente también gestiona paralelamente las

transiciones entre los diferentes estados del componente Animator que posee el gameObject del jefe de nivel. Este componente se encarga de hacer factibles las animaciones gráficas del personaje como atacar, ser herido, quieto con guardia alta/baja, etc. Las animaciones acompañan a cada fase/estado de la batalla contra el samurái.

Idealmente, hubiese sido mejor crear una máquina de estados con diversas clases que manejasen los estados por los que transcurre la batalla contra el enemigo, pero se ha tenido que llegar a un compromiso entre tiempo y resultados que ha propiciado la programación del comportamiento del jefe de nivel de esta forma.

El jefe de nivel sigue el patrón ciclos de ataque y pausa para recuperar el ataque. En esta pausa el samurái baja la guardia y es ahí donde puede ser golpeado por el jugador. También tiene un estado de recarga de ataque con la guardia alta en el cual no se le puede herir. Tanto la barra de salud como la de ataque se gestionan desde el componente principal y se han trasladado a la parte superior derecha de la pantalla para enfatizar que no se trata de un enemigo cualquiera y en contraposición a la barra de salud del personaje para mostrar el duelo/batalla entre ambos.

Como se puede observar, el nivel es un reaprovechamiento del primer nivel, pero en un espacio delimitado, para que el jugador no pueda rehuir la batalla con el enemigo.

A continuación, se pasa a explicar las tres fases de este jefe de nivel y sus principales detalles técnicos.

- Fase 1

En la fase 1 el enemigo samurái primero en posición de guarda recarga su barra de ataque para empezar a correr de lado a lado de la pantalla lanzando espadazos en movimientos rápidos y fuertes. Este estado tiene tres ciclos después de los cuales el enemigo se para, baja la guardia y se pone a recargar el ataque. En este momento si se ataca al enemigo se le puede herir, volverá a ponerse en guardia, recargará el ataque y volverá a repetir los ciclos. Si no se le ataca, simplemente recargará el ataque con la guardia baja y volverá a repetir los ciclos.

Cuando el enemigo ataca, para determinar si se ha golpeado al jugador hay cuatro puntos clave en el recorrido de la espada del samurái que están coordinados con la animación de ataque para saber en que momento se debe mirar si se ha golpeado al jugador. En caso de que se detecte que se ha conseguido atacar al jugador con éxito, se le resta vida y se le aplica un empuje haciéndolo saltar por los aires en dirección contraria a la que va el enemigo. Se ha implementado de esta manera para evitar que una vez que el jugador sea golpeado por el enemigo se pueda volver a golpear fácilmente al jugador. Los ataques del jefe de nivel hacen más daño que los de los enemigos básicos y varios ataques

seguidos podrían acabar con la vida del personaje rápidamente y frustrar al jugador.

Una vez se haya conseguido herir al jefe del nivel un par de veces se pasará a la siguiente fase.

- Fase 2

En la segunda fase, aparecen unas plataformas estáticas llenas de pinchos mortales y el enemigo se coloca en la más alta de todas. Una lluvia de kunais empieza a perseguir al jugador, se lanza un kunai a la vez por encima del jugador y éste desciende de arriba a bajo. Si el kunai impacta sobre el personaje le restará vida, si impacta contra el suelo desaparecerá. Un efecto de partículas en forma de cola de humo se puede apreciar en la caída del kunai. El comportamiento de estos kunais está regido por el *script* SamuraiHeavyBossKunai y la instanciación la realiza el propio componente del enemigo.

Este ciclo dura unos segundos y tras acabar los pinchos mortales de las plataformas desparecen mientras el enemigo recarga su barra de ataque. Es en este momento cuando el jugador debe subir por las plataformas y atacar al jefe de nivel. El sistema de recarga de ataque y guarda alta/baja es el mismo que en la fase anterior.

Tanto si se consigue atacar al enemigo como si no, una vez el samurái recarga su barra de ataque los pinchos mortales de las plataformas aparecen de forma parpadeante indicando al jugador que debe bajar al suelo. Si el jugador se queda en alguna plataforma cuando los pinchos reaparecen por completos lo matarán al instante. Esta forma de parpadeo de peligros es también muy característica de este género de videojuegos.

Tras dos ataques en esta fase realizados con éxito al enemigo se pasa a la siguiente fase.

- Fase 3

En la tercera y última fase, el enemigo levita en una de las posiciones arbitrarias repartidas por la escena. En ese momento el enemigo recarga el ataque y se lanza directo a la posición última que tenía el jugador cuando ha acabado de recargar la barra de ataque. Seguidamente, el samurái busca la posición a izquierda o derecha para realizar una estocada rápida hasta ese punto donde estaba el jugador. Las estocadas en el suelo se repiten dos veces y tienen un breve lapso de tiempo entre ellas para dejar que el jugador puedo reaccionar.

En cualquiera de los ciclos de ataque de esta fase, si el jugador es alcanzado y atacado por el jefe de nivel pasará lo mismo que en la primera fase.

Como en las anteriores fases, si se logra herir al samurái en dos ocasiones se habrá conseguido derrotarlo y se superará el nivel. El jefe cae derrotado al juego y la gestión del final de nivel es parecida a la de superación de cualquier otro nivel, con un *banner* y efecto de sonido.

Este jefe de nivel cuenta con sus propios efectos de sonido para desmarcarlo de los enemigos más simple de los otros niveles. Por otro lado, para realizar una mejor y más suave transición entre las fases de la batalla se ha optado por hacer que el enemigo se desvanezca lentamente con una risa malévola y transcurridos unos segundos reaparezca en la nueva posición de la fase de la batalla.

Por último, se ha reaprovechado de tal manera la estructura de los niveles normales para poder reutilizar parte de la programación que éstos siguen. Una de las cosas que cambian es que al morir se recarga la escena para volver a empezar desde el principio ya que no hay puntos de control en la batalla contra el jefe de nivel. Ésta puede no ser la mejor manera de optimizar los recursos, pero evita tener que realizar programaciones más complejas para guardar estados, posiciones, etc. Aun así, en un juego con requerimientos elevados se debería de optar por otra estrategia para gestionar el ciclo vida-muerte-batalla.

4.15 UI – Interfaz gráfica del juego

Dentro de la UI del proyecto se encuentran diversos elementos que dan la apariencia visual final al videojuego. Estos elementos están gobernados por diferentes *scripts* para adecuar sus comportamientos a la manera que es necesaria para cada parte de la escena y para las interacciones entre los elementos que se puedan establecer.

Los elementos de la interfaz gráfica están implementados de tal forma que cuando se cambia la resolución del juego éstos se adaptan para mantener el aspecto. De esta manera se evitan descuadres, superposiciones o elementos movidos a posiciones extrañas cuando se ajusta la resolución de la pantalla.

- Menús, botones y otros elementos

Son elementos seleccionables dentro de las escenas, generalmente en menús. Los botones tienen cuatro aspectos: color azul que indica que el botón no está seleccionado, color rojo para indicar que el botón está seleccionado, color verde indica que se ha pulsado sobre el botón seleccionado y color gris para indicar que el botón está bloqueado.

Para poder añadir ciertas funcionalidades extras a los botones, así como sus colores dependiendo de su estado, se ha creado el *script* UIButton y añadido como componente a los gameObjects de los botones.

Por otro lado, la gestión de los botones tanto la navegación como la acción que tienen asociada se gestiona a través de dos *scripts*:

UINavigation y los *scripts* de menús UIXXXMenu (por ejemplo, UIMainMenu). Los *scripts* de menú requieren de un componente UINavigation para gestionar los botones y además también asignan las acciones que deben realizar los botones al ser pulsados. El propio *script* de menú también contiene los métodos a los controles del juego para ser capaz de gestionar la navegación y acciones realizadas por el jugador.

Como punto extra particular a resaltar en el menú de niveles, los estados de los botones que enlazan con las escenas de juego dependen de si el jugador ha conseguido desbloquear el nivel en cuestión. El *script* del menú UILevelsMenu se nutre, a través del GameManager, del archivo de configuración de niveles “levels.xml” para determinar que botones de nivel activar y cuales no.

Además, el menú de niveles incorpora una ventana emergente que el jugador puede activar a voluntad para ver estadísticas relacionadas con el nivel. Estas estadísticas informan de la mejor y última puntuación del nivel y de los tres mejores tiempos históricos conseguidos en el modo contrarreloj. La gestión de esta ventana y la información de las estadísticas la hace también el componente UILevelsMenu.

Finalmente, los *scripts* de menús pueden también gestionar otros elementos de la UI como otro tipo de botón para activar o desactivar la pantalla completa y un deslizador gráfico para visualizar el nivel de volumen del juego.

- Elementos gráficos de la UI de los menús

Tanto el menú principal como el menú de tutorial incorporan algunos elementos gráficos que mejoran la experiencia visual del juego o que sirven para mostrar cierta información al jugador.

En el caso del menú principal, existen unos gameObjects con sus correspondientes *scripts* de control para mostrar unas animaciones por pantalla. Estas animaciones son un árbol de sakura con un efecto de partículas, un kunai que atraviesa la escena y al personaje principal apareciendo y desapareciendo en diferentes partes del lado derecho de la escena.

Para el menú del tutorial, la gestión se hace a través de un modo “diapositiva” en la que las varias pantallas que aparecen son como diapositivas que se pueden navegar adelante y atrás. Además de texto e imágenes, se añade una pequeña animación que muestra de manera muy básica al personaje principal atacando y matando a un enemigo.

- Elementos de la UI de los niveles

Otro grupo de elementos a destacar dentro la interfaz gráfica del juego son los que atañe a los niveles de juego, que van destinados a mostrar información útil del juego/nivel al jugador. De izquierda a derecha de la

pantalla, primero se encuentra la barra de vida del personaje principal y la cantidad de vidas que posee. La barra de vida es un deslizador que se gestiona a través de un componente del personaje principal explicado anteriormente. Una particularidad de la barra es que a medida que se reduce la vida el color cambia gradualmente de verde a rojo (con colores intermedios), dando un poco más de dramatismo y tensión al jugador a la vez que estos colores se asocian a situaciones de tranquilidad o peligro.

Las vidas que posee el jugador son simplemente un texto al lado de la barra de vida para indicar visualmente las vidas restantes al jugador. El número de vidas y el texto asociado en la UI de los niveles lo gestiona el GameManager.

A la derecha del todo de la pantalla se encuentra el marcador de puntuación compuesto por un texto y una imagen de gema. Para cada nivel, aparecen diferentes imágenes de gemas que indican al jugador cuales puede encontrar en el nivel en cuestión. El texto del marcador de puntuación está referenciado por el GameManager en cada nivel para que los gameObjects de las gemas puedan acceder a él y cambiar la puntuación cuando se recolectan por el jugador.

Figura 21 – Elementos gráficos de los niveles

Para el cuarto nivel donde no hay ítems para recolectar y es donde se desarrolla la batalla contra el jefe de nivel, el lado de la derecha de la pantalla se sustituye por las dos barras asociadas al jefe enemigo. Estas barras, que son deslizadores, muestran al jugador el ataque acumulado del enemigo y la cantidad de vida restante. Estos elementos de la interfaz relacionados con el jefe se controlan por sus *scripts*, como se explica en el apartado correspondiente.

Figura 22 – Elementos de la UI del cuarto nivel

Otro pequeño elemento relacionado con los enemigos, pero no por ello menos importante es su barra de vida. Esta barra está situada encima de los enemigos y se mueve con ellos para mostrar al jugador la vida restante. Es un elemento de la UI que solo aparece cuando se encuentra un enemigo en el nivel y éste aparece en la pantalla. Al igual que con el jefe de nivel, los *scripts* asociados a los enemigos controlan el deslizador de esta barra.

En el modo contrarreloj aparecen un par de elementos en la pantalla del juego. Estos elementos son un texto con el tiempo actual gastado del modo y una pequeña animación de *sprites* de un reloj de arena. Estos

elementos se controlan a desde los *scripts* del modo contrarreloj que se explican más adelante.

Figura 23 – Elementos de la UI del modo contrarreloj

Por último, se han implementado unos *banners* que se despliegan en ciertas situaciones como un elemento de la interfaz gráfica y enseñan un mensaje al jugador para informar de algún tipo de situación del juego. Estos *banners* están controlados por diferentes *scripts* y el funcionamiento es muy básico para todos: aparecer, quedarse y desaparecer en unos intervalos de tiempo definidos. Las situaciones donde se pueden encontrar estos elementos son cuando el jugador llega a un punto de control y al superar el nivel ya sea en el modo normal, contrarreloj o derrotar al jefe del cuarto nivel.

Figura 24 – *Banner* punto de control

4.16 Modo contrarreloj

La elección de añadir este modo juego se debe a la necesidad de añadir un reto extra para el jugador. Dentro del género de plataformas suelen añadirse retos o modos de juego que tienen como papel principal el tiempo. Ya sea el de completar objetivos, derrotar enemigos o superar el nivel en un tiempo determinado, ir a contrarreloj añade otro tipo de experiencia al juego que se complementa francamente bien con este tipo de género. Poner al límite las habilidades que se han aprendido durante el juego siempre puede resultar retador y gratificante.

Centrándose en este proyecto, como se puede observar en la planificación, este desarrollo no estaba contemplado en las tareas iniciales, aun así, llegados a un punto del desarrollo del proyecto, se pensó que era importante poder añadir un modo contrarreloj como modo de juego adicional para aportar una dimensión más a la jugabilidad.

Ciertamente, una vez teniendo los niveles funcionando y la gestión de vida-muerte del jugador, añadir una forma básica de modo contrarreloj no suponía un gasto de tiempo y recursos elevados, pero sí aportaba algo nuevo e interesante al videojuego.

El modo contrarreloj implementado es bastante sencillo, se activa dentro del nivel al coger un ítem en forma de reloj que se ubica al principio de los niveles. Acto seguido, se entra en el modo contrarreloj, aparece un marcador de tiempo en la interfaz gráfica (**Figura 23**), se elimina el marcador de puntuación, el marcador de vidas del jugador, los ítems para recolectar y los puntos de control (pero no los enemigos) y el tiempo empieza a contar. A partir de aquí el jugador debe llegar al final del nivel lo antes posible para obtener la mejor marca de tiempo posible.

La gestión de las características de este modo se hace conjuntamente entre el *script* TimeTrial acoplado al gameObject del reloj y el GameManager. Por un lado, el componente TimeTrial prepara la escena como se ha descrito anteriormente para que aparezcan y desaparezcan los elementos de este modo. También gestiona la contabilización del tiempo y en la parte final cuando el jugador supera el nivel se muestra el *banner* correspondiente y una pantalla con el tiempo establecido conjuntamente a los tres mejores tiempos del nivel hasta el momento. Si se establece un mejor tiempo del top tres del nivel, el componente TimeTrial gestiona, a través del GameManager, la actualización de la información que se guarda en el fichero “levels.xml”.

Por otro lado, el GameManager gestiona la muerte del jugador cuando ocurre en este modo, no restando ninguna vida y reiniciando completamente la escena en su modo normal de juego. Esta última situación es debida a que es más fácil volver a cargar la escena en su modo normal (que es su estado natural) que intentar volver a “recolocar” y volver a poner todos los elementos de la escena en su estado anterior en el modo normal de juego. Posiblemente, para un juego con grandes requerimientos técnicos en la que cada escena cuesta de cargar, esta última opción podría ser la más factible para evitar tiempos de espera elevados al jugador. En este caso, volver a cargar la escena del nivel no penaliza prácticamente nada la experiencia del jugador.

Otros *scripts* adicionales complementan y ayudan a la gestión del modo contrarreloj: LevelEnd y TimeTrialCompletedBanner. Este modo se sustenta en el flujo y estructura del juego que ya vienen dados por el modo normal de juego, así que los complementos que ya gestionan ciertas partes de este modo normal acogen una parte también del modo contrarreloj.

Finalmente, otro aspecto relacionado al modo contrarreloj se puede observar en el menú de selección de niveles. Comentado anteriormente, en este menú existe la posibilidad de abrir una pequeña ventana emergente que muestra estadísticas del nivel, entre ellas los tres mejores tiempos. El componente encargado de gestionar el menú con la

ayuda del GamManager recuperan y muestran esta información desde el archivo de xml de niveles. Los tiempos se guardan en minutos, segundos y milésimas de segundo que es una escala de tiempo bien proporcionada para la magnitud de los niveles del juego.

5. Diseño de niveles

5.1 Nivel 1

- Diseño del nivel

El primer nivel del juego quiere servir como una introducción suave al mundo y las mecánicas del ninja. Al ser un juego de plataformas lo más importante al principio es que el jugador se familiarice con la mecánica de salto para así ir incrementando la dificultad en los niveles introduciendo nuevos condicionantes a esta mecánica. Es por ello que, en el primer nivel, aparte de ser el de menor extensión, la mayoría de las dificultades son saltos entre huecos, saltos a plataformas de diferentes tamaños y salto de desniveles (las últimas plataformas del nivel son de las más pequeñas para añadir dificultad).

Otro factor importante en este nivel es enseñar al jugador la peligrosidad de los pinchos metálicos, los cuales matan al instante. Caer por un hueco o tocar los pinchos es muerte asegurada. Por este motivo hay un par de situaciones donde el jugador debe superar unos pinchos saltando por encima y otros deslizándose por debajo. Esta última situación introduce el deslizamiento y su mecánica y sobretodo quiere enseñar que el deslizamiento no es indefinido, sino que tiene un rango determinado. Si se calcula mal el jugador puede quedarse debajo de los pinchos y morir.

Por otro lado, también se introducen otros elementos como un enemigo, las gemas que se pueden recolectar y los puntos de control. Este último elemento pone en conocimiento del jugador que si muere en alguna parte del nivel podrá continuar desde el último punto de control que haya alcanzado.

Todos estos elementos básicos se presentan al jugador de forma paulatina y suave, dando tiempo al jugador para que los asimile. La estética del nivel es la de un escenario soleado con fondo llamativo y piezas de nivel acorde a la escena para empezar el juego en buenas condiciones.

- Guía del nivel

Para superar le nivel solo hay que seguir avanzando hacia la derecha e ir superando los obstáculos y al enemigo. Hay que aprovechar cuando el enemigo deja de atacar para herirlo y conseguir matarlo. Con un par de golpes caerá derrotado. Tras pasar la primera cascada es posible tomar otro camino ascendiendo a través de unos salientes. Arriba existen algunas trampas de pinchos, pero se pueden recolectar más gemas que yendo por el camino de inferior. Finalmente, el camino se volverá a unir

con el que viene por abajo y tocará saltar una serie de plataformas para llegar al final del nivel. Estas últimas plataformas son de menor ancho así que habrá que calcular bien el salto para no caer a los pinchos.

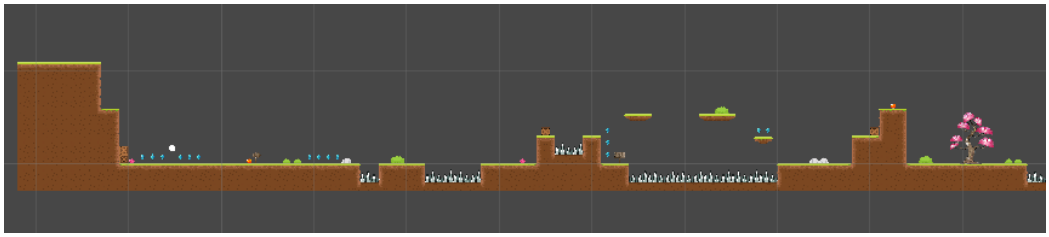

Figura 25 – Primera parte del nivel 1

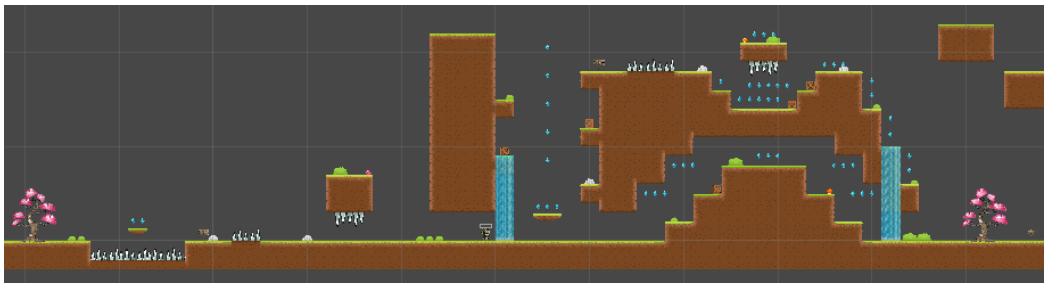

Figura 26 – Segunda parte del nivel 1

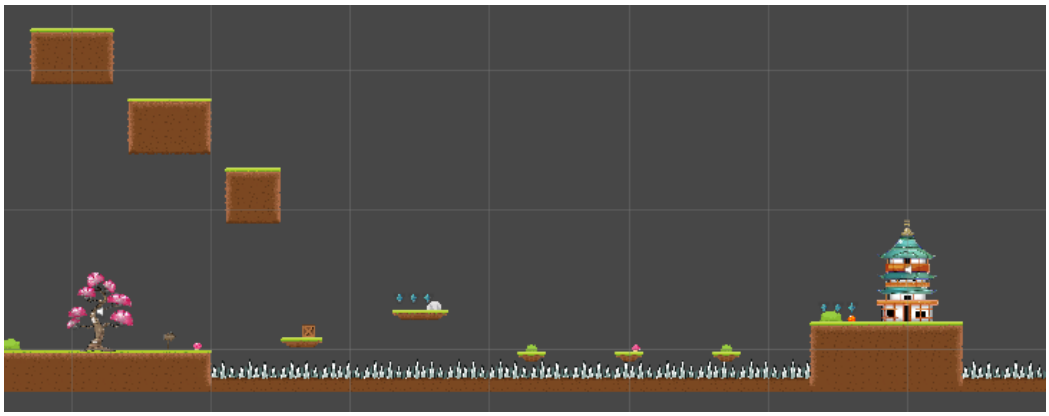

Figura 27 – Tercera parte del nivel 1

5.2 Nivel 2

- Diseño del nivel

En el segundo nivel del juego el objetivo es seguir familiarizándose con las mecánicas e introducir trampas (bola de fuego) y situaciones más complicadas de superar con las plataformas. De esta forma se pretende que el jugador vaya evolucionado en su curva de aprendizaje progresivamente. Además de las trampas se introduce también el trampolín con su versión de dirección fija.

El nivel cuenta también con objetos para que el jugador los recolecte y añade dos nuevos tipos de gema. El primer tipo proporciona más puntuación al jugador cuando se coge y también se encuentra en menor proporción en el nivel. De esta forma el jugador podrá valorar que ítems son más preciados. El segundo tipo es una gema que recupera parte de

la vida del personaje, así el jugador sabe que también existe la posibilidad de recuperar vida y tener más posibilidades de superar el nivel.

Se incorporan más enemigos en este nivel para que el jugador pueda acostumbrarse a luchar con ellos. Los enemigos se encuentran en zonas claves que dificultan el avance del jugador pero que se pueden superar sin poner en demasiados aprietos al jugador.

La extensión del nivel también se amplía en bastante medida y hay más cambios de altura que en el primer nivel. De esta forma se puede dar la sensación de que el nivel es más grande. Además, se incorporan algunas desviaciones en el camino principal con la intención de que el jugador las explore y recoja más gemas, considerando el peligro que supone tomar estos caminos alternativos. Estos caminos diferenciados del principal suelen encontrarse en los juegos de plataformas y suelen ser un poco más arriesgados, pero también suelen aportar mayores recompensas para el jugador (en vidas, puntuación, ítems, …).

La estética de este segundo nivel puede recordar al atardecer. En comparación con el primero parece que el día va avanzando y los niveles se adaptan. Los elementos para construir el nivel también van acorde a la estética del escenario.

- Guía del nivel

Empezar avanzando hacia la derecha para subir por las plataformas para seguidamente encontrar el primer trampolín. Saltando sobre él se llega a una zona elevada con gemas plateadas de más valor que las turquesa. Para bajar se puede saltar un poco más a la izquierda del hueco para que en la caída se puedan recolectar más gemas. Siguiendo hacia la derecha se encuentran las primeras plataformas con trampas. Cuidado con las bolas de fuego, no harán caer al jugador, pero sí que le harán daño. Seguir con el pequeño ascenso por plataformas que llevará al jugador al primer punto de control.

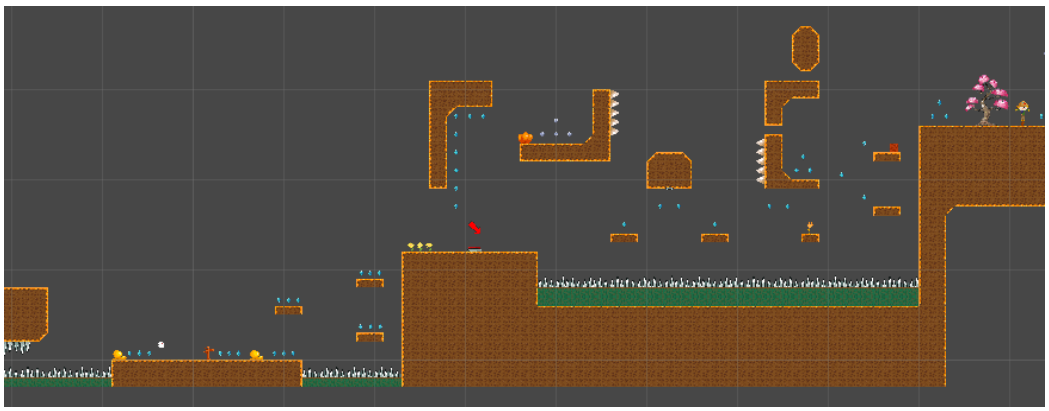

Figura 28 – Primera parte del nivel 2

Continuar ascendiendo por las plataformas hasta el árbol anaranjado. Para superar el salto que viene a continuación es preciso caer en

paralelo a la pared siguiendo el rastro de gemas turquesas para rebotar en el trampolín que se encuentra abajo del todo. En ese momento el jugador saldrá impulsado hacia arriba y a la derecha superando los pinchos metálicos. Pero para acabar de superar los pinchos será necesario hacer un doble salto en el momento oportuno.

Superado este salto el jugador se encuentra con una plataforma y un par de trampas de bolas de fuego. Hay que esperar al momento oportuno para poder saltar hacia el extremo derecho esquivando las bolas de fuego. Para salvar el salto siguiente se debe intentar seguir el camino de gemas turquesa. El salto simple es la forma más óptima de conseguir pasar este hueco, aunque se puede realizar también con un salto doble o planeando. Antes de llegar al segundo punto de control hace falta derrotar a un enemigo y pasar una trampa. Esta trampa escupe bolas de fuego y tiene pinchos metálicos por lo que hay que correr en el momento correcto para esquivar la bola de fuego y tener suficiente velocidad para deslizarse por debajo de los pinchos.

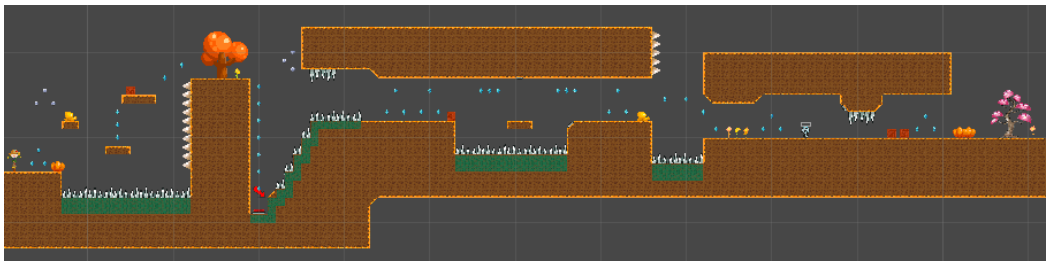

Figura 29 – Segunda parte del nivel 2

Tras alcanzar el punto de control hay una gema para recuperar vida si se ha sido herido por el camino. Siguiendo hacia la derecha se llega a un trampolín y un hueco extenso que no es posible superar saltando. Hay que saltar sobre el trampolín y seguir la trayectoria del impulso hasta ver una pequeña plataforma con otro trampolín. Al caer encima el jugador será impulsado hacia arriba, pudiendo recoger en el camino más gemas. Arriba del todo hay unas cuantas gemas plateadas que para conseguirlas no basta con el impulso del trampolín, hay que doble saltar para alcanzarlas.

Seguir hacia la derecha para superar unos cuantos saltos y ascender por las plataformas hasta la siguiente parte del nivel. Cuidado con la trampa de bolas de fuego en la parte superior de las plataformas. Arriba del todo se encuentra otro enemigo al que habrá que derrotar. Una vez se haya acabado con el enemigo el jugador podrá ver dos caminos a seguir, uno en sentido hacia la derecha y el otro hacia abajo. A la derecha se encuentra una trampa de bolas de fuego con pinchos metálicos, parecida a la anterior pero esta vez uno no se puede fiar ya que en el extremo derecho también hay un disparador de bolas de fuego. El camino inferior lleva a una serie de trampas de pinchos que para superarlas hay que moverse con cuidado. También habrá otro enemigo al que derrotar, pero bastantes gemas que recoger, entre ellas gemas plateadas. Un trampolín servirá al jugador para volver arriba y tras superar unos desniveles se llegará al punto de control.

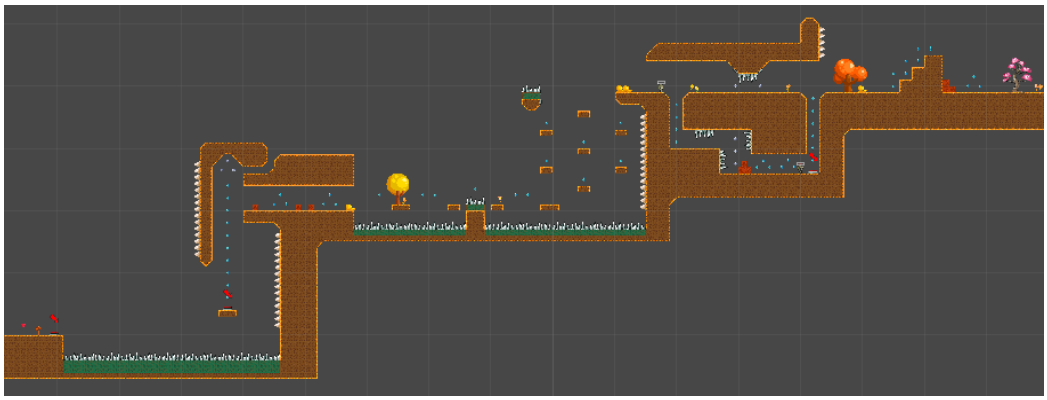

Figura 30 – Tercera parte del nivel 2

Después de este último punto de control y una gema para recuperar vida, llega la parte final del nivel. Superar con cuidado los saltos y trampas de bolas de fuego y descender por el desnivel de escalones hasta encontrar un trampolín. Una vez se salte sobre él el jugador será impulsado hacia arriba donde verá unos pinchos a la derecha que no se pueden superar ni doble saltando ni planeando. Es necesario impulsarse en el trampolín de la pared de la izquierda para superar esta trampa.

Finalmente se llega a la zona de saltos, plataformas y pinchos metálicos. Para superar esta parte será necesario saltar con cuidado y quizás utilizar un poco de planeo para aterrizar bien ya que las plataformas son estrechas. Es importante coger buen impulso en el salto para superar los pinchos intermedios y tener cuidado con el último salto ya que hay trampas de bolas de fuego. Una vez superado todo se llega por fin al final del nivel.

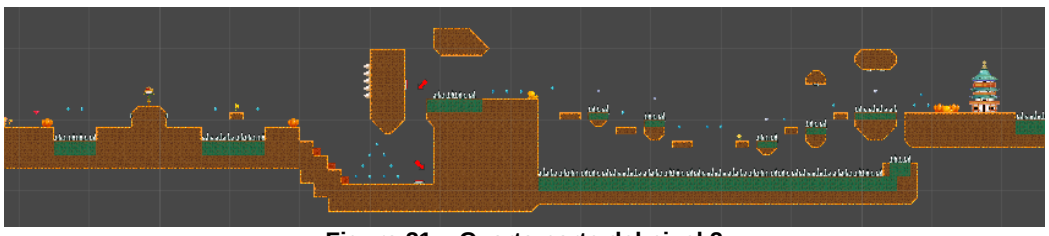

Figura 31 – Cuarta parte del nivel 2

5.3 Nivel 3

- Diseño del nivel

El tercer nivel sigue en la misma línea que los anteriores, incorporar más elementos para que el jugador los conozca y siga progresando en su curva de aprendizaje del juego. En este nivel, el más extenso de todos, se encuentran plataformas móviles, zonas de impulso para planear, nuevas trampas, enemigos y más tipos de gemas.

Las plataformas móviles aportan más jugabilidad, retos y posibilidades al nivel. Estas plataformas pueden conectar zonas del nivel de diferentes maneras o crear situaciones más complejas de superar. Las nuevas

trampas son las hachas y pinchos colgantes que oscilan de un lado a otro. Aquí el jugador deberá aprender a superarlas y sortear las combinaciones de éstas con otros elementos del nivel. Los enemigos seguirán interponiéndose en el camino del jugador en sitios molestos y que pueden ser más difíciles de superar. El nuevo tipo de gema amarilla otorga más puntuación que cualquiera de las anteriores.

La idea del nivel es intentar combinar todos los elementos para crear situaciones más complejas y que el jugador pueda disfrutar al superarlas. La curva de habilidad se llevará a otro nivel para hacer que el jugador se esfuerce y perfeccione más los movimientos ante los retos de este escenario. También se incorporan rutas alternativas para diversificar un poco los caminos a seguir dentro del nivel.

A medida que se incorporan nuevos elementos al juego se intenta que pueden combinarse entre ellos para sorprender al jugador y que no se confíe o se aburra. El incremento de dificultad en este nivel está pensado precisamente para que el jugador se supere a sí mismo y mejore sus habilidades con el personaje principal del juego.

La estética de este tercer nivel ha evolucionado a un escenario de noche con una gran luna como fondo. Se puede seguir la evolución del día desde el primer nivel hasta este tercero. La variedad en la estética de los niveles aporta frescura a la experiencia de juego.

- Guía del nivel

En la primera parte del nivel se encuentran una serie de plataformas móviles que harán avanzar al jugador hacia la derecha. Hay plataformas horizontales y verticales. Llegados a un punto aparece la primera zona de impulso para planear. La zona sirve para salvar el desnivel y acceder a la parte superior. Desde aquí se puede saltar un momento a la izquierda para recoger las primeras gemas amarillas.

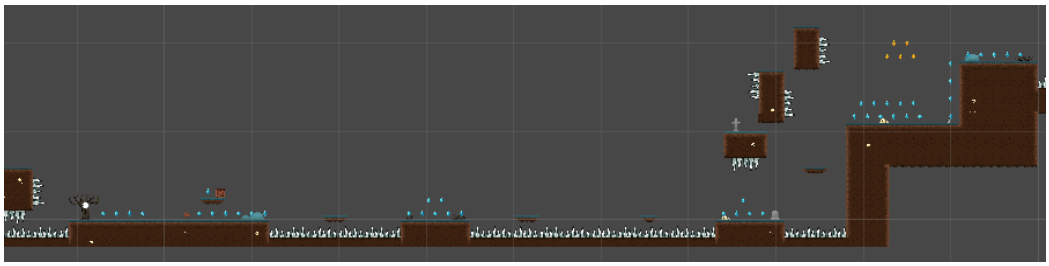

Figura 32 – Primera parte del nivel 3

Al continuar hacia la derecha, cuidado con la plataforma móvil cuando pasa por debajo de la trampa de bolas de fuego. Se puede esquivar con un pequeño salto o moviéndose en el momento correcto dentro de la misma plataforma. Después de superar el salto de los pinchos metálicos y coger la siguiente plataforma móvil se puede ascender por la zona de impulso para llegar a la siguiente sección. Cuidado para no caer porque el aterrizaje puede ser en el hueco de pinchos y morir al instante.

Otra trampa de bolas de fuego espera al jugador arriba del desnivel y ésta da paso a una serie de plataformas y más trampas por las que hay que ascender. La plataforma móvil superior puede llevar al jugador en dos direcciones. Hacia la derecha y tras superar alguna trampa más se llega al primer punto de control. Hacia la izquierda el jugador toma un camino alternativo que le llevará a superar varios retos, pero podrá conseguir muchas gemas, entre ellas bastantes amarillas. En este camino alternativo se combinan varias trampas y plataformas que no lo pondrán fácil para avanzar. También hay un enemigo para añadir más dificultad. Si se muere sin haber llegado al punto de control habrá que empezar desde el principio del nivel. Una posibilidad es llegar primero al punto de control y luego volver atrás con la plataforma para realizar este camino alternativo y si pasa algo no tener que empezar desde el principio. En el camino alternativo, cuidado con el trampolín ya que, aunque permite subir el desnivel, hay una zona de pinchos que debe superarse saltando otra vez. Antes de llegar al final del camino el jugador deberá derrotar a un enemigo y poder avanzar. Al final de este camino el jugador debe caer por un hueco recogiendo una gema de vida y gemas amarillas para llegar al principio del camino alternativo y poder seguir a la derecha.

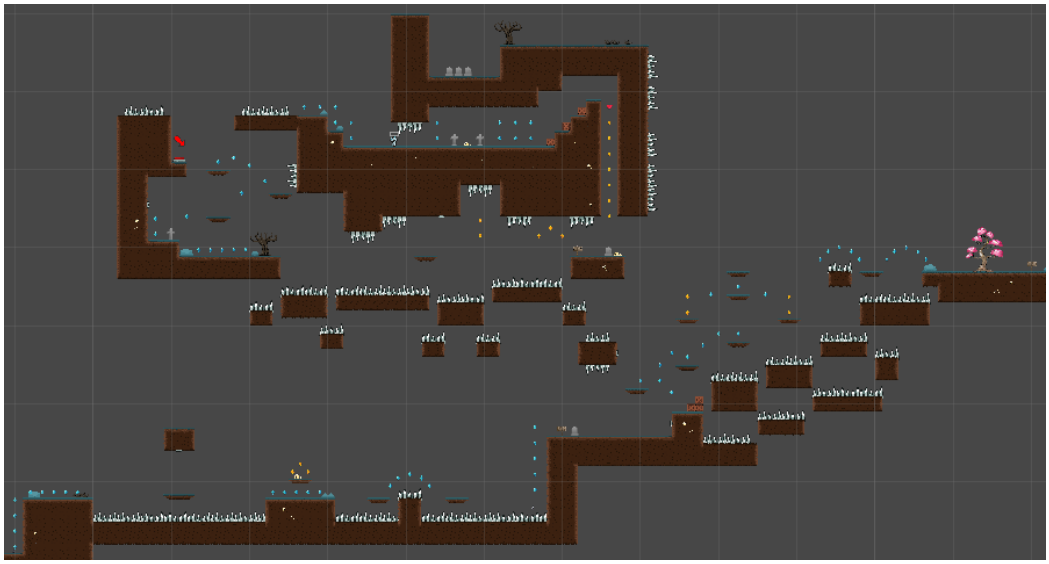

Figura 33 – Segunda parte del nivel 3

Después del primer punto de control entran en escena las primeras trampas oscilantes. La primera está pensada para superarse saltando de forma que se puede seguir el camino de gemas. La segunda trampa puede superarse deslizándose por abajo tal y como indica el camino de gemas. Se puede intentar cualquiera de las dos formas en las dos trampas, aunque las más óptimas están marcadas por el camino de gemas. Antes de llegar a la tercera trampa de este estilo, hay que derrotar a un enemigo con la sistemática habitual. La trampa posterior en esta zona del nivel incluye un hueco con pinchos metálicos donde hay calcular bien el salto para no caer o topar con la trampa oscilante. Seguidamente, se llega a una sección de plataformas móviles y alguna

que otra trampa de bolas de fuego, cuidado al saltar a la primera plataforma para que el jugador no salga herido por la bola de fuego.

La última parte de esta sección del nivel presenta al jugador una combinación de saltos con descenso y trampas de pinchos. Hay que vigilar al saltar de no tocar los pinchos de las paredes y del suelo al aterrizar. Es recomendable usar el planeo para superar con éxito estos retos. Una última combinación de trampas con pinchos y bolas de fuego es el último obstáculo antes de llegar al siguiente punto de control. Hay que tener cuidado y calcular bien el salto y cuando realizarlo para no colisionar con las bolas de fuego ni tocar los pinchos metálicos. Antes del punto de control espera un enemigo con ganas de molestar en el camino.

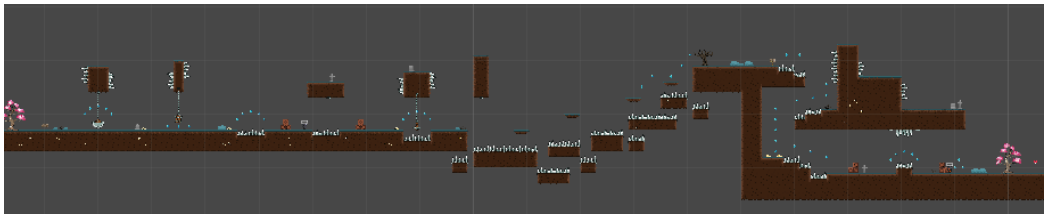

Figura 34 – Tercera parte del nivel 3

Segundo punto de control alcanzado. El salto para llegar a la plataforma de la derecha es peligroso y oportunista, hay que esperar a que la plataforma móvil llegue y saltar encima de ella. Al final del camino de la plataforma se encuentra una nueva zona de impulso para planear. La idea es ascender para conseguir superar el desnivel, pero hay que vigilar con el camino que se tiene que recorrer planeando, hay pinchos y desplazamientos dentro de la zona que son difíciles y pueden llevar a una muerta segura.

Pasado el árbol, para llegar a las plataformas y seguir el camino es necesario esquivar el hacha oscilante y caer bien en la primera plataforma. Tras ascender por las plataformas se llega a una zona elevada donde hay que luchar contra un enemigo para continuar el camino. Una vez derrotado llega una sección difícil de superar. Un salto descendiente con pinchos arriba y abajo que, aunque parezca fácil, es mejor no confiarse y saltar con margen para llegar hasta el otro extremo sano y salvo. Una combinación de salto con planeo debería bastar para superar los pinchos (cuidado de no saltar muy alto).

Antes de llegar al siguiente punto de control hay que planear en una zona de impulso más extensa de lo habitual. Primero hay que seguir hacia la derecha hasta encontrarse con la pared de pinchos (intentando conseguir las gemas amarillas) y al llegar hay que dejarse caer siguiendo las gemas turquesa. Cuidado porque la caída no puede ser indefinida, cuando se haya descendido un poco hay una plataforma móvil que se utiliza para conectar con la siguiente área. Si la plataforma no está en el momento de descenso, habrá que controlar el planeo y esperar a que vuelva sin caer. Una vez encima de la plataforma, se llega a otra zona de impulso para planear con pinchos en la parte de arriba.

Recoger las gemas amarillas avanzando hacia la derecha hasta encontrar una situación similar a la anterior. Esta vez habrá que descender, pero no hay ninguna plataforma esperando, sino que cuando se vea la apertura en la parte derecha hay que moverse hacia esa dirección para llega a tierra segura. Ahora sí, el último punto de control.

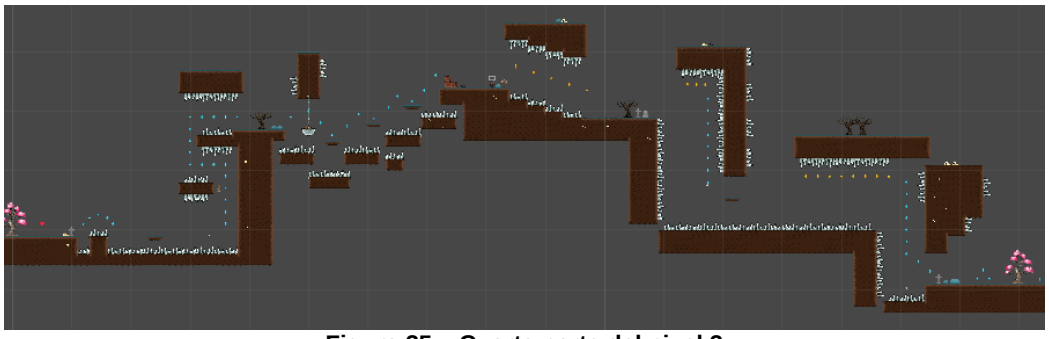

Figura 35 – Cuarta parte del nivel 3

Una vez se llega al último punto de control un enemigo espera impaciente para bloquear el camino al jugador. Cuando se consiga eliminarlo se entra en la parte final y más desafiante del nivel. Esta última sección combina prácticamente todos los elementos anteriores para poner a prueba al jugador.

Para seguir el camino hay que ascender por los trampolines que hay incrustados en la pared. Si se salta correctamente el jugador será impulsado hasta la parte superior izquierda. Se puede intentar rectificar la trayectoria con un pequeño salto en el momento debido, pero cuidado con los pinchos de las paredes. Una vez arriba, saltar por encima del trampolín para ir un poco más a la izquierda y recoger valiosas gemas amarillas. Luego saltar sobre el trampolín para llegar a la parte derecha (se puede usar un doble salta si se ve que no se va a llegar bien al desnivel de la parte derecha). Seguir hacia la derecha superando las trampas de pinchos y oscilantes hasta la plataforma móvil. Cuidado porque en el camino de la plataforma hay una trampa de bolas de fuego que se puede esquivar saltando. Seguidamente, coger la plataforma móvil para conectar con las plataformas dispuestas de manera ascendente para superar el desnivel.

Tras superar la trampa combinada de bola de fuego y pinchos metálicos, hay que coger la plataforma móvil la cual tiene un trampolín. Al final de su recorrido será el momento de saltar sobre el trampolín y ascender entre las paredes de pinchos, hay que intentar saltar de manera vertical. Aquí no hay que despistarse pues para poder llegar a la plataforma que hay justo arriba será necesario ejecutar un doble salto ya que el impulso del trampolín no es suficiente para hacer llegar al personaje hasta la plataforma en la parte superior.

Una vez arriba, tocará combinar las plataformas móviles para seguir avanzando hacia la derecha del nivel. En una de las plataformas móviles será necesario deslizarse para evitar los pinchos que cuelgan en una sección del recorrido. Tras esquivar esta trampa, solo queda el último

salto que hay que medir bien para llegar otra a vez a tierra firme. Unos pinchos metálicos rellenan el hueco y pondrán la dificultad a este último salto. Después de un largo recorrido se llega al final del nivel.

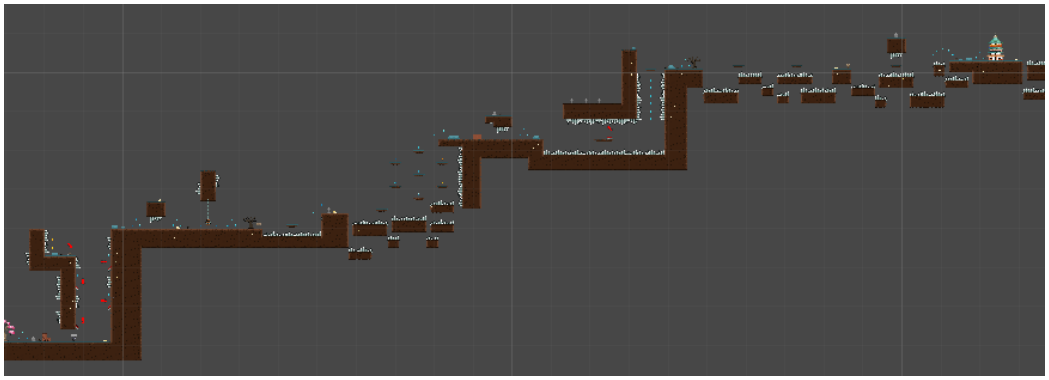

Figura 36 – Quinta parte del nivel 3

5.4 Nivel 4

- Diseño del nivel

El cuarto nivel se diferencia de los anteriores en que el personaje principal debe enfrentarse en una batalla a un enemigo más fuerte que los que se pueden encontrar en los niveles anteriores, un jefe de nivel. Se ha reutilizado la estética del primer nivel para dar al jugador una sensación de conocimiento del escenario y pueda tener un poco más de seguridad para afrontar la batalla.

Para el diseño del nivel se ha tenido en cuenta que debe ser un espacio cerrado donde prácticamente la cámara abarque en horizontal la escena y que tenga un pequeño desplazamiento vertical que se utilizará para unas plataformas por las que podrá subir el jugador. Como se ha explicado en el capítulo anterior, el jefe de nivel (con estética samurái) posee tres fases de ataque y en una de ellas se sirve de las plataformas antes mencionadas para no estar al alcance del jugador mientras ataca con los kunai. El personaje no debe ser capaz de rehuir la batalla por eso el espacio es cerrado y sin posibilidad de escapar, la única forma es derrotando al enemigo.

En la primera y tercera fase de la batalla el nivel se presenta igual, solo las paredes laterales que impiden al jugador escaparse. En la segunda fase, las plataformas que usa el enemigo aparecen en escena.

Figura 37 – Nivel 4 en la fase 1 del enemigo

Figura 38 – Nivel 4 en la fase 2 del enemigo

- Guía del nivel

Para superar el nivel hay que vencer al samurái enemigo en sus tres fases. En la primera fase la mejor forma es saltar por encima de él cuando se aproxime y en sentido contrario a su movimiento. Después de las tres repeticiones, cuando esté recargando la barra de ataque aprovechar para herirlo y repetir esta táctica dos veces.

En la segunda fase aparecerán unas plataformas en el centro de la escena y el enemigo se subirá en la más alta. Las plataformas inicialmente tienen pinchos metálicos y son muerte asegurada si se intenta subir a ellas. El enemigo empezará a lanzar kunais en la posición que esté el jugador así que hay que moverse de un lado a otro para esquivarlos (parece fácil, pero si uno se descuida pueden alcanzar varios kunais al jugador). Atención a la barra de ataque del samurái pues

cuando se agote una plataforma central emergerá a la par que los pinchos de las demás desaparecerán. En este momento hay que subir hasta donde se encuentra el enemigo y herirlo. Tras conseguirlo hay que descender rápidamente al suelo ya que los pinchos parpadeantes anuncian que volverán a reaparecer en poco tiempo y si el jugador es atrapado morirá al instante. Repetir la táctica dos veces para llegar a la última fase de la batalla. Si no se consigue herir al jefe de nivel habrá que repetir una vez más la secuencia para poder pasar a la siguiente fase.

En la tercera y última fase el samurái enemigo levita en una posición aleatoria en el cielo mientras recarga la barra de ataque. Una vez se llene del todo descenderá a gran velocidad para atacar al jugador así que hay que esquivarlo deslizándose o saltando sobre él. No hay que bajar la guardia porque una vez en el suelo, el enemigo ejecutará dos embestidas para atacar al jugador así que nuevamente hay que esquivarlo saltando. Tras las embestidas será el momento de aprovechar para herir al enemigo y tras dos veces se conseguirá derrotarlo y superar el nivel.

6. Manual de usuario

6.1 Requerimientos técnicos para jugar

Teniendo en cuenta las dimensiones de este proyecto y los recursos tanto gráficos como sonoros, el juego resultante debería poder ejecutarse en la gran mayoría de dispositivos con un mínimo de requerimientos.

Puesto que principalmente el juego tiene como objetivo las plataformas tipo PC que ejecuten Windows o Mac OS (y por extensión también Ubuntu), las siguientes características harán referencia a estas plataformas. Ante la imposibilidad de poder realizar pruebas en diferentes equipos con diferentes configuraciones y recursos de hardware/software se ha decidido indicar aquellos requerimientos mínimos que aconseja Unity dentro de la generalización que abarca su motor de desarrollo [8]:

- Sistema operativo
  - Windows 7 SP1+
  - Mac OS 10.12+
  - Ubuntu 16.04+
- CPU con soporte para conjunto de instrucciones SSE2
- GPU tarjeta gráfica con DX10 (modelo de shader 4.0)
- Tamaño en el disco aproximadamente 500 Mb (depende del sistema operativo)
- Teclado o mandos PS3/PS4 (cable o wireless)

En cuanto a la tarjeta gráfica, posiblemente el juego es capaz de funcionar con versiones inferiores a DX10.

Al final los requerimientos mínimos para obtener una experiencia decente de juego variarán en función de cómo esté hecho el videojuego y de si se ha optimizado de alguna forma en particular para cada plataforma donde puede ejecutarse. En este caso, se ha optado por dejar que Unity haga las configuraciones básicas de requerimientos y optimización en función de la plataforma de destino (que para este tipo de proyectos en general suelen rendir bien). Cuando se realiza la *build* del juego, Unity ofrece diferentes plataformas de destino, modos de publicación, opciones de compresión, etc. que afectarán a la ejecución del juego y por ende a los requerimientos técnicos mínimos.

Aun así, para un proyecto comercial o para el público general esta parte de generación del juego y requerimientos técnicos debe cobrar una importancia elevada ya que el éxito del videojuego puede depender, en gran medida, de su *performance* en los dispositivos de los jugadores. Un mal rendimiento o una mala optimización de recursos puede significar el

fracaso para el juego, aunque sea excepcionalmente bueno o innovador. Por otro lado, a cuantos más dispositivos se pueda llegar manteniendo una experiencia homogénea entre todos ellos, más posibilidades de éxito y ventas tendrá el videojuego. Aquí juega un papel importante el tema de los requerimientos técnicos y la optimización específica por plataforma para sacar más partido de las capacidades particulares de cada una.

6.2 Instrucciones del juego

- Iniciar el juego

Para iniciar el juego, ejecutar la *build* del mismo (dependiendo del sistema operativo se tendrá un fichero .exe o diferente). Después de la escena de presentación, el juego mostrará el menú principal. En él se pueden escoger varias opciones.

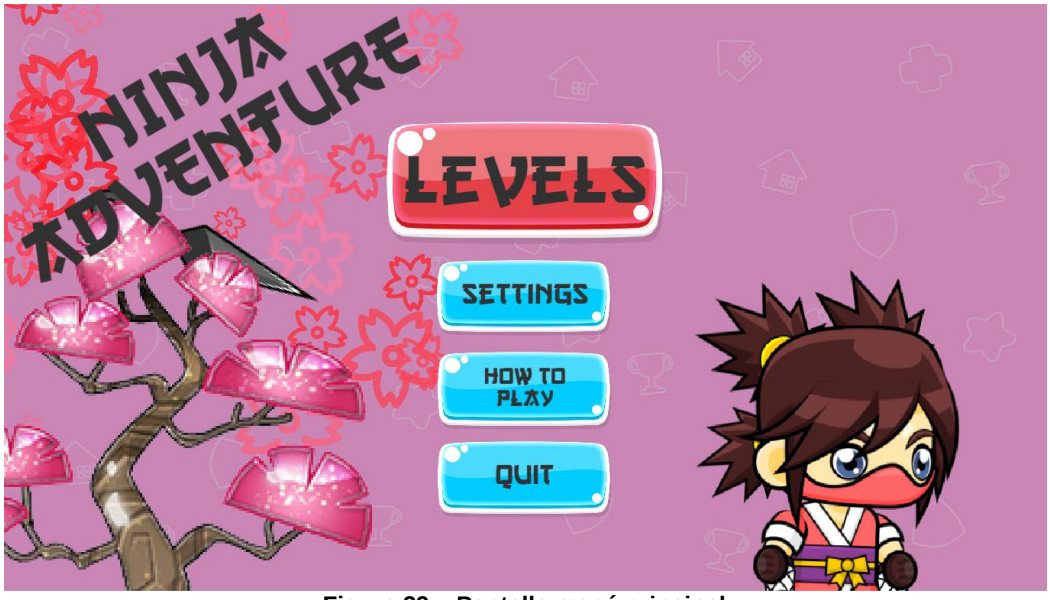

Figura 39 – Pantalla menú principal

- Objetivo del juego

Como aprendiz de ninja, la misión será superar todos los niveles para entrenar y perfeccionar tus habilidades ninja y llegar a ser el mejor maestro ninja entre todos los clanes del mundo.

Alcanzar el final del nivel es tu objetivo principal, pero las dificultades físicas, trampas y enemigos no te lo pondrán nada fácil. Cuentas con 5 vidas que irás perdiendo a medida que mueras o te hieran. Llega hasta un punto de control para no tener que empezar desde el principio si tienes mala suerte y caes derrotado (si pierdes todas las vidas sí deberás empezar el nivel desde el principio). Además, consigue cuanta más puntuación mejor recogiendo las gemas que hay esparcidas por los niveles. Una gema en concreto recuperará parte de tu vida para poder seguir con la aventura.

Figura 40 – Templo final de nivel (izquierda) | Árbol punto de control (derecha)

Durante el camino para convertirte en un maestro ninja, te encontrarás con muchos enemigos que aparecerán en niveles concretos dentro del juego. En estos niveles la prioridad será derrotar al jefe sin morir en el intento, seguro que ellos irán a por todas para no dejarte avanzar.

Una vez superes un nivel, intenta conseguir la mejor puntuación recogiendo todas las gemas posibles. Hay diferentes gemas cada una con puntuación diferente. Si además te sientes con ganas de aceptar más retos, el modo contrarreloj se habilitará una vez hayas superado el nivel. En este modo completa el nivel lo antes posible sin morir una sola vez. Tranquilo, este modo no descuenta una vida si mueres en el intento. Consigue las mejores marcas de tiempo e intenta superarte cada vez.

Figura 41 – Reloj prueba contrarreloj

¿Te has quedado sin vidas? No te preocupes, el camino para ser un maestro ninja no es fácil, por eso se te otorgarán 5 vidas más para que puedas seguir con la aventura.

- Controles

Desplazamiento: el personaje se puede desplazar a izquierda o derecha a voluntad del jugador. Para recorrer el nivel y llegar hasta el final será necesario desplazarse.

Salto: el personaje puede saltar tanto estando quieto como en desplazamiento. Usa los saltos para salvar obstáculos y agujeros o subir a otros elementos del nivel.

Doble salto: si el salto no es suficiente, un ninja puede realizar un segundo salto en el aire. Esta habilidad única permite salvar mayores distancias o saltar más alto.

Deslizamiento: algunos obstáculos no se pueden franquear por arriba. Así pues, usa el deslizamiento para sortearlos. Cuidado, el deslizamiento tiene un alcance limitado, no te quedes debajo de los obstáculos peligrosos.

Planear: este as bajo la manga te permite usar una tela ninja para planear como si de una ardilla se tratase. Este truco ninja permite controlar mejor las caídas y salvar mayores distancias.

Atacar: para derrotar a los enemigos deberás usar el ataque con *katana* para herirlos y finalmente eliminarlos. Utiliza esta habilidad para superar a quien se interponga en tu camino.

Menús: el juego dispone de diferentes menús. Navega fácilmente por los menús, acepta opciones o vuelve hacia atrás.

Pausa: si necesitas un descanso en tu aventura, pausa el juego. Una vez te sientas con fuerzas de nuevo, reanuda el juego para seguir con la aventura.

La tabla siguiente muestra los controles del juego según el método de entrada físico conectado al dispositivo.

Control	Teclado	PS3	PS4
Atacar	M	Cuadrado	Cuadrado
Saltar	Espacio	Cruz	Cruz
Doble salto	Espacio x2	Cruz x2	Cruz x2
Desplazamiento	Flechas dirección	Pad dirección	Pad dirección
Deslizarse	Shift	R1	R1
Planear	G	L1	L1
Pausa	P	Start	Options
Aceptar	Enter	Cruz	Cruz
Cancelar	Escape	Redonda	Redonda
Menú navegación	Flechas dirección	Pad dirección	Pad dirección

Los mandos de consolas como PS3 o PS4 se pueden conectar por cable o *wireless* para poder jugar. Consultar referencias [11] y [12] para configurar correctamente el mando de PS3 en Windows y Mac OS respectivamente. Del mismo modo, consultar las referencias [13] y [14] para configurar correctamente el mando de PS4 en Windows y Mac OS respectivamente.

- Menús

    - *Menú principal*: da acceso al menú de selección de niveles, la configuración, instrucciones dentro del juego de como jugar y finalizar el juego.
    - *Menú de niveles*: acceso a los niveles desbloqueados. Dispone de opción para presentar una pantalla emergente con las estadísticas del nivel (puntuaciones y tiempos).
    - *Menú configuración*: permite configurar ciertos aspectos del juego como la pantalla completa, la resolución de la pantalla y ajustar el nivel global de volumen del juego.
    - *Menú como jugar*: escena articulada a modo de diapositivas donde el jugador puede ver un resumen de los aspectos básicos para empezar a jugar.
    - *Menú pausa*: dentro de los niveles, el jugador puede pausar el juego para acceder a este menú. Se permite reanudar el juego, volver al menú principal o finalizar el juego.

7. Conclusiones

En la época actual la industria de los videojuegos ha logrado posicionarse como uno de los sectores que más ingresos genera, incluso por delante del cine y la música juntos [15]. La evolución de la tecnología y los servicios de conexión en todo el mundo ha hecho posible que los videojuegos lleguen prácticamente a todos los rincones y dispositivos del mundo. Por otro lado, los videojuegos han ido evolucionando en las últimas décadas proporcionando actualmente una diversidad de productos con mucho rango, desde videojuegos con presupuestos millonarios hasta juegos *indie* con presupuestos casi inexistentes.

Cada vez más gente se siente atraída por una industria que no solo proporciona diversión y entretenimiento, sino que ofrece una gran variedad de salidas profesionales y oportunidades para muchas personas en el mundo. Crear un videojuego no solo es programar código o crear modelos 3D realistas, un videojuego es un producto que para producirlo es necesaria la colaboración interdisciplinar de muchos profesionales ya que, al fin y al cabo, un videojuego pretende simular un mundo, real o imaginario, pero que necesita de sus reglas e interacciones entre todos los elementos que se implementen. Al haber llevado a cabo este proyecto en la asignatura final del máster, uno se da cuenta que un videojuego, por sencillo que sea, requiere de una buena idea, una planificación, disponer de recursos tanto de desarrollo como gráficos, tiempo, dedicación y ganas de crear un producto que llegue a los jugadores para que puedan disfrutarlo.

En muchas ocasiones a lo largo del desarrollo del proyecto se ha tenido que replanificar las tareas ya sea por falta de tiempo o recursos o por ideas nuevas que no estaban contempladas en la planificación inicial, pero que han surgido al ir avanzando en el proyecto. Durante el proyecto se ha necesitado de una capacidad de adaptación para poder generar los recursos necesarios para implementar las ideas del juego. Ya sea por parte de recursos gráficos o implementaciones de código, los elementos del videojuego han tenido que trabajarse para llegar al resultado deseado y se ha dedicado mucho esfuerzo y tiempo a pulir errores y mejorar varios aspectos del videojuego para poder entregar un producto a la altura.

La experiencia que se ha tenido a lo largo de todo el proyecto se podría resumir en las siguientes lecciones:

- Un videojuego abarca muchas áreas de conocimiento y une diferentes tipos de perfiles que juntos logran encajar todos los elementos del mundo del videojuego para que funcionen en conjunto.

- El desarrollo de un videojuego es una tarea ardua y consume mucho tiempo, sobre todo en el desarrollo de recursos, implementación de las relaciones entre los diferentes elementos y todas las fases de prueba-error para verificar el correcto funcionamiento del conjunto del videojuego y sus elementos. Es importante ser detallista en el sentido de poder crear un producto con el mínimo de errores posibles para no degenerar la experiencia del jugador.
- Para llevar a cabo el proyecto y progresar de una manera correcta en el desarrollo es necesario planificar con antelación, pero también saber adaptarse a los imprevistos y cambios que puedan surgir a lo largo del desarrollo.
- En referencia al punto anterior, es importante contar con buenas herramientas de control de versiones y una buena metodología de trabajo. Un desarrollo del proyecto desorganizado y caótico puede ser la diferencia entre poder crear el videojuego o perderse en el camino y que nunca vea la luz.

Si bien se ha conseguido entregar un producto acabado y funcional, no todos los objetivos iniciales se han podido cumplir, aunque también se han implementado otras funcionalidades no previstas inicialmente. La causa más probable en cuanto a la no implementación de algunos puntos de la planificación es seguramente una mezcla entre ambición del proyecto, recursos y tiempo disponible. La habilidad para trepar por escaleras, lanzar kunais y las habilidades especiales se han quedado fuera del proyecto por el momento, pero han dado paso a un nivel que alberga la batalla contra un jefe, un nuevo modo de juego contrarreloj y a un producto mejor acabado tanto en la interfaz gráfica como en el funcionamiento en general. Se ha preferido dedicar más tiempo a asentar una base solida e íntegra del videojuego que pueda ser expandida en futuras actualizaciones con facilidad y siguiendo una buena estructura de diseño.

La planificación del proyecto en cuanto a tareas y tiempo cubría ampliamente todos los aspectos base para garantizar unos mínimos a la entrega del producto acabado. Para poder llevarlo a cabo, se ha hecho uso de herramientas de control de versiones para poder trabajar en paralelo en algunas tareas y llevar una mejor organización de las implementaciones. Una vez acabado el proyecto, es cierto que algunas tareas que se habían planteado en un principio ahora se replanificarían con otro criterio como es por ejemplo el diseño e integración de niveles o más tiempo para la interfaz gráfica. Con la experiencia se puede aprender mejor a refinar que tareas y que tiempos se necesitan para el proyecto en cuestión. Las tareas no se han podido seguir al 100% a causa de eventualidades y de que, a medida que avanza el proyecto y se empieza a tener una mejor visión global del conjunto de tareas y tiempo disponible, se deben realizar ajustes para reconducir correctamente el rumbo del desarrollo. También se han tomado decisiones en ciertos puntos del proyecto para desviar tiempo y recursos a mejorar la estructura del videojuego, mejorar las mecánicas

implementadas, pruebas y corrección de errores y nuevos modos de juego porque se ha pensado que era importante para la experiencia de juego. Aun así, en términos generales el camino que ha marcado la planificación inicial ha sido ampliamente seguido y ha propiciado la consecución de la gran mayoría de las tareas, como así demuestra el producto entregado.

Aunque el proyecto se ha acabado y entregado, el videojuego aun tiene mucho potencial para crecer y evolucionar. Las habilidades del personaje principal que no se han podido implementar son un buen ejemplo ya que aportarían más experiencia de juego y posibilidades para el mundo del videojuego. Se podrían añadir también más formas de atacar como combos o armas diferentes.

Más implementaciones en la línea de trampas, elementos del juego y enemigos se podrían explorar para dar más variedad y riqueza a los retos y situaciones en los niveles. Por ejemplo, se podrían implementar trampas como minas, descargas eléctricas entre dos puntos, plataformas inestables que o bien vuelquen o bien se desplomen transcurridos unos instantes, suelos con efectos que hagan deslizar al jugador hacia una dirección, …

En cuanto a niveles, las posibilidades son prácticamente infinitas y el juego está preparado para recibir de forma fácil actualizaciones con más niveles. Una idea que se ha quedado fuera del proyecto sería un nivel con temática helada donde haya zonas deslizantes. Otro nivel que no ha podido entrar en el desarrollo podría ser un nivel completamente aéreo, con zonas de impulso de planeo y otros elementos que propongan desafíos nuevos al jugador. Para cualquier nivel, se podría implementar un modo “difícil” que limitase ciertas habilidades como el doble salto o el planeo para así poner más interesante la superación del nivel. También se puede explorar la idea de implementar niveles especiales con bonus o con formas de juego diferentes que los niveles de tipo “normal”.

Los enemigos son otro foco de ideas. Pueden diseñarse e implementarse más enemigos con diferentes habilidades, tamaños y patrones incluso también enemigos aéreos. Los enemigos pueden tener ciertas características que hagan que solo se puedan atacar de una forma determinada o incluso con elementos del estilo fuego, agua, aire, tierra, etc. Esto abriría las puertas a personalización de accesorios como armas o escudos y la mejora de sus capacidades. Más jefes de niveles se pueden implementar como niveles especiales entre los niveles de tipo “normal”. Las posibilidades de integrar más enemigos que concuerden con la estética y el mundo del videojuego son grandes y variadas.

Como se puede observar, las líneas futuras de desarrollo para este juego son variadas, con muchas posibilidades y abarcan muchas áreas del videojuego. Los videojuegos actuales son productos en los que se pueda actualizar contenido o corregir errores, así que todas las ideas

tienen sus posibilidades de incorporarse al juego y hacerlo crecer (y así atraer y mantener a los jugadores jugando al videojuego).

Por último, aunque la realización de este proyecto y del máster en general ha supuesto un gran esfuerzo personal, todo lo aprendido y las experiencias vividas han valido la pena de principio a fin.

8. Glosario

2D: referente a los videojuegos, dimensiones en las que se puede jugar y se renderizan los gráficos dentro del juego. En este caso dos dimensiones.

2.5D: referente a los videojuegos, dimensiones en las que se puede jugar y se renderizan los gráficos dentro del juego. En este caso suelen combinarse elementos de dos y tres dimensiones para crear un efecto visual.

3D: referente a los videojuegos, dimensiones en las que se puede jugar y se renderizan los gráficos dentro del juego. En este caso tres dimensiones.

build: término inglés para referirse a la creación del ejecutable del juego o el ejecutable mismo.

consola: dispositivo generalmente doméstico para la ejecución de software de entretenimiento, es decir, videojuegos.

CPU: procesador lógico de un dispositivo electrónico.

frame: cada renderizado completo que se construye en el videojuego para mostrar la imagen por pantalla.

freemium: modelo de negocio en el que se deja al usuario un producto básico gratuito y se venden por separado otras características del producto.

gameObject: relativo a Unity, son todos los objetos que conforman la escena de juego, así como prefabs, componentes y otros recursos del motor.

GPU: procesador gráfico de un dispositivo electrónico.

kunai: arma utilizada por los ninjas parecida a un cuchillo.

PC: hace referencia a todos los dispositivos como ordenadores personales o portátiles.

performance: término inglés para referirse al rendimiento del juego en términos tecnológicos y de recursos.

plataformas (dispositivos): cuando hace referencia a dispositivos en el mundo de los videojuegos, son todos aquellos objetos electrónicos con la capacidad de ejecutar videojuegos.

plataformas (género): cuando hace referencia a un género de videojuegos, son todos aquellos juegos que reúnen las características básicas que los agrupan en esta categoría.

prefab: recurso preconstruido que puede crear Unity para su uso dentro del videojuego.

script: en referencia a los videojuegos, son archivos que contienen código donde se definen lógicas y comportamientos de componentes del juego. En Unity, muchas veces se puede intercambiar el término por componente, ya que los *scripts* se añaden a los gameObjects como componentes.

sprite: recurso gráfico utilizado en la creación de videojuegos.

tile: recurso gráfico en forma de *sprite* que suele utilizarse para la construcción de niveles/escenarios en juegos 2D. Suele ser *sprites* cuadrados que conforman suelos, paredes, etc.

trigger: interacción física entre dos elementos del juego que no se resuelve como un contacto físico sino como un aviso del contacto.

UI: interfaz gráfica, en este contexto del videojuego.

Unity: motor de desarrollo principalmente de videojuegos de uso libre en su edición estándar.

9. Bibliografía

[1] *Videojuego de plataformas* (2019) [en línea]. Wikipedia. [Consulta: 18 de mayo de 2020] <https://es.wikipedia.org/wiki/Videojuego_de_plataformas>

[2] *Five Critical Moments in Platform Game History* (2014) [en línea]. USGamer. [Consulta: 18 de mayo de 2020] <https://www.usgamer.net/articles/five-critical-moments-in-platform-game-history>

[3] *The evolution of platform games in 9 steps* (2017) [en línea]. Redbull. [Consultado: 19 de mayo de 2020] <https://www.redbull.com/in-en/evolution-of-platformers>

[4] *Todo sobre los juegos de plataformas. Parte 1: Los inicios* [en línea]. Retrogamers. [Consulta: 20 de mayo de 2020] <https://retrogamers.es/todo-sobre-los-juegos-de-plataformas-parte-1-los-inicios/>

[5] *Todo sobre los juegos de plataformas. Parte 2: Evolución* [en línea]. Retrogamers. [Consulta: 20 de mayo de 2020] <https://retrogamers.es/todo-sobre-los-juegos-de-plataformas-parte-2-evolucion/>

[6] *Todo sobre los juegos de plataformas. Parte 3: Madurez* [en línea]. Retrogamers. [Consulta: 20 de mayo de 2020] <https://retrogamers.es/todo-sobre-los-juegos-de-plataformas-parte-3-madurez/>

[7] *Todo sobre los juegos de plataformas. Parte 4: Tiempos modernos* [en línea]. Retrogamers. [Consulta: 20 de mayo de 2020] <https://retrogamers.es/todo-sobre-los-juegos-de-plataformas-parte-4-tiempos-modernos/>

[8] *System requirements for Unity 2019.1* (2019) [en línea]. Unity. [Consulta: 20 de mayo de 2020] <https://docs.unity3d.com/2019.1/Documentation/Manual/system-requirements.html>

[9] *Video game development* (2020) [en línea]. Wikipedia. [Consulta: 21 de mayo de 2020] <https://en.wikipedia.org/wiki/Video_game_development>

[10] *The History Of Gaming: An Evolving Community* (2015) [en línea]. Techcrunch. [Consulta: 21 de mayo de 2020] <https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/>

[11] *Aprende cómo conectar los controles de la PS3 al PC* (2020) [en línea]. DigitalTrends. [Consulta: 3 de abril de 2020] <https://es.digitaltrends.com/computadoras/conectar-controles-ps3-al-pc/>

[12] Cómo *conectar un control de PS3 a una Mac* [en línea]. WikiHow. [Consulta: 3 de abril de 2020] <https://es.wikihow.com/conectar-un-control-de-PS3-a-una-Mac>

[13] *How to Sync a PS4 Controller on PC or Mac* (2020) [en línea]. WikiHow. [Consulta: 3 de abril de 2020] <https://www.wikihow.com/Sync-a-PS4-Controller-on-PC-or-Mac>

[14] *How to use a PS4 or Xbox One controller on Mac* (2019) [en línea]. MacWorld. [Consulta: 3 de abril de 2020] <https://www.macworld.co.uk/how-to/mac/use-ps4-xbox-controller-mac-3626259/>

[15] *El 'boom' del videojuego: una industria que factura más que el cine y la música juntos* (2020) [en línea]. TheConversation. [Consulta: 29 de mayo de 2020] <https://theconversation.com/el-boom-del-videojuego-una-industria-que-factura-mas-que-el-cine-y-la-musica-juntos-131871>