Diseño y programación de un prototipo interactivo para un Metroidvania 2D: Explorando mecánicas y puzzles

Autor: Gorka Martinez Eguiluz
Tutora: M. Teresa Vidal Peig
Profesor: Joan Arnedo Moreno

Máster U. en Diseño y Programación de Vídeojuegos
Programación Avanzada

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	Diseño y programación de una demo interactiva para un Metroidvania 2D: Explorando mecánicas y puzzles
Nombre del autor:	Gorka Martinez Eguiluz
Nombre del colaborador/a docente :	Jordi Duch Gavaldà
Nombre del PRA:	Joan Arnedo Moreno
Fecha de entrega (mm/aaaa):	01/2024
Titulación o programa:	Máster universitario Online de Diseño y Programación de Videojuegos
Área del Trabajo Final:	TFM - Programación avanzada
Idioma del trabajo:	Español
Palabras clave	Metroidvania, Prototipo, Puzzles.

Resumen del Trabajo (máximo 250 palabras):

Este proyecto tiene como objetivo crear una propuesta completa para un juego hipotético en el género de metroidvania en 2D, tomando como referencia juegos populares como Hollow Knight, The Legend of Zelda y Kirby. El objetivo es desarrollar un juego único que recicle y rediseñe diferentes elementos de estos juegos, al tiempo que se añaden mecánicas originales para evitar que sea una mera copia.

Para planificar esta propuesta, se llevará a cabo el desarrollo y programación de un prototipo funcional que demuestre el potencial del juego. Este prototipo incluirá un nivel tutorial y un nivel principal que mostrarán el flujo del juego, sus mecánicas principales, su estética y ejemplos de posibles puzzles que se encontrarían en la versión final. En resumen, se creará una demostración que abarque gran parte del contenido que se esperaría encontrar en la versión final del juego.

Abstract (in English, 250 words or less):

The objective of this project is to create a comprehensive proposal for a hypothetical 2D Metroidvania game, drawing inspiration from popular games such as Hollow Knight, The Legend of Zelda, and Kirby. The goal is to develop a unique game that reuses and redesigns different elements from these games, while also incorporating original mechanics to avoid being a mere copy.

To plan this proposal, I will develop and program a functional prototype that showcases the game's potential. This prototype will include a tutorial level and a main level that demonstrate the game flow, its core mechanics, aesthetics, and examples of possible puzzles that would be found in the final version. In summary, a demonstration will be created that encompasses a significant portion of the content expected in the final version of the game.

Agradecimientos

Agradecer en primera instancia a mi familia, a mi madre, padre y hermana ya que sin ellos no habría sido capaz de llegar tan lejos en mi vida, y por ende en este máster y trabajo. Gracias a ellos por todos los momentos en los que les he necesitado, por el apoyo, la ayuda, y por haberme acompañado hasta aquí.

Agradecer también a mi pareja todo el apoyo que me ha dado, su ayuda, sus recomendaciones, propuestas, y en general por acompañarme durante todo este proceso.

Agradecer toda la ayuda por parte de todos por todo el tiempo que me han brindado para que pueda realizar este trabajo, ya que han sido unos meses de mucho trabajo duro y mucho tiempo invertido.

Finalmente agradecer a la universidad la oportunidad de cursar este máster relacionado con un mundo tan interesante como es el de los videojuegos, y en especial a mi tutor del trabajo Jordi por apoyarme y ayudarme durante todo el proceso del trabajo.

Abstract

The objective of this project is to create a comprehensive proposal for a hypothetical 2D Metroidvania game, drawing inspiration from popular games such as Hollow Knight, The Legend of Zelda, and Kirby. The goal is to develop a unique game that reuses and redesigns different elements from these games, while also incorporating original mechanics to avoid being a mere copy.

To plan this proposal, I will develop and program a functional prototype that showcases the game's potential. This prototype will include a tutorial level and a main level that demonstrate the game flow, its core mechanics, aesthetics, and examples of possible puzzles that would be found in the final version. In summary, a demonstration will be created that encompasses a significant portion of the content expected in the final version of the game.

Resumen

Este proyecto tiene como objetivo crear una propuesta completa para un juego hipotético en el género de metroidvania en 2D, tomando como referencia juegos populares como Hollow Knight, The Legend of Zelda y Kirby. El objetivo es desarrollar un juego único que recicle y rediseñe diferentes elementos de estos juegos, al tiempo que se añaden mecánicas originales para evitar que sea una mera copia.

Para planificar esta propuesta, se llevará a cabo el desarrollo y programación de un prototipo funcional que demuestre el potencial del juego. Este prototipo incluirá un nivel tutorial y un nivel principal que mostrarán el flujo del juego, sus mecánicas principales, su estética y ejemplos de posibles puzzles que se encontrarían en la versión final. En resumen, se creará una demostración que abarque gran parte del contenido que se esperaría encontrar en la versión final del juego.

Palabras clave
Metroidvania, Prototipo, Puzzles.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción

1.1. Introducción al trabajo

El género Metroidvania combina elementos de exploración no lineal, plataformas y resolución de puzzles en un entorno de mundo abierto. Este tipo de juegos ha ganado popularidad en los últimos años debido a su jugabilidad inmersiva y su capacidad para cautivar a los jugadores con su narrativa y desafíos estratégicos. Es un claro ejemplo de que los juegos en 2 dimensiones tienen aún mucho que mostrar en esta industria.

La motivación detrás de este trabajo se centra en la pasión por los videojuegos, especialmente por el género Metroidvania y el estilo de dos dimensiones. La oportunidad de diseñar y programar una demostración permitirá sumergirse en un proceso creativo, explorar nuevas mecánicas y acertijos, y ofrecer una experiencia emocionante a los posibles jugadores que estén dispuestos a darle una oportunidad a un título como este.

1.2. Descripción

Este trabajo tiene como objetivo diseñar y programar una demostración interactiva de un juego completo del género Metroidvania 2D. El propósito principal es mostrar las principales mecánicas y puzzles que se encontrarían dentro del hipotético juego planteado. Se hará enfoque en la planificación y diseño de niveles, la implementación de mecánicas de juego, la creación de puzzles desafiantes y la optimización del rendimiento de las propias mecánicas para demostrar el potencial detrás de estas. A través de una combinación de habilidades de diseño visual y programación, se busca crear un prototipo que demuestre la calidad y el potencial que tendría juego completo.

En este caso, el trabajo parte desde 0. El trabajo empezó tras pensar una propuesta realista. A partir de ahí, se fueron añadiendo ideas que encajasen con la propuesta, valorando diferentes vías hasta llegar a la propuesta exhibida dentro de este documento. Es decir, el trabajo no solo consiste en explotar una idea, sino que también hay que tener en cuenta la creación y el nacimiento de esa idea, la cual ha sido parte larga del proceso del trabajo.

1.3. Objetivos generales

El objetivo general de este trabajo es la creación de una demo/prototipo que represente el potencial del hipotético juego, así como sus mecánicas principales y su uso dentro del mismo, estéticas, flujo del juego, etc.

1.3.1. Objetivos principales

- Programar unas mecánicas acorde a su descripción, y hacer un buen uso de estas.

- Crear puzzles que tengan relación con las mecánicas de manera que se cree una armonía entre ambas cosas, dando vida al juego.

- Realizar un diseño de niveles que haga posible la creación de los puzzles mencionados anteriormente.

- Homogeneizar todos los elementos correctamente para crear una demo/prototipo completamente funcional y acorde a su descripción. Esto implica una buena y limpia programación, relación entre todos los elementos, diseño, buena organización…

- Completar la memoria detallando todos los procesos llevados a cabo durante el trabajo, explicandolos bien y de manera correcta y ordenada.

1.3.2. Objetivos secundarios

- Diseñar la demo y sus componentes de manera armónica, creando un ambiente completamente sincronizado a nivel de diseño y estéticas (se va a priorizar la parte de la programación de mecánicas antes que esto, ya que es un proyecto enfocado a la programación más que al diseño)

1.4. Metodología y proceso de trabajo

En este proyecto, se ha utilizado una estrategia de desarrollo de un producto nuevo para crear una demostración de un juego Metroidvania 2D. Esta estrategia es la más apropiada porque ha permitido tener la libertad de diseñar y desarrollar un juego desde cero, adaptándolo a ideas y visión personales.

En cuanto al proceso de trabajo realizado, éste sigue una metodología ágil llamada Scrum. Esta metodología permite gestionar el proyecto de manera iterativa e incremental, lo que hace que adaptarse a los cambios y mejorar continuamente el producto no sea un problema. Por ello, para abordar el proyecto, se utilizan diferentes recursos y metodologías:

- **Investigación**: Realización de una investigación exhaustiva sobre el género Metroidvania 2D y los juegos similares existentes para comprender las mecánicas clave y las mejores prácticas (por ejemplo Hollow Knight[1]).

- **Diseño de niveles**: Utilización herramientas de diseño de niveles para crear y planificar la estructura de la Demo del juego. Utilizando herramientas como Unity y su integrado Tilemap por ejemplo.

- **Recursos**: Uso de bibliotecas de arte, sonido, sprites, imágenes y animaciones para enriquecer la experiencia visual y auditiva del juego.

- **Programación**: Implementación de las mecánicas de juego, los sistemas de combate y los puzzles utilizando el motor de desarrollo Unity. Utilización del lenguaje

de programación C# para crear la lógica del juego y asegurar que todas las interacciones sean fluidas y coherentes.

- **Pruebas y retroalimentación**: Realización de pruebas continuas para recopilar información sobre el desarrollo de la Demo y mejorar la experiencia de la misma.

En resumen, se ha aplicado una estrategia de desarrollo de un producto nuevo utilizando la metodología ágil Scrum. Al mismo tiempo, se ha utilizado diferentes metodologías de investigación y desarrollo, así como recursos adicionales, para abordar el proyecto y alcanzar los objetivos propuestos en la anterior sección. Con esta metodología, se busca cumplir con la propuesta al mismo tiempo que se permite la evolución del trabajo a medida de su desarrollo y de las necesidades en cada momento.

1.5. Planificación

En cuanto a la planificación del trabajo, estos serían los hitos planificados para el mismo:

- **(1º Hito) Investigación inicial y planteamiento de la idea:** Realización de una investigación exhaustiva sobre el género Metroidvania 2D y los juegos similares existentes. En base a eso, pensar en la idea inicial para el proyecto final. *Fecha clave: del 1 al 30 de Septiembre.*

- **(2º Hito) Diseño de niveles y mecánicas:** Creación del diseño de los niveles y las mecánicas principales del juego, así como su lógica e historia. *Fecha clave: del 1 al 31 de Octubre.*

- **(3º Hito) Desarrollo de prototipo:** Implementación de un prototipo básico de la Demo que incluye las mecánicas y los puzzles diseñados. *Fecha clave: del 1 de Noviembre al 17 de Diciembre.*

- **(4º Hito) Integración de arte y sonido:** Incorporación de los elementos de arte, animaciones, sprites, imágenes y sonido a la Demo para mejorar la experiencia visual y auditiva. *Fecha clave: del 1 al 31 de Diciembre.*

- **(5º Hito) Pruebas, mejoras y optimización:** Realización de pruebas sobre el prototipo e implementar mejoras en él, optimización del rendimiento y ajuste sobre la jugabilidad. *Fecha clave: del 18 al 31 de Diciembre.*

- **(6º Hito) Pruebas finales y pulido:** Realización de pruebas exhaustivas de la Demo completa, solución de errores y realización de ajustes finales para garantizar una experiencia de juego de alta calidad. *Fecha clave: del 1 al 14 de Enero.*

- **(7º Hito) Documentación y escritura de la Memoria:** Desarrollo y escritura de la memoria explicando todos los puntos hechos, diseños, desarrollo, pruebas, etc. *Fecha clave: del 1 de octubre al 14 de Enero.*

Aquí podemos ver la representación de estos Hitos así como las fechas y duración de las diferentes entregas (PECs) de este Trabajo Final:

	Septiembre				Octubre				Noviembre				Diciembre				Enero			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1º Hito																				
2º Hito																				
3º Hito																				
4º Hito																				
5º Hito																				
6º Hito																				
7º Hito																				
PEC1																				
PEC2																				
PEC3																				
PEC4																				
PEC5																				

Tabla 1: Planificación de hitos y PECs mediante un diagrama de Gantt

1.6. Presupuesto

En este caso, el trabajo final propuesto tiene un coste de **0€** ya que las herramientas utilizadas así como la mano de obra del estudiante en este caso no van a tener ningún coste. Aun así, el proyecto propone la creación de un prototipo sobre un hipotético juego. La creación de este hipotético juego si conlleva sus gastos. De modo que estos serian los puntos a tener en cuenta para la creación del juego al completo:

- **Recursos humanos:** Si el juego final es desarrollado solamente por el estudiante, no hay costos de contratación de personal adicional. Sin embargo, se debe considerar el valor del tiempo invertido por parte del estudiante en el desarrollo del juego. En caso de contratación de personal adicional, habría que tener en cuenta su tiempo invertido también.

- **Software y herramientas:** Teniendo acceso a un motor de desarrollo como Unity, no hay costos adicionales. Aunque según las condiciones de Unity, si el juego final supera cierto rango en ganancias, habría que comprar una licencia del mismo.

- **Arte y sonido:** Al no contar con habilidades en diseño gráfico o creación de sonido ya que la rama cursada es la de programación, hay que considerar la contratación de artistas o músicos independientes. Estos costos variarán según los acuerdos y las tarifas de cada profesional. Otra opción es la compra de Assets mas profesionales que encajen con el juego.

- **Marketing digital:** Hay que destinar un presupuesto para estrategias de marketing digital, como campañas en redes sociales, publicidad en línea y promoción en blogs o sitios especializados. Estos costos dependen de la duración y alcance de las campañas publicitarias que se decidan implementar.

- **Participación en eventos:** Asistir a convenciones o ferias de videojuegos para promocionar el juego también tiene que ser considerado, por lo que habría costos de transporte, alojamiento, material promocional y tarifas de exhibición.

- **Plataformas de distribución:** Al tener que publicar el juego en plataformas como Steam, PlayStation Store o Xbox Live, hay que tener en cuenta las tarifas de publicación y las comisiones que se aplican a las ventas del juego.

- **Localización y traducción:** Si se busca que el juego esté disponible en diferentes idiomas, hay que considerar los costos de localización y traducción del contenido.

Todos estos puntos son hipotéticos y pueden variar considerablemente según las necesidades, objetivos y acuerdos específicos. Tras investigar y obtener cotizaciones reales de profesionales o servicios potencialmente necesarios, el presupuesto para crear y lanzar un juego de estas dimensiones oscila desde los **5.000€** hasta los **50.000€.** La diferencia entre las dos cifras es bastante elevada dado que las decisiones tomadas durante el proceso del proyecto podrían variar mucho. Si se busca hacer un proyecto muy profesional buscando mayor alcance, sin escatimar en gastos, contratando toda la gente/servicios necesarios, etc. la cifra podría ascender hasta los **100.000€ - 200.000€.** El mayor ejemplo podrían ser los juegos Triple A, desarrollados por multinacionales con presupuestos millonarios. En cambio, este juego es un proyecto pequeño, no muy ambicioso que se desarrollaría de manera individual o con un equipo de pequeñas dimensiones, sin tener demasiado gasto en recursos humanos. Además, de cara al diseño no se buscaría nada extravagante que disparase el presupuesto, así como no haría falta ninguna licencia de ningún motor gráfico. Habría que considerar los costos de marketing, eventos y traducción, por lo que el juego realmente estaría en el primer ejemplo de rango de precios.

Otro punto a valorar sería buscar financiación del juego. Esto puede ayudar a cubrir los costes del mismo, pudiendo invertir más dinero en el juego para obtener más beneficio. Aunque hay que tener en cuenta que si aplicamos una estrategia como esa, los beneficios del juego al final serán menores ya que normalmente la parte financiadora del proyecto se debe llevar una comisión de estos beneficios. Finalmente, el gasto en la creación y publicación del juego sería directamente proporcional al alcance y beneficios de este (normalmente es así, a no ser que surgiera una excepción, tanto para bien como para mal).

1.7. Estructura del resto del documento

En los siguientes apartados de esta memoria, encontramos apartados interesantes como puede ser el análisis del mercado y estado del arte, en el cual se explaya toda la información relevante sobre el mercado de los juegos 2D, más concretamente los del género Metroidvania. Aquí se enseñan competencias del juego, fortalezas y debilidades.

De seguido, se presentará la propuesta del juego, explicando su origen e idea, todas sus mecánicas explicadas al detalle y el funcionamiento de las mismas, la historia y las estéticas. Tras esto, en el apartado de Diseño muestra la manera en la que se procederá a la implementación de todos estos datos dentro del prototipo. Esta parte es crucial ya que describe al milímetro cómo va a ser el producto final de este proyecto.

Tras eso, se explicará la implementación de dicho prototipo, explicando sensaciones y mostrando gráficamente el resultado obtenido. Aquí se explicará metódicamente todos los procesos que se han superado para crear el producto objetivo del trabajo. También se harán demostraciones para comprobar que todo funciona correctamente y se sacarán datos relevantes para la última parte del documento.

Finalmente, se recogerán los datos del apartado comentado anteriormente para sacar conclusiones del trabajo realizado, analizar objetivos obtenidos y valorar el resultado sobre la idea inicial del trabajo. Del mismo modo, se presentarán posibles mejoras para un futuro, así como información sobre la creación del juego al completo en base al prototipo creado. Tras esto, el documento termina con una conclusión que hace un análisis y valoración personal de manera resumida sobre el trabajo al completo.

2. Análisis de mercado y estado del arte

2.1. Público objetivo y perfiles de usuario

Para comenzar con este apartado, se ha realizado una búsqueda exhaustiva sobre el público objetivo del producto a crear en este proyecto. Ya que el producto es un juego, lo primero ha sido recurrir a plataformas de distribución de juegos como Steam[3] para adquirir estadísticas sobre jugadores. En este caso, el público objetivo del juego son los aficionados a los videojuegos de plataformas y aventuras, especialmente los fanáticos del género del Metroidvania. La información que podemos obtener de Steam es bastante limitada, tan solo podemos sacar estadísticas que muestran que el público objetivo del juego no es muy elevado, ya que la cantidad de gente que juega a juegos del género metroidvania es bastante bajo comparado con otros géneros.

Entonces, hay que hacer una búsqueda más exhaustiva para poder conocer el perfil de estos jugadores. Para ello, se han buscado encuestas y comparado la fiabilidad de cada una para obtener resultados lo más fiables posibles. La encuesta que más se ajusta al caso proviene de la página de Reddit[4], del usuario Sirradez[5]. Podemos ver en ella que el nicho de mercado es muy diverso. Aquí están los datos más relevantes de la encuesta:

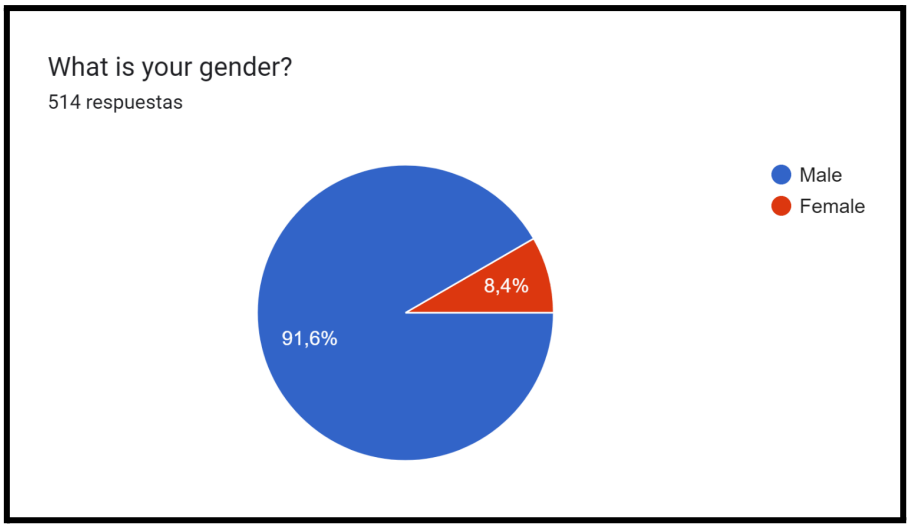

Figura 1: Encuesta - genero de los jugadores

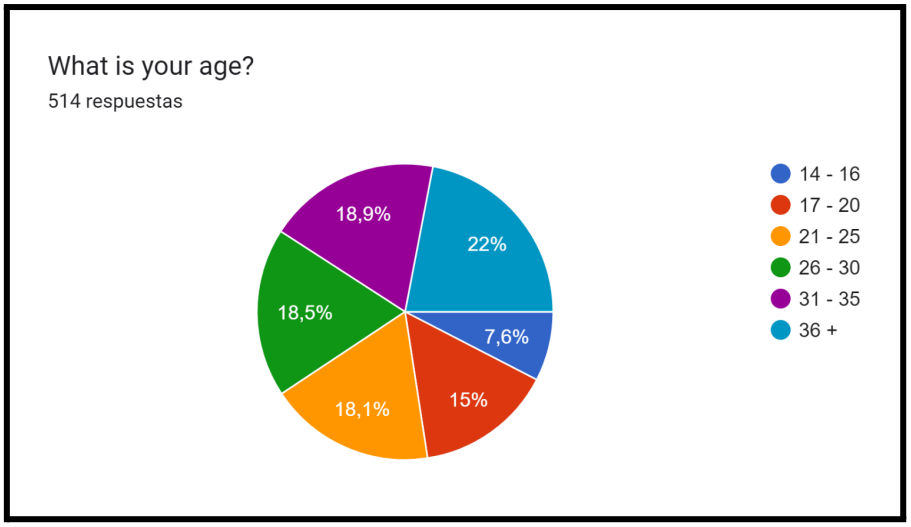

Figura 2: Encuesta - edad de los jugadores

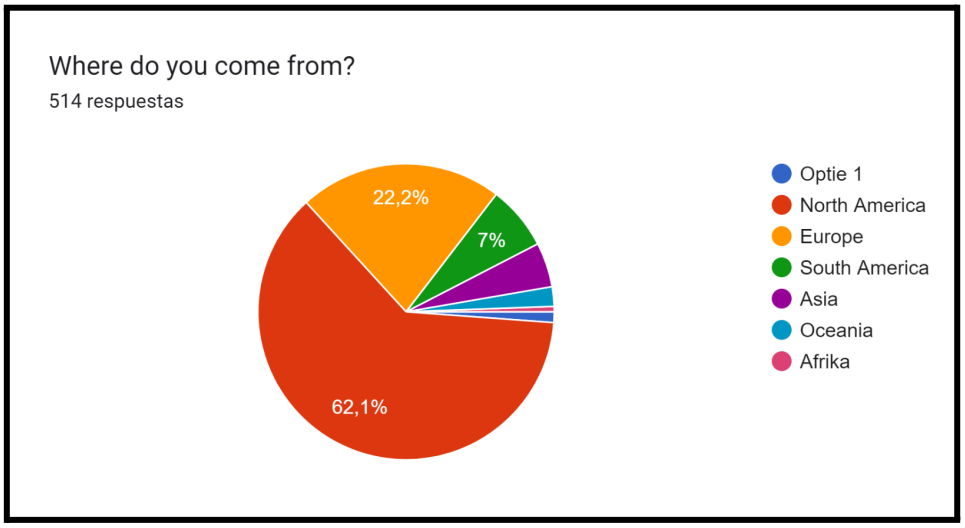

Figura 3: Encuesta - proveniencia de los jugadores

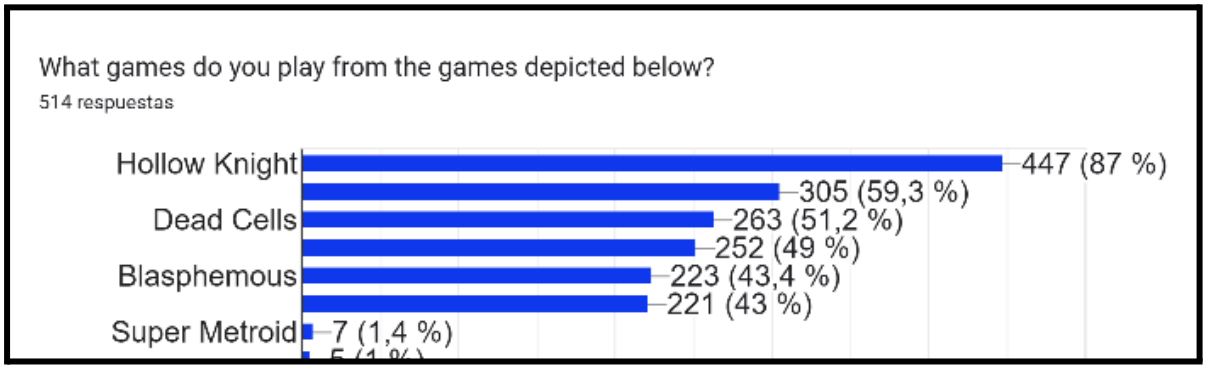

Figura 4: Encuesta - juegos más jugados de los jugadores

Gracias a esta encuesta, podemos crear un perfil bastante preciso sobre el potencial usuario del juego a crear: el género predominante a la hora de jugar a este tipo de juegos son los hombres. Las personas que los juegan van desde jóvenes adolescentes hasta adultos. Cabe destacar que más del 50% de los jugadores son mayores de 26 años, y que la edad promedio de los jugadores está sobre los 30 años. Esto es porque la mayoría de ellos y ellas tienen un gran interés en los juegos desafiantes con una historia bien elaborada (objetivo del juego de este proyecto). Si nos fijamos en el lugar de residencia de los jugadores, vemos que la mayoría reside en Norteamérica y Europa, siendo el idioma inglés predominante en ambas regiones. El segundo idioma más utilizado por los usuarios sería el español, por parte de Europa y Sudamérica. Esto es un punto importante a tener en cuenta a la hora de crear el juego.

Con estos datos, tenemos identificado el perfil de los usuarios y por ello el público objetivo al que debe ir dirigido el juego dado que este va a pertenecer a este género. Hay que ir creando el juego teniendo en cuenta estos datos, ya que de esa manera podemos crear un producto más atractivo para los usuarios y conseguir en este caso una venta más exitosa del juego. Finalmente, en la encuesta podemos observar que juegos son los más jugados dentro del género, información que se analiza en el siguiente apartado.

2.2. Competencia

A la hora de analizar la competencia, se ha decidido seleccionar los 3 juegos del género más jugados según la encuesta del anterior apartado. El primero de ellos es Hollow Knight[1], uno de los juegos en el que se va a inspirar este proyecto. Esto se decidió previamente ya que se considera el juego más exitoso dentro del género Metroidvania moderno, por lo que es un gran ejemplo para desarrollar el juego, y al mismo tiempo un gran ejemplo para hacer análisis de competencia. El segundo juego más jugado según la encuesta contiene dos entregas: Ori and the Blind Forest[6], y Ori and the Will of the Wisps[7]. Dado que estos juegos son de la misma desarrolladora y uno es la continuación del otro, solo se va a utilizar el más moderno de los dos para la comparación: Ori and the Will of the Wisps[7]. El tercer juego para la comparación va a ser Dead Cells[8].

Para realizar la comparación entre ellos, se van a seleccionar los puntos más importantes a tener en cuenta de los juegos. Estos puntos también están definidos en la encuesta, donde se pregunta directamente a los usuarios que es lo más importante para ellos en estos juegos. En este caso, se van a seleccionar los 5 puntos que más se repiten por los jugadores y que a su vez son los más relevantes para un juego del género metroidvania: Exploración, Puzzles, Combate, Dificultad y la Atmósfera (ambientación, música, diseño del mundo…). Teniendo esto preparado, toca crear la tabla con la puntuación de cada uno de estos puntos en los juegos. Para eso, se van a utilizar puntuaciones y estadísticas recogidas de la página Metacritic[9], una de las páginas más famosas y respetada en la valoración de juegos la cual recoge puntuaciones de todo tipo, tanto al nivel de usuario como de otras empresas.

Con todo esto preparado, estas son las tablas que nos quedan para posteriormente realizar la investigación sobre la competencia:

	Hollow Knight[1]	**Ori and the Will of the Wisps[7]**	**Dead Cells[8]**
Exploración	Puntuación: **9.5**	Puntuación: **8.5**	Puntuación: **7.5**
Puzzles	Puntuación: **8**	Puntuación: **8**	Puntuación: **7**
Combate	Puntuación: **9**	Puntuación: **8**	Puntuación: **9**
Dificultad	Puntuación: **9**	Puntuación: **8**	Puntuación: **9**
Atmósfera	Puntuación: **10**	Puntuación: **9**	Puntuación: **8**
Puntuación Media	Puntuación: **9.1**	Puntuación: **8.3**	Puntuación: **8.1**

Tabla 2: Puntuación de la competencia sacada de Metacritic

Habiendo recogido estos datos, toca hacer un análisis sobre ellos y así identificar los puntos fuertes de la competencia para sacar conclusiones acerca del mercado. Aunque existen otros puntos también importantes a la hora de crear un juego 2D del género metroidvania, estos 5 son los más importantes a la hora de valorar un juego de este estilo. Podemos observar que las puntuaciones de la competencia en estos ámbitos es muy alta, esto refleja que los 3 juegos están muy bien creados a vista de las empresas y consumidores. Esto explica porqué los 3 juegos lideran el mercado de los metroidvania, al mismo tiempo que nos da mucha información para poder realizar un buen trabajo.

Podemos ver que hay muchas exigencias para poder crear un juego con posibilidades ante su competencia en este ámbito. Esto a su vez es lógico, y es lo que empuja a que un juego sea exitoso o no. Por ello, a la hora de hacer este proyecto habría que tener las siguientes cosas en mente:

- El juego debe ofrecer una atmósfera original, con armonía entre la historia, la ambientación, las estéticas y los sonidos. Si tomamos como ejemplo los juegos mencionados anteriormente, cada uno tiene su mundo creado, con una historia interesante que contar, con estéticas a su medida y armonía entre todo el diseño del juego.

- A su vez, el juego debe llevar algo original desde el punto de vista de las mecánicas, así añadiendo un modo de combate interesante. Lo difícil de este punto es que la mecánica debe ser algo viable dentro de las posibilidades de la atmósfera creada. Y no solo eso, sino que la mecánica debe ser algo llamativo, que no sea repetitiva y que sea sencilla de entender, sino podría volverse algo tediosa a la hora de utilizarla.

- A parte de crear un modo de combate original, hay que realizar un balance de este correcto para que el juego suponga un reto, añadiendo una dificultad óptima. Por lo que si se crea una mecánica que te de demasiadas ventajas, no se va a poder

realizar un buen balance, y lo mismo si la mecánica no te da opciones creativas y es todo el rato igual.

- Finalmente, el juego debe tener un diseño de niveles interesante. Esto implica crear un mundo donde la exploración sea algo crucial y motive a realizarla. Al mismo, tiempo debe haber unos retos en forma de puzzle que enganche a los jugadores. Este punto es de los más complicados, ya que en los Metroidvania hay que trabajarlo muy bien para no realizar mundos repetitivos, ya que por el tipo de juego y al ser en 2D no es nada raro hacerlo repetitivo. Por ello, la colocación de diferentes biomas, puzzles, secretos ayudan a que el mundo no sea tan monótono.

Parece que hay que crear el juego “perfecto” para tener alguna posibilidad con la competencia, pero esto tiene algo bueno también: Los juegos de la competencia ya se pueden considerar “perfectos” para el mercado, por lo que utilizarlos de referencia es una muy buena estrategia para crear el juego de este proyecto.

Finalmente, hay un último juego que mencionar y es el Rogue Spirit[10] de Kids With Sticks. Aunque este juego no pertenezca al género Metroidvania (se trata de un Roguelike/RPG), tiene la misma mecánica principal que el juego a desarrollar en este proyecto (explicada más adelante). Esto es una casualidad ya que el juego se desconocía antes de plantear la propuesta y diseño de este proyecto (no es un juego demasiado conocido, apenas tiene 100 reseñas frente a las casi 300.000 de Hollow Knight[1]). Al final es muy difícil crear una idea original en el sector de los videojuegos, ya que hay tanta cantidad de estos que es muy fácil que la idea ya esté en uso o por lo menos algo parecido. De hecho, la idea de la mecánica principal del juego a crear en este proyecto es un concepto reciclado sacado de un personaje de League Of Legends[12] (más tarde se explica también) y de la saga Kirby[11], por lo que no es creada desde 0. La conclusión de esto es que por un lado no es malo reutilizar ideas que ya existan en el sector, siempre y cuando se añada algo de valor, algo diferente y original que llame la atención y por el otro lado crear ideas originales esta muy bien, pero es muy complicado porque siempre habrá algún juego con un concepto parecido (ya que se trabaja con las mismas tecnologías).

2.3. Debilidades y fortalezas del proyecto

Tras haber identificado a el público objetivo y haber analizado a la competencia, hay que analizar las debilidades y fortalezas del proyecto. Comenzando con las debilidades, estas son las tres más destacables:

- Realizar un juego de estas dimensiones con oportunidad en el mercado es muy difícil, y menos con el tiempo que tenemos para realizar este proyecto. Es por ello que solo se va a realizar un prototipo para demostrar las posibilidades del juego planteado en este proyecto. Hoy en día desarrollar un juego exitoso tiene muy baja probabilidad (porque hay mucha cantidad y mucha competencia), por lo que hay que invertir mucho más tiempo para dejar lo menos posible a la suerte en lo que al éxito del juego se refiere.

- Otra debilidad es que el juego solo va a ser realizado por el estudiante, y este no contiene los conocimientos suficientes para realizar todas las partes de una manera perfecta. Esto es algo lógico ya que por eso existen las especializaciones, y por eso empresas de videojuegos tienen expertos para todos los sectores: dirección, arte, diseño, programación, etc.

- Como última debilidad destacaría el presupuesto del juego, que en este caso es de 0€. Esto fusionado con lo anterior hace que el juego tenga menos posibilidades de un buen desarrollo (a nivel comercial), porque no se puede acceder a herramientas de pago de terceros. Hay casos en los que un juego puede estar realizado por una sola persona, ya que si tiene algo de presupuesto lo puede destinar para cubrir los sectores en los que no está especializado utilizando servicios de terceros (diseñadores, marketing).

Por suerte, no todo son debilidades. El proyecto tiene varias fortalezas muy destacables, estas serían las dos más importantes:

- En contraparte con la primera debilidad, puede que crear un juego al completo sea una idea impensable, pero por ello existe la posibilidad de crear un prototipo. Esto es mucho más viable en el tiempo estimado, además de ser una excelente estrategia. Esto es porque hoy en día un prototipo te puede dar muchísima ventaja a la hora de hacer un videojuego. Al crear un prototipo, generas una demostración de lo que puede ser un juego con las ideas presentadas en él. Este prototipo puede ser testeado por diferentes usuarios, y de esa manera recibir feedback y tener oportunidad de evitar fallos antes de crear el juego al completo. Al mismo tiempo, este prototipo servirá para ver el interés que pueda haber por un juego final, y así cambiar a tiempo lo que propone el juego para que sea más atractivo. Sin duda alguna, sacar un prototipo de un juego antes de desarrollar uno al 100% es un acierto, por lo menos siendo un desarrollador no conocido (ya que si como desarrollador ya te has ganado confianza en el sector con un juego exitoso, puedes realizar un segundo juego completo sin prototipo ya que tienes una confianza ya creada y sabes que lo van a jugar).

- La otra fortaleza que existe hoy en día son las oportunidades gratuitas que existen. Puede que tener 0€ de presupuesto no sea lo más óptimo, pero no por ello no existen alternativas para impulsar el juego. Para empezar, existen las Redes Sociales, una herramienta muy potente para hacer marketing gratuito, generando expectación para el juego. Otra herramienta muy conocida son las incubadoras. Estas tratan de apoyar tu juego de manera gratuita ayudándote en su desarrollo, tanto con expertos, como con ayudas economicas. A cambio, se llevan un porcentaje de la venta del videojuego una vez esté completado. Otra opción es el uso de Assets gratuitos, al no tener demasiado conocimiento en el arte y el diseño es muy buena opción para presentar un prototipo. Un juego no solo trata de hacerlo y lanzarlo, sino que hay que saber explotar bien todos los recursos que tengamos a nuestro alcance.

Puede que realizar un proyecto de este estilo tenga sus dificultades, pero también tiene oportunidades. Precisamente, se ha tenido en cuenta la primera fortaleza ya que el objetivo del proyecto es crear un prototipo, algo viable y realista para el tamaño y tiempo del proyecto. Si se aprovechan bien las oportunidades y se tienen en cuenta las limitaciones creando un equilibrio entre ambas, la creación de un buen juego es viable.

2.4. Amenazas y oportunidades del proyecto

Para finalizar con este apartado, se van a comentar las posibles amenazas y oportunidades del proyecto de una manera dinámica y resumida. Comenzando con las amenazas, estas serían las más peligrosas y destacables:

- Complicaciones a la hora de desarrollar el proyecto. Errores y dificultades a nivel de desarrollo que amenacen el crear un prototipo a la altura de las expectativas en el tiempo establecido, o falta de tiempo por circunstancias inesperadas o mala planificación.

- Dado al éxito actual de los Metroidvania, hay mucha competencia a la hora de mostrar el juego o el prototipo y con ello conseguir repercusión, amenazando su éxito.

- Robo de la idea. Puede que sea muy improbable, pero existe la amenaza de que la idea de este juego sea robada y desarrollada por un equipo más completo en menor tiempo, cambiando lo justo para evitar problemas legales y adjudicándose así la idea. Desgraciadamente, esto es algo muy habitual en la creación de contenidos varios como juegos, música, videos, etc.

Como se puede ver, son amenazas activas y con posibilidad de ocurrir. Pero al final, todo proyecto o producto tiene sus riesgos y amenazas, por lo que es algo habitual tener que tenerlas en cuenta. Eso sí, el proyecto también tiene sus oportunidades. Estas son las más interesantes:

- Un proyecto de estas dimensiones no solo es lo ideal para cerrar un Máster, sino que también puede brindar oportunidades laborales. Al final, el juego va a servir para demostrar de que es capaz el estudiante dentro del tiempo ofrecido, lo cual será una carta de presentación para posibles oportunidades laborales con otras empresas, ya que es contenido del proyecto será parte del portafolio y reflejará directamente las habilidades empleadas.

- Otra posibilidad es el éxito del juego, tras presentarlo como trabajo existe la oportunidad de presentarlo públicamente al mercado y comprobar sensaciones de los usuarios. Esto puede tener una repercusión positiva, planteando el desarrollo completo del videojuego que es su objetivo real (no tiene que ser inmediato, puede ocurrir más años adelante).

- La última oportunidad destacable sería la oportunidad de aprender más. A la hora de desarrollar un juego de manera más profesional (como puede ser este prototipo), es donde más cosas se aprenden. Y esto al final sirve de experiencia para mejorar y cada vez desarrollar prototipos mejores, haciendo así una escalera hacia el éxito. Por mucha competencia que haya en el sector, si trabajas duro es posible ganar un hueco en la industria, y todo comienza por proyectos de este estilo.

Si tenemos en cuenta todas las amenazas y oportunidades, estas están bastante balanceadas. Como se ha comentado anteriormente, a la hora de crear un producto existe el riesgo de fallar, pero también el riesgo de salir con éxito. Además, esto no suele estar ligado tanto a la suerte, sino que está más ligado al trabajo y habilidad a la hora de crear y presentar el producto. Al fin y al cabo, eso es un punto de motivación a la hora de desarrollar un videojuego, y es lo que finalmente marca la diferencia entre un buen videojuego y uno que no esté tan completo.

3. Propuesta

3.1. Descripción general del juego

La idea del juego nace después de jugar a Hollow Knight y ver directamente el potencial del género de los Metroidvania 2D. Diferentes puzzles, una aventura que contenga acción, plataformas, jefes, retos, etc. Como generalmente los metroidvanias funcionan alrededor de una mecánica principal, tocaba pensar esta. En ese momento se pensó en la habilidad de absorber a los enemigos como hace Kirby[11] en sus juegos, o absorber las habilidades de los enemigos como hace Viego de League Of Legends[12]. La idea de utilizar a tus enemigos como herramienta llamó mucho la atención, y se pensó una idea alrededor de esto.

De aquí nace la idea de que el jugador principal sea un alma vagante que pueda poseer a sus enemigos, entrando en los cuerpos y controlando estos desde dentro. Esto es muy parecido a la habilidad de Viego comentada anteriormente. En un mundo con criaturas corrientes como las de La Tierra no tendría demasiada gracia, ya que los animales que habitan el planeta no son lo suficientemente interesantes como para brindar habilidades sorprendentes a nuestro personaje. Por ello, la idea de crear un universo que contenga dragones escupe fuego, criaturas mágicas y criaturas con todo tipo de habilidades es la idea correcta para desarrollar este concepto.

De este modo, el juego cambiaría dependiendo de a qué criatura estemos controlando, abriendo un amplio abanico a la creación de puzzles, mundo, historia, mecánicas, estrategias y en general dando oportunidad al desarrollo de un gran juego, si se utilizan bien los recursos. Teniendo la idea base creada, queda matizar los detalles que completan esta idea para luego hacer posible su desarrollo.

3.2. Introducción al juego

Lo primero que había que hacer era crear una pequeña historia sobre la que rondaría todo el juego (basta con crear una historia base, la cual se podría completar y desarrollar al completo más tarde, dependiendo del desarrollo). Esta, debe introducir al jugador al juego de manera lógica. La base de esta historia sería la siguiente:

Nos situamos en un mundo repartido en diferentes reinos, con una fauna muy hostil. Es por ello que los terrenos de los reinos están muy bien establecidos y protegidos, delimitando cada zona. De uno de estos reinos aparece el protagonista/personaje principal: una especie de caballero normal y corriente, con su espada, escudo, etc.. El jugador controlará a este personaje principal. Su misión al principio será “sencilla”, enfrentarse a un demonio que está acechando el reino donde reside nuestro caballero. Inevitablemente, el caballero va a ser derrotado por este demonio (este será un evento canónico obligatorio, por lo que el jugador no tendrá ninguna oportunidad de derrotarlo).

Esta sección de la historia del juego va a ser introductoria, mezclando cinemáticas junto a las primeras interacciones del jugador para crear un vínculo entre el juego y el jugador. Aquí, el jugador aprenderá los primeros controles del juego. Tras esta introducción, el jugador será derrotado en la batalla contra el demonio como se ha explicado arriba. Aun así, el castigo de este demonio no va a ser matarlo, sino que lo va a condenar a estar eternamente entre la vida y la muerte mediante una maldición. Esto lo va a hacer sacando el alma del caballero de su cuerpo físico, dejando al personaje en una especie de transformación fantasmal. Luego, el demonio cogerá la parte física del cuerpo del jugador para guardarla como trofeo, e invadirá su reino ya que el jugador no ha sido capaz de derrotarlo. El personaje será expulsado de ahí por el demonio a un lugar muy lejano. Tras éstos sucesos, ya estaría planteado el objetivo principal del jugador que será buscar al demonio para recuperar su cuerpo, y esta vez si, derrotarlo para liberar el reino y volver la paz.

Este planteamiento sirve de introducción lógica al juego, pero no se va a incluir en el prototipo directamente, ya que solamente es parte del *Lore* del juego y no tiene relevancia para las competencias del proyecto (ya que solo sería contar la historia que se plantea en este documento de manera gráfica). Simplemente es para dar introducción a la historia de modo que a la hora de desarrollar el juego al completo no haya lagunas. Lo que si se va a introducir es un pequeño nivel Tutorial (Sin nada de Lore, directamente Gameplay) que va a enseñar al jugador como controlar a su personaje y cuales son las mecánicas de este.

Finalmente, con la nueva forma fantasmal de nuestro personaje principal, el gameplay va a cambiar por completo a lo que parecía en la hipotética introducción. Aquí es donde realmente empieza el juego y donde el jugador va a tener que aprender las mecánicas del juego. Como se ha comentado anteriormente, se va a plantear un tutorial que ayude al jugador con sus primeros pasos. Tras eso, conseguimos crear una base donde poder plasmar la idea principal del juego: Un alma/fantasma que puede vagar por el mundo con la posibilidad de poseer los cuerpos de las diferentes criaturas que lo componen. A continuación se explica esta mecánica y su funcionamiento.

3.3. Mecánicas del juego

3.3.1. Primera Mecánica Principal

Es hora de explicar al detalle las mecánicas. Como se ha explicado anteriormente, el jugador es un especie de fantasma, alma, espíritu que puede poseer cuerpos vacíos y controlarlos, ¿pero cómo funciona esto realmente?. Siguiendo una lógica general, un alma/fantasma no puede morir, pero tampoco puede “vivir”. Es por ello que el jugador al ser un alma no puede interactuar con el mundo físico de manera directa, aunque hay una excepción: los cuerpos muertos o cadáveres de las diferentes criaturas del mundo (cuerpos físicos sin un alma). La base de todo esto es que cuando una persona/animal/ser vivo muere, se dice que el alma de este viaja a otro mundo, dejando el cuerpo físico vacío (formando así un cadáver). En cambio, nuestro caballero al estar maldito se ha quedado en un estado intermedio donde ha abandonado su cuerpo físico, pero su alma no ha viajado a

otro mundo, sino que se ha quedado en el mismo mundo. Es por eso que nuestro personaje principal tiene la habilidad de entrar a cadáveres o cuerpos vacíos de diferentes criaturas y "poseerlos" dándoles el alma que le falta a ese cuerpo. De esa manera, obtendrá el control del cuerpo de las diferentes criaturas y así podrá interactuar con el mundo físico.

Esta es la mecánica principal en torno a la que se mueve el juego. Esta mecánica principal del juego nos permite ir avanzando por el mundo, entrando en diferentes cadáveres/cuerpos y utilizando las habilidades y posibilidades de estos según nos convenga. Por ejemplo, si se entra en el cuerpo de un dragón de fuego, se va a poder lanzar fuego y atacar con eso, quemar cosas, etc… o si se entra en el cuerpo de un caballero, este tendrá la posibilidad de utilizar sus brazos para blandir una espada para atacar o cortar cosas. Esta mecánica principal está inspirada en el poder principal del famoso personaje de videojuegos Kirby[11], que es capaz de engullir a sus enemigos y con ello robar sus habilidades y en Viego de League Of Legends[12], que hace algo muy parecido.

3.3.2. Segunda Mecánica Principal

La cosa no se queda ahí, ya que esta mecánica principal nos permite crear una contraparte de la misma ofreciendo originalidad al juego y algo diferente (teniendo en cuenta las similitudes de los ejemplos mencionados arriba). Esta contraparte trata de que el juego se va a dividir en dos estados: el **Estado de Alma** y el **Estado de Cuerpo**. El **Estado de Alma** va a ser cuando el jugador está en modo fantasma/alma, y el **Estado de Cuerpo** va a ser cuando el jugador está dentro de un cuerpo físico/cadáver. Cuando el jugador está en el **Estado de Alma**, este no puede ser atacado ni puede atacar a nadie, ya que no puede interactuar con el mundo físico exceptuando los cadáveres que pueda poseer como se ha explicado arriba. Tan solo puede moverse en el escenario libremente levitando y atravesando las cosas físicas. El jugador puede atravesar todo tipo de paredes y suelos, pero hay que ponerle límites para así respetar los límites de cada escenario. Aparte de esto, hay que evitar que el jugador simplemente atraviese todo y se salte los puzzles. Por ello, habrá un campo mágico de fuerza que no podrá atravesar el Alma, pero que sí se podrá atravesar si estamos dentro de un cuerpo, es decir, un muro mágico que solo actúa cuando estamos en el **Estado de Alma**. Aun así, en este **Estado de Alma** se desbloquea una mecánica de exploración donde el jugador puede ver con total libertad a que se enfrenta cada vez que entra en un escenario, ayudando así para completar puzzles y acertijos (siempre y cuando el campo mágico de fuerza lo permita, habrá escenarios que deberemos ir explorando paso a paso). Habrá escenarios donde el jugador esté obligado a explorar en el **Estado de Alma** para poder informarse sobre cómo superar la pantalla, donde podrá ver los enemigos a los que se va a tener que enfrentar, los puzzles que tiene que superar, qué enemigo le conviene poseer para cada momento, etc. Para este prototipo solo se ha planteado que este **Estado de Alma** sea completamente exploratorio, pero en un futuro se podría llevar más allá esta mecánica, añadiendo elementos que solo existan en el **Estado de Alma** (como ver las almas atrapadas de otras criaturas, interactuar con cosas que en el mundo físico no puedas, etc.).

Por otro lado, cuando el jugador está en **Estado de Cuerpo**, su movimiento, capacidades, modo de atacar y vida van a depender de la criatura a la que esté controlando en cada

momento. Para facilitar todas las opciones que pudiéramos tener, cada criatura diferente no tendrá un set muy amplio de habilidades o ataques, ya que sino sería muy difícil aprender todas las habilidades de cada criatura que podemos controlar. Por ello, se va a utilizar una lógica sistemática, donde por ejemplo cada criatura tenga un ataque normal y quizás una habilidad especial. Así facilitará al jugador para ver qué posibilidades tiene con cada criatura y así saber cual es la más óptima para cada momento (sería fácil aprender qué hace cada una, o sino probarlo sencillamente). También se utilizaran los mismos controles para cada criatura, por ejemplo, atacar con la tecla “E”. Luego, dependiendo de la criatura, el jugador tendrá también diferente velocidad de movimiento, salto, capacidad de volar, nadar, etc.. Finalmente, al poseer un cuerpo/cadáver, este tendrá una vida dependiendo de qué criatura sea (por poner un ejemplo, no va a tener la misma vida un caballero con armadura, que un insecto). Si nos quitan toda la vida, el Alma del jugador será expulsada de ese cuerpo pasando de nuevo al **Estado de Alma**. También, el jugador puede abandonar voluntariamente el cuerpo que tiene poseído para volver al **Estado de Alma**. Esto va a servir para generar versatilidad en la mecánica principal, dando la oportunidad al jugador de cambiar entre Estados según le convenga.

3.3.3. Condición de victoria y derrota

De seguido, hay que tratar un tema muy importante del juego: ¿Cómo pierde el jugador? El jugador no puede morir ya que en **Estado de Alma** este es inmortal, y si nos derrotan estando en un cuerpo en el **Estado de Cuerpo** simplemente nos expulsarán de él y volveremos al **Estado de Alma** de nuevo. Por eso, se ha planteado un sistema de energía. El jugador tendrá **X** energía. Cada vez que este sea expulsado del **Estado de Cuerpo** o salga voluntariamente de él perderá energía. De esta manera, la mecánica de poseer un cuerpo y abandonarlo voluntariamente tendrá un coste para así no abusar de ella, por ejemplo, si vamos a recibir un golpe en el **Estado de Cuerpo** y queremos evitarlo nos salimos del cuerpo, y cuando pase el ataque volvemos a entrar el cuerpo. Podremos hacerlo pero perdiendo energía, así que no se podrá abusar de ello porque perdemos toda la energía (sino el juego sería muy sencillo). Cuando esta energía llegue a 0, el Alma del jugador se desmayará, reapareciendo así al principio del nivel y reiniciandolo a modo de penalización (simulando el efecto muerte de muchos juegos al perder toda la vida). Si nos derrotan continuamente mientras estamos en el **Estado de Cuerpo,** no podremos avanzar. Esta energía va a estar representada con una barra al igual que la vida que tendremos al poseer un cadáver. Hay una manera de recuperar energía y es cuando derrotemos enemigos, ya que al estar “sacándoles el alma”, tendremos la capacidad de recolectar esta y guardarla en forma de energía para nosotros mismos. Esto tiene todo el sentido del mundo ya que nosotros al salir de un cuerpo o al ser expulsado del mismo si nos derrotan, simularemos como un cuerpo está perdiendo su alma, por eso tiene sentido perder energía en ese preciso instante. En resumen, nuestra vida principal se mide en energía, cada vez que nos expulsen de un cuerpo por ser derrotados (por perder la vida que tenga ese tipo de criatura) o salgamos voluntariamente de él perderemos energía y al derrotar a enemigos recuperaremos energía, sencillo y efectivo.

Para finalizar con las mecánicas, hay que tratar una situación muy importante: ¿Que ocurre si no hay cadáveres para poseer? Para evitar un escenario donde no sea posible seguir

jugando ya que el jugador no tiene cadáveres a su alcance (estando en **Estado de Alma** no podría interactuar con nada), se añadirán una especie de seguros de vida, como pueden ser tumbas que contienen cuerpos infinitos, donde el jugador puede poseerlos tantas veces como quiera. Así nunca se cerrará la posibilidad de no tener cuerpos al alcance en modo alma. Otra opción es que exista una posibilidad de reiniciar el nivel/escenario donde nos encontremos si hemos perdido algún recurso necesario para superarlo. Por último, otra solución sería la siguiente: puede ser que perdamos un cuerpo si se cae al vacío (imposible de recuperar). En ese caso, se detectaría la pérdida del cuerpo de ese enemigo, generando otro idéntico en el lugar donde estaba ese cadáver inicialmente. La idea de esto es que el jugador no se quede nunca sin posibilidades, habilitando de esta manera tener un recurso a nuestro alcance todo el rato. Para esta solución se ha tenido en cuenta una vez más en el juego de Kirby's Adventure Wii[11], ya que en éste juego, antes de un jefe importante, se ofrece diferentes poderes por si no se ha podido llegar a él transformado con el poder de algún enemigo.

3.4. Niveles del juego

Al ser un Metroidvania 2D, el juego no debería contar con niveles separados, sino más bien pantallas diferentes que formen un mundo entero, con sus diferentes zonas. En este caso, para este prototipo es imposible crear un mundo grande y representar correctamente cómo sería esto en el juego final. Es por ello que en vez de plantar un mundo con escena interconectadas entre sí con diferentes zonas, se van a plantear dos niveles cerrados para el prototipo: un nivel tutorial como se ha explicado anteriormente, y un nivel principal que proponga un reto y enseñe mediante su gameplay las posibilidades y oportunidades del juego, tal y como lo debería hacer un prototipo. Entonces, a la hora de jugar la juego entraremos al nivel tutorial donde nos familiarizamos con el juego. Tras superar este (habrá un objetivo, un punto final) pasaremos al nivel principal en el que tendremos la misma meta u objetivo, pero esta vez más difícil de alcanzar, con obstáculos, puzzles, enemigos, etc.

3.5. Representación de la propuesta

En la siguiente Figura, podemos ver un ejemplo de representación del estilo del juego sacada el juego de Hollow Knight[1]. En esta tenemos redondeado en Rojo una interfaz que nos mostrará datos relevantes como la energía y que tenemos y la vida de la criatura que estamos controlando. En azul, podemos ver la posible distribución de plataformas en el escenario, ya que al ser un metroidvania el juego tendrá mucha zona de plataformas. En Naranja podemos ver el ejemplo de un enemigo al que podemos derrotar para luego posteriormente poder poseer su cadáver. En verde tenemos la representación del jugador en un ejemplo de **Estado de Cuerpo** donde estamos dentro del cuerpo de un insecto. Finalmente, el juego tendra obstaculos como podemos ver en la imagen de color morado, los cuales van a servir para crear el escenario y sus puzzles junto a las plataformas.

Figura 5: Ejemplo visual de la estructura del juego

4. Diseño

4.1. Arquitectura general del juego

El juego se va a desarrollar utilizando el motor gráfico de Unity, motor que se ha utilizado cursando el máster. En este caso, se va a utilizar el potencial de Unity para desarrollo en 2D. Siguiendo la idea del proyecto, se va a desarrollar un MVP (Minimum Viable Product). Este prototipo va a enseñar las diferentes mecánicas y estéticas del juego, creando puzzles diferentes como la versión final del juego tendría. Es por ello que se ha utilizado una arquitectura basada en componentes, que se compone de los siguientes elementos:

- **Personaje Principal y Enemigos:** Estos elementos son representados por sprites y se encargan de la interacción del jugador con el entorno del juego. El personaje principal posee sus características especiales para enfrentar a los enemigos de forma creativa y diferente dependiendo del momento del juego (transformaciones).
- **Escenario:** El escenario va de la mano con los sprites del jugador y de los enemigos. Este, está formado por plataformas, obstáculos y enemigos. Se han planteado escenarios con diferentes temáticas y desafíos, dependiendo del momento del juego en el que se encuentre el jugador, pero de cara al juego completo. Para este prototipo se han planteado dos escenarios diferentes: un nivel de tutorial y un nivel que enseña el potencial del juego. Por ello, el prototipo estará basado en esos dos niveles. Se utilizan herramientas de nivel diseño para crear los escenarios.
- **Sistema de Físicas:** El juego utiliza el sistema de físicas de Unity para simular el movimiento y la colisión de los personajes y enemigos en el juego. Esto permite que los personajes salten (si tienen la habilidad de hacerlo), les afecte la gravedad, colisiones con el entorno, etc… Cada tipo de enemigo y jugador se verá afectado de forma diferente a las fuerzas que hay dentro del juego (dependen las masas de dichos personajes, su fuerza, tamaño, etc…).
- **Interfaz de usuario:** El juego cuenta con una interfaz de usuario que muestra información relevante al jugador, como la salud de este representada en energía. También incluirá menús, Pop-Ups y otros elementos de interacción.
- **Sonido y música:** El juego incluye efectos de sonido y música para mejorar la experiencia del jugador, siempre acorde de la atmósfera del juego. Esto puede incluir sonidos de saltos, ataques, impactos con el entorno, música de fondo, entre otros.

El motivo de la elección para desarrollar el juego Metroidvania 2D en Unity es que Unity es un motor de juego ampliamente utilizado y cuenta con una amplia gama de características y herramientas que hacen que sea adecuado para el desarrollo de juegos 2D. Algunos de los aspectos más importantes que justifican esta elección de Unity son:

- **Facilidad de uso:** Unity proporciona una interfaz gráfica intuitiva y fácil de usar, lo que facilita el desarrollo y la iteración rápida del juego, sobre todo en entornos 2D.
- **Multiplataforma**: Unity permite la creación de juegos para una variedad de plataformas, como PC, consolas y dispositivos móviles. Esto permite que el juego

Metroidvania 2D tenga la posibilidad de llegar a una amplia audiencia en un futuro tras su desarrollo (en el caso de este prototipo solo será lanzado para PC).

- **Comunidad y recursos:** Unity cuenta con una gran comunidad de desarrolladores y una amplia variedad de recursos, tutoriales y documentación disponibles. Esto facilita la resolución de problemas y el aprendizaje de nuevas técnicas y herramientas, así como la obtención de recursos gratuitos diferentes como sprites o sonidos que se van a poder utilizar en el proyecto.
- **Rendimiento:** Unity está optimizado para el rendimiento en tiempo real, lo que permite una experiencia de juego fluida y sin problemas.
- **Experiencia en su uso:** Unity es el motor gráfico que se ha utilizado en incontables asignaturas dentro del máster, por lo que la experiencia de uso es elevada a esta altura del curso, facilitando el desarrollo de un proyecto en el.

Por ultimo, tenemos que echar un vistazo a los requisitos técnicos del entorno de desarrollo:

- **Unity 2D:** El juego Metroidvania 2D se desarrolla utilizando el motor de juego Unity en su modo 2D. Unity proporciona herramientas y funcionalidades específicas para el desarrollo de juegos en dos dimensiones, como la gestión de sprites, colisiones, animaciones y físicas.
- **Lenguaje de programación:** El juego se desarrolla utilizando C# como lenguaje de programación principal en Unity. C# es un lenguaje de programación potente dirigido a objetos, versátil y que se integra perfectamente con Unity y permite desarrollar la lógica del juego, la interacción con los elementos del juego y la implementación de las mecánicas de juego.
- **Herramientas de diseño gráfico:** Para crear los gráficos y sprites del juego, se utilizan herramientas de diseño gráfico como GIMP, Paint-3D o la web Photopea, todos gratuitos. Estas herramientas permiten crear y editar los elementos visuales del juego, como personajes, enemigos, fondos y objetos.
- **Herramientas de edición de audio:** Para el sonido y la música del juego, se utiliza una herramienta de edición de audio como Audacity. Esta herramienta permite crear y editar efectos de sonido, música de fondo y otros elementos de audio para mejorar la experiencia del jugador.
- **Herramientas de guardado de versiones:** Para la organización de versiones del proyecto y copia de seguridad de estas se utiliza la herramienta Sourcetree. Esta herramienta permite crear diferentes Commits y subirlos directamente a la página de GitLab donde se organizan todas las versiones del proyecto.

En resumen de este apartado, el prototipo Metroidvania 2D desarrollado en Unity sigue una arquitectura basada en componentes, donde se utilizan diversos elementos como personajes, escenario, físicas, progresión, interfaz de usuario, sonido y música. La elección de Unity como entorno de desarrollo se basa en su facilidad de uso, soporte multiplataforma, comunidad y recursos, rendimiento, experiencia de uso y la integración con el lenguaje de programación C#.

4.2. Diseño general del Prototipo

En cuanto al diseño general del prototipo, se ha planteado crear un escenario que permita encajar una historia completa de manera factible. Este escenario debe contener los elementos necesarios para dar vida a esa historia y al universo del juego, de modo que los eventos del juego puedan desarrollarse de manera coherente. Muchos juegos se crean sin pensar en una historia o universo que los respalde, simplemente se enfocan en el juego en sí. Sin embargo, en este caso, se ha planteado tener una base sobre la cual trabajar, aunque no sea muy extensa. Por eso, se ha diseñado un pequeño universo donde se pueda encajar la historia que el juego pretende contar. Este universo o entorno no es muy amplio, simplemente sirve como una plantilla sobre la cual se pueda trabajar en el futuro sin interferir con el juego y siempre respetando sus fundamentos. Muchos juegos no hacen esto, y luego, al intentar explicar su origen, crear una historia o desarrollar secuelas, se encuentran con incoherencias entre los elementos del juego, situación que pretende evitar.

4.2.1. Historia y Entorno

El entorno establecido para este proyecto es un universo similar al nuestro, pero con ciertos cambios. Dentro de este hipotético universo, en un planeta o mundo específico, tiene lugar este juego. Este mundo se asemeja mucho a nuestro mundo durante la época medieval. Está dividido en diferentes reinos habitados por humanos y criaturas, cada uno con su propia civilización, normas y costumbres. Estos reinos están claramente definidos, ya que fuera de ellos el mundo es salvaje y hostil, con criaturas fantásticas que se explicarán más adelante.

De este modo, establecemos el origen de nuestro personaje principal y de los enemigos hostiles que encontraremos en el mundo. Como se mencionó anteriormente, el personaje principal es un caballero humano perteneciente a un reino, encargado de protegerlo de amenazas externas. Este reino es atacado por un demonio, momento en el que el personaje decide intentar defenderlo, y produce a caer derrotado desencadenando los sucesos contados en el apartado de Propuesta. Este demonio es un tipo de criatura completamente posible dentro de este universo, ya que las criaturas que encontramos en él son diferentes a las del planeta Tierra, tal y como se ha explicado. Aun así, estas criaturas pueden ser consideradas como animales del planeta Tierra. Esto es porque cada tipo de criatura tendrá su hábitat natural, y vivirán en grupo o en solitario. Lo que las distingue de los animales del planeta Tierra es que la mayoría de las criaturas que encontramos en el juego son fantasiosas, territoriales y hostiles. Esto se ajusta a la idea de que existan diferentes reinos individuales, cerrados y protegidos contra posibles amenazas externas. Es similar a los reinos de la edad media, con sus fortificaciones y preparaciones para enfrentar amenazas externas a sus murallas.

Además de esto, la magia estará presente en este mundo como algo común. Por ejemplo, el demonio tendrá la capacidad de separar nuestro cuerpo y alma como se describe en la propuesta. También existirán objetos que puedan interactuar con el personaje en modo alma, como el muro mágico que lo detiene entre otros ejemplos. Todo ello no significa que

haya magos y magia por todas partes, pero sí que estas prácticas denominadas “Magia” en el planeta Tierra pueden ser una realidad dentro de este mundo.

En resumen, el juego se desarrolla en un mundo con una distribución similar a la edad media de La Tierra, con diferentes reinos, criaturas fantásticas, fuertes y hostiles que dominan las diferentes zonas del mundo, e incluso la presencia de la magia. Sin duda, es el escenario perfecto para un juego de acción en tiempo real como este.

4.2.2. Atmósfera general del juego

Para que coincida con el entorno mencionado anteriormente, es importante crear la atmósfera del juego de manera adecuada. Aquí es donde comienza la elección de los elementos visuales para el juego, también conocidos como assets. Siguiendo el entorno descrito, el mundo estará compuesto por diferentes biomas y entornos, que darán forma a los territorios de los distintos reinos y los hábitats de las criaturas que los habitan. Dado que solo se creará un prototipo, el prototipo se enfocará en un solo entorno o bioma.

Este bioma estará protagonizado por cuevas, un lugar oscuro, húmedo y melancólico. De esta manera, se podrá representar un lugar donde la vida es difícil y peligrosa, un lugar habitado principalmente por criaturas no humanas. Aquí se encontrarán elementos más peligrosos de lo que se encontrarían en una cueva del mundo real, ya que en este universo los peligros y las criaturas peligrosas están mucho más presentes. Por ejemplo, en lugar de haber lagos y ríos de agua corriente como en una cueva normal, estos serán de aguas ácidas. Esto tiene su explicación, y en este caso es una criatura que habita en el bioma: los Slimes. Esta criatura explicada a continuación, contamina la zona, la mantiene húmeda y la intoxica, creando así el entorno o bioma mencionado.

Vamos ahora con los enemigos. Vamos a encontrar 3 diferentes dentro de este bioma:

- **Slimes:** Masas amorfas gelatinosas. Es una criatura lenta pero con gran capacidad de salto. Son los causantes de que el bioma esté en esas condiciones ya que su baba genera las toxinas que contaminan el ambiente. Ellas por sí mismas no son realmente tóxicas, sino que la combinación de su baba con diferentes minerales de la tierra crean el bioma tóxico y sus aguas ácidas. Aun así, por su tipo de cuerpo pueden vivir sin problema en este ambiente, he incluso vivir bajo el ácido de los ríos de agua ácida ya que su composición gelatinosa no se ve afectada por él.
- **Dragones:** Dragones un poco peculiares ya que son de tierra, no pueden volar. Aun así, sí que tienen la capacidad de escupir fuego. Son pesados, lentos y les cuesta empezar a moverse y saltar. Una vez cogen ritmo, tienen una buena velocidad punta. Son bastante resistentes. Viven en esta zona ya que se alimentan de los Slimes, están preparados para digerirlos y que sea un alimento para ellos. A parte de eso, gracias a los slimes que comen son capaces de crear el aliento en llamas que les permite lanzar fuego. Eso sí, no son capaces de entrar a las aguas ácidas de la zona, ya que su dura piel no está preparada para ello y mueren al entrar en contacto con ellas..
- **Goblins:** Una pequeña tribu de duendes con mazas que trabajan en grupo para cazar. Son como una especie de reino desorganizado, muchos años atrasados en

comparación con los humanos. Son muy hostiles. Individualmente son débiles, pero rápidos y hábiles. Normalmente van en grupo. Viven en este hábitat ya que cazan a los dragones para alimentarse. Sus pieles tampoco están preparadas para tocar el agua ácida, pero si lo utilizan para elaborar pociones y herramientas.

4.2.3. Assets elegidos

Ahora, ya con la atmósfera creada y el entorno y las criaturas planteadas es hora de elegir unos assets apropiados para representar la idea creada, teniendo en cuenta de que deben estar relacionados entre sí, manteniendo así una armonía de diseño. En este caso, se ha elegido utilizar la técnica de arte Pixel Art, de modo que todas las ilustraciones como entorno, nivel, criaturas, enemigos y personajes van a ser representados con sprites en Pixel Art. Estos han sido los assets seleccionados y creados para este proyecto:

- **Elementos del nivel:** Para representar el hábitat del nivel, se ha seleccionado un Tileset gratuito de la página itch.io creado por Brullov Studios[13]. Este sileset es simple pero tiene los elementos perfectos para representar un bioma de cuevas:

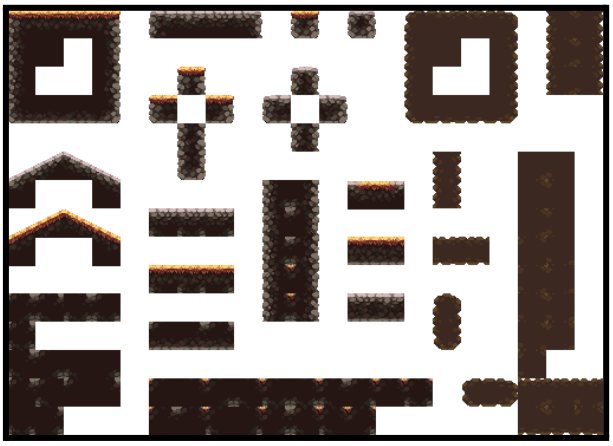

Figura 6: Tileset creado por Brullov Studios

Para acompañar a este Tileset, se han utilizado de elementos decorativos, también gratuitos, del set Pixel Art Platformer - Village Prop de Cainos de la Asset Store[14]:

Figura 7: Pixel Art Platformer - Village Props de Cainos

Finalmente, para crear el agua ácida que va a dar ese toque peligroso al bioma, se han creado manualmente unos Tiles para representarla:

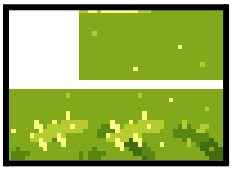

Figura 8: Tiles Agua ácida

- **Personaje Principal:** Para representar el personaje principal, sólo hay que crear el fantasma que el jugador va a controlar. Dado que las acciones de este personaje son simplemente moverse y transformarse, se ha decidido crear presonalmente el personaje y sus animaciones basándose en varios diseños encontrados en internet (sin copiar ninguno, simplemente cogiendo ideas):

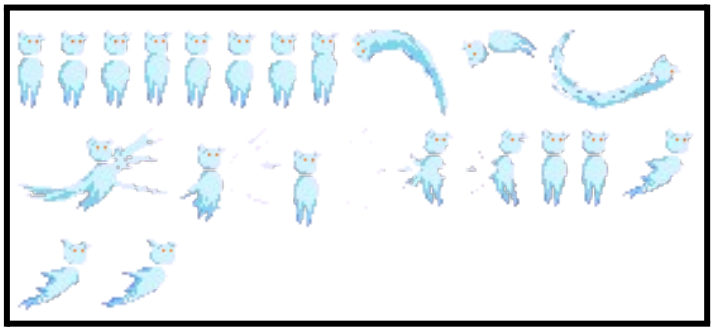

Figura 9: Fantasma

- **Goblins:** En este caso, las animaciones de este personaje eran más extensas y complicadas, por lo que se ha decidido buscar un asset que encajase con la propuesta, encontrando el Fantasy enemies pack de Lilwillydesigns[15] y su goblin:

Figura 10: Goblin de Fantasy enemies pack de Lilwillydesigns

- **Slimes:** El anterior paquete también incluye el diseño y animaciones de un slime, pero en este caso es de pago, por lo que se ha optado por elegir uno diferente: High fantasy - Slime enemy de Warsvault[16]. Aun asi, el tamaño de los sprites de la animación de muerte no encajaba, por lo que se ha ajustado manualmente:

Figura 11: Slime de High fantasy - Slime enemy de Warsvault

- **Dragones:** Finalmente, para el diseño y animación de los dragones, se ha utilizado el paquete 2D pixel art Drake sprites de Elthen[17]. El problema de este paquete es que el tamaño de pixeles no coincide con el resto de elementos, por lo que se ha

tenido que ajustar cada sprite manualmente para que se ajuste correctamente, a parte de añadir ciertos detalles a algunos sprites para el juego:

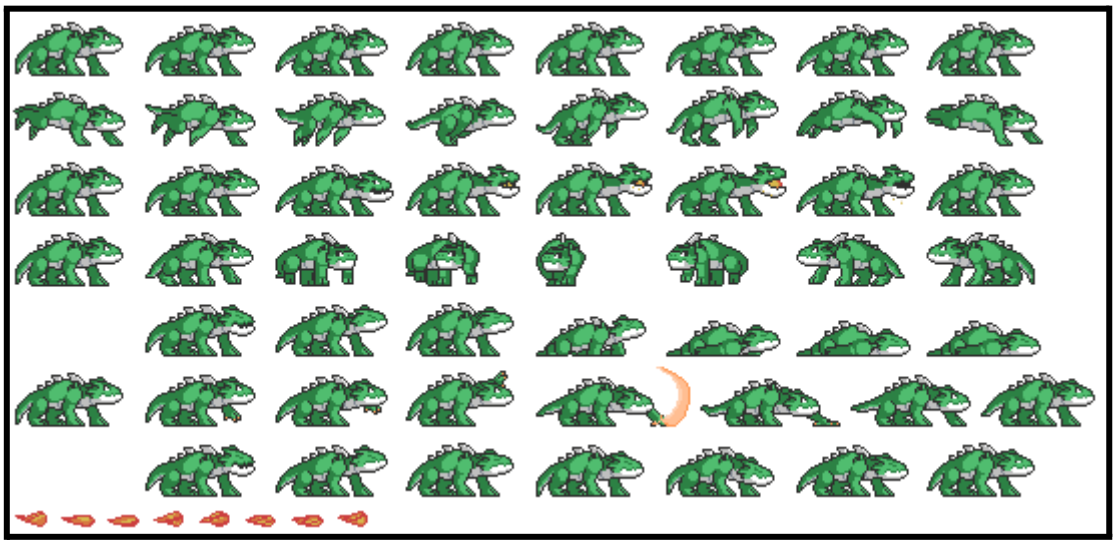

Figura 12: Dragon de 2D pixel art Drake sprites de Elthen

4.2.4. Diseño de los niveles

Una vez establecidos los Assets y sus animaciones, es el turno de crear unos escenarios/niveles acorde a la descripción de nuestro prototipo. En este caso, el prototipo va a constar de dos niveles diferentes: Un nivel tutorial que muestre los controles y las mecánicas del juego, y un nivel principal que será el encargado de enseñar el potencial del juego, con diferentes enemigos, puzzles y mecánicas.

El nivel tutorial no va a tener mucha complicación. Va a ser un escenario sencillo, donde estará el personaje principal y se le enseñará la mecánica principal. Con un escenario plano servirá. Aquí se enseñará el manejo del fantasma al jugador. También, se podrán encontrar cadáveres de diferentes enemigos para así ver las posibilidades que existen a la hora de controlar estos. Por ejemplo, estará el cadáver de un goblin y de un dragón, de manera que se puedan poseer, y un obstáculo de madera que se tenga que quemar con el fuego del dragón, y una pequeña zona de plataformas que se tenga que superar con el goblin. Finalmente, se enseñará a controlar también al Slime y sus saltos. Con esto sería suficiente para enseñar al jugador controles de movimientos básicos, mecánica principal y posibilidades de los enemigos que controlemos. Este es el diseño del nivel tutorial:

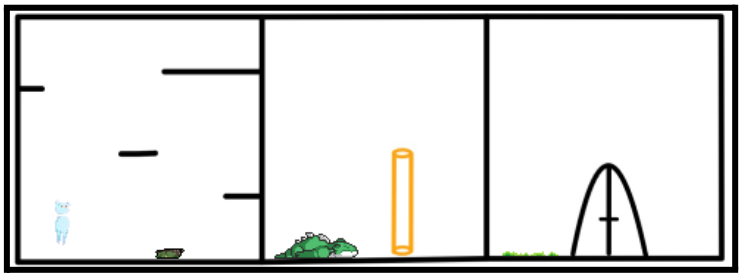

Figura 13: Diseño Del nivel tutorial

En cuanto al nivel principal del prototipo, se deberá establecer una jugabilidad con más reto, por lo que en este caso existirán enemigos que derrotar en vez de los cadáveres directamente. A parte de esto, la lógica del puzzle debe ser más completa, por lo que a parte de añadir un obstáculo ya mostrado como la madera que podemos quemar con el dragón, hay que añadir otro elemento obstáculo que nos requiera utilizar una nueva mecánica. En este caso, el nivel contendrá un charco de ácido que solo se podrá atravesar utilizando un Slime (como antes se ha mencionado en el diseño de la atmósfera). Para conectar todos los elementos y que solo haya una manera de superar el nivel haciendo bien el puzzle, la puerta del final estará bloqueada por una barrera con cerradura y una llave que se deberá de conseguir. Una vez pensada la lógica del puzzle del nivel, resta establecer los puntos donde se colocarán los enemigos y el comienzo del jugado. Hay que pensar que el jugador entra al nivel en modo de alma/fantasma, y que para poder interactuar debe poseer un cadáver. En el nivel tutorial tenía cadáveres directamente, en este nivel en cambio solo tendrá 1 disponible, por lo que tendrá que empezar a superar el nivel utilizando ese cadáver. Este es el Diseño del nivel principal del prototipo:

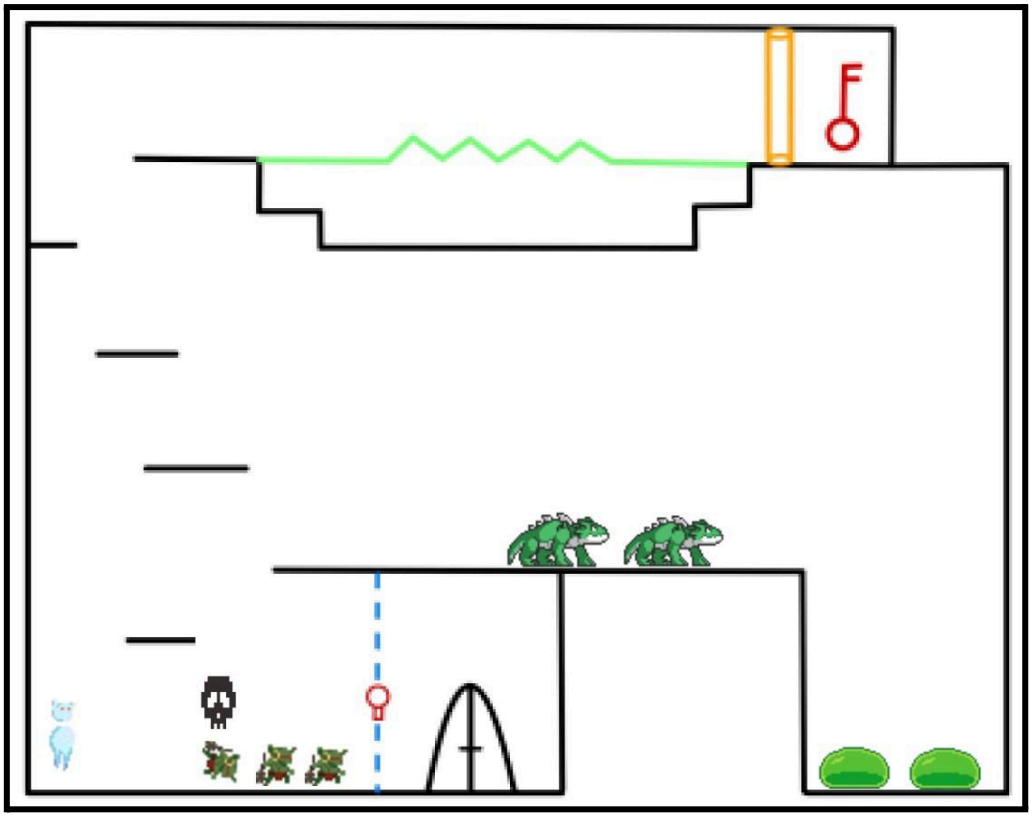

Figura 14: Diseño Del nivel principal

Estos dos diseños representan la idea inicial de los niveles del juego, pero siempre pueden sufrir variaciones a la hora de su implementación, ya sea por fallo en el diseño a nivel de jugabilidad, o adaptaciones para que el nivel suponga mejor diseño, reto, etc.

4.3. Diseño de la parte lógica de la programación

Continuando con la parte de programación del proyecto, esta está repartida en varios apartados ya que al ser un trabajo fin de máster enfocado en la programación, esta es muy compleja. Hasta ahora, a la hora de programar un juego bastaba con ir haciendo las cosas poco a poco, añadiendo funciones básicas y utilizando continuamente sentencias condicionales como los “IF”, pero para este proyecto hay que ir un paso más adelante. Es sencillo programar un personaje principal utilizando sentencias como “If(Space) then Jump” y continuamente así, pero a la hora de la verdad es muy poco eficiente, ya que se estarán haciendo comparaciones inútiles todo el rato, ejemplo: ¿porque se debería mirar si el jugador está intentando saltar, si en ese momento está nadando?. Por ello, este tipo de programación funciona muy bien para un juego sencillo, pero cuando hablamos de un juego al completo no se puede realizar de esa manera. Entonces, como este prototipo tiene como objetivo enseñar el potencial de un producto completo final, la programación de este debe ir de la mano con ello haciendo una propuesta de programación avanzada, entonces se va a hacer uso de máquinas de estado complejas para manejar todos los posibles estados del personaje principal, los enemigos y diferentes elementos como UI, menus, etc.

4.3.1. Máquina de estados del jugador

Para crear la máquina de estados del jugador se ha creado una sintaxis sencilla y efectiva teniendo en cuenta de que se trata de una máquina de estados. Con ello, se puede programar cada estado diferente del jugador en un Script individual, y cambiar entre estos dependiendo de la situación del jugador. Todos los Scripts van a compartir funciones bases del jugador, de modo que puedan hacer uso de ellas. El sistema es muy simple:

- Un script general Monobehavior con los datos del jugador y con las funciones base de Unity, como las funciones Start o Update.
- Un script base abstracto con funciones abstractas que va a definir cada estado.
- Dentro del script general se va a ejecutar el código de las funciones de cada script de cada estado, por ejemplo, unas funciones clon Start y Update.
- Solamente se va a ejecutar las funciones del estado activo en ese momento.
- El script general crea una instancia de un estado concreto para pasarle los datos y el contexto de la clase Monobehavior para su funcionamiento.

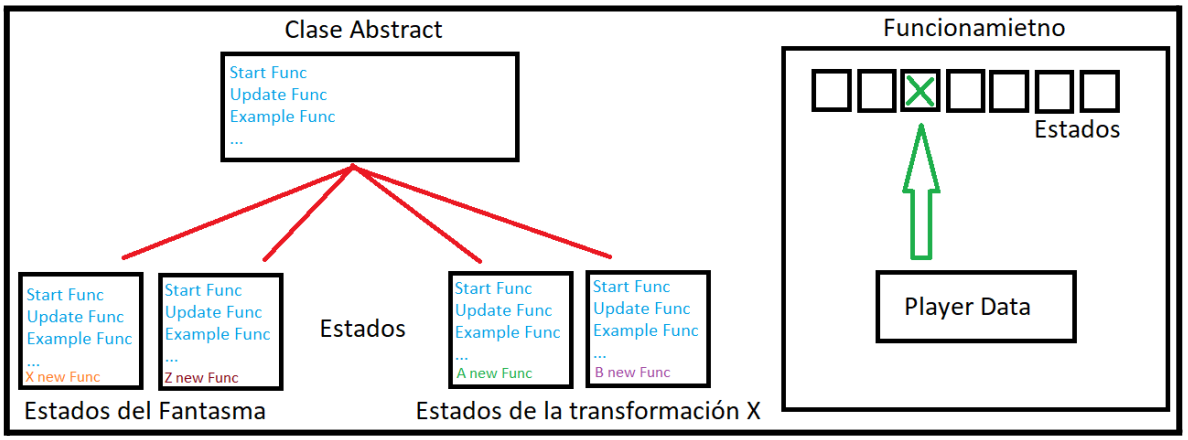

Figura 15: Diseño de la maquina de estados del jugador

De esta manera se consigue que si el jugador está en modo dragón, tenga en cuenta sólo las posibilidades que tiene en este modo: disparar fuego, atacar con la garra, moverse y saltar. En cambio, si el jugador está en modo alma/fantasma, se debe mirar que pueda volar hacia todas las direcciones y transformarse cuando esté junto a un cadáver. Es tan simple como eso, dependiendo de lo que esté haciendo el jugador, ejecutar el código de un Script u otro. Esto también facilita las transiciones entre transformaciones: al pasar de fantasma a dragón, hay que modificar el collider del jugador para que se adapte a la nueva forma y para que no atraviese cosas. Aparte, hay que establecer la masa y gravedad acordes a la nueva transformación, así como su velocidad y potencia de salto, cosa que el modo de alma no tiene. De esa manera, se consigue que con un solo objeto unido al jugador se puedan controlar todas las acciones de este, sus transformaciones y sus estados diferentes. El jugador solo necesita tener el script general de la máquina de estados, ya que este tiene referenciado una clase por cada estado, y pone en funcionamiento cada uno cuando es necesario.

4.3.2. Máquina de estados de los enemigos

Para crear la máquina de estados de los enemigos se ha replicado el mismo modelo que el de la máquina utilizada para el jugador, con ciertos cambios. Aunque ambas máquinas utilizan casi las mismas funciones, su funcionamiento general es diferente. En el caso del jugador, se debe tener en cuenta cualquier transformación que este pueda hacer, creando estados diferentes por cada criatura y acciones de esta. En el caso de los enemigos es igual, con la diferencia de que cada enemigo solo se va a mover por los estados que componen a ese tipo de enemigo, aunque la máquina de estados sea la misma para todos. Por ello, esta máquina de estados de los enemigos va a tener a todos los enemigos, pero solo podrá utilizar los estados del tipo de enemigo que esté controlando la máquina. Esto se ha planteado de esta manera ya que de lo contrario había que crear máquinas de estados para cada enemigo prácticamente idénticas unas a otras haciendo redundancia innecesaria.

Para hacer funcionar esto de esta manera se han utilizado los prefabs de unity, creando un tipo de enemigo por cada enemigo teniendo en cuenta sus parámetros únicos. Esto es muy cómodo en el momento de tener que tocar parámetros para un solo tipo de enemigo, al mismo tiempo que eficiente.

Para la IA de los enemigos, se ha programado una IA básica, pero con sus peculiaridades. El comportamiento de los enemigos es común: Estos irán a por el jugador cuando este entre en su rango de visión, y le atacarán cuando estén en rango para ello. En el momento que derroten al jugador o pierdan a este de su campo de visión, volverán a su posición inicial. La parte peculiar de este comportamiento viene relacionada con las transformaciones del jugador. Cuando el jugador esté en modo Alma/fantasma será indetectable por los enemigos, y si este posee un tipo de enemigo, los enemigos del mismo tipo no le atacaran en un principio. Esto es bastante lógico ya que sino los enemigos estarían atacando a uno de su especie. Eso sí, esto tiene un límite, ya que si el jugador decide lanzar un ataque los enemigos detectarán un comportamiento extraño, y aunque seas de la misma especie que ellos irán a atacarte. Aquí se puede ver un pequeño esquema del funcionamiento de la IA:

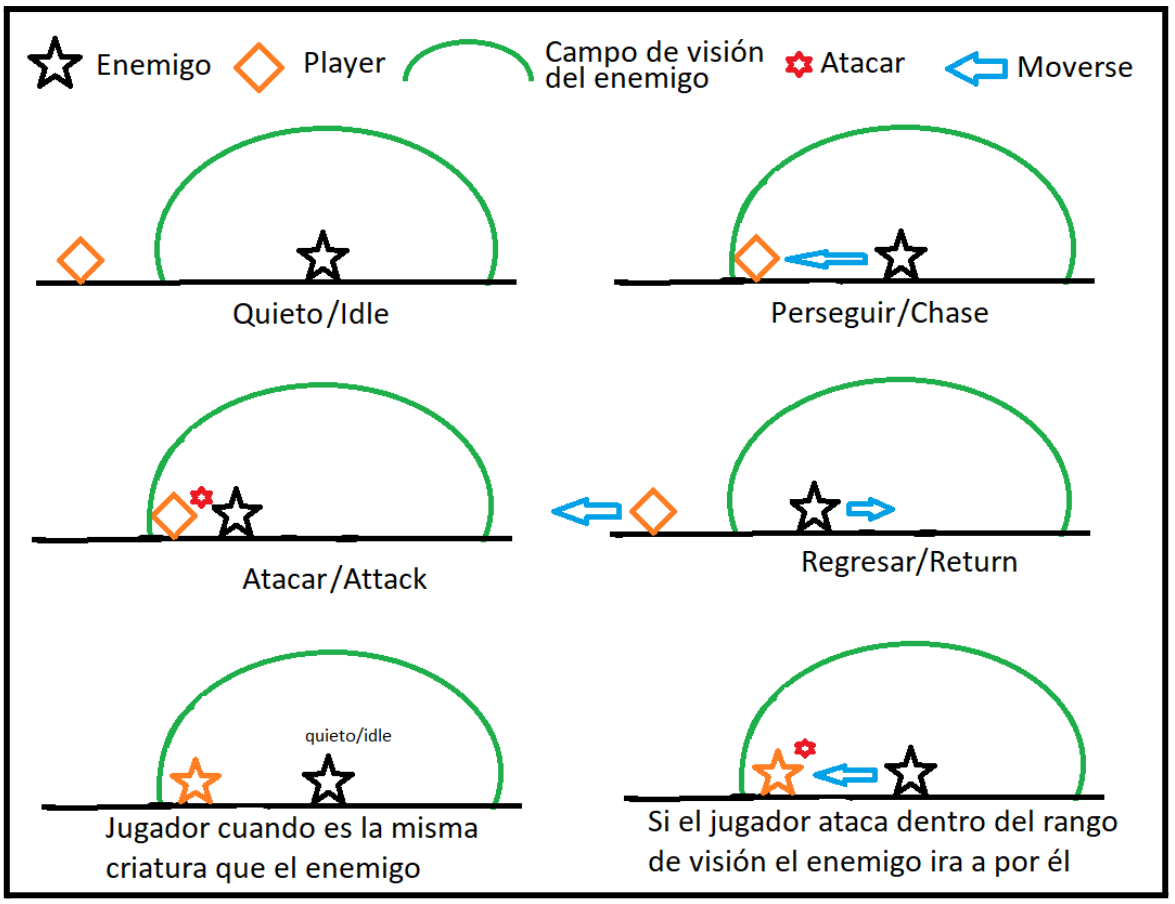

Figura 16: Diseño de la Inteligencia Artificial de los enemigos

4.3.3. Programación del resto de elementos

Otros elementos del juego tienen un diseño de programación más general sin perder en ningún momento la optimización. Estos elementos son por ejemplo la barrera que nos pone límites en el modo de alma, el ácido mortal (excepto para los slimes), las cajas que pueden arder o el sistema de llave y puerta. Para realizar estos elementos se va a utilizar generalmente el sistema de físicas y colisiones del motor gráfico. Otros elementos como menú principal, interfaz gráfica, manejo de sonidos y música han sido programados de manera básica combinándolos con los elementos fuertes de la programación como las máquinas de estados.

4.4. Organización del proyecto

Para finalizar con el diseño del prototipo, tenemos la organización general del proyecto, cosa muy importante a tener en cuenta. Una vez más, aunque el proyecto solo se trata de un prototipo, este representa el potencial de un juego al completo, por lo que debe estar bien organizado. Esto requiere tener un orden de assets y elementos dentro de la escena ya que el número de estos puede ser abismal en un trabajo completo (y realmente solo los de este prototipo no son pocos).

4.4.1. Organización de los Assets

Comenzando con la organización dentro de la jerarquía del proyecto, hay que poner atención en la carpeta de los Assets. Dentro de esta se encuentran todos y cada uno de los elementos a utilizar en el proyecto. Van a estar organizados en carpetas diferenciando todos los diferentes elementos entre ellos, y luego dentro de estas carpetas tendremos subcarpetas que organizan los elementos dependiendo del objeto al que pertenezcan. De esa manera, dentro de la carpeta Assets se encuentran las siguientes subcarpetas: Animaciones, Prefabs, Scripts, Sprites, … y por ejemplo dentro de la carpeta de los Scripts, se encuentran subcarpetas como Player (guardando todos los scripts que tengan relación con este), Enemigos (lo mismo que Player), etc…

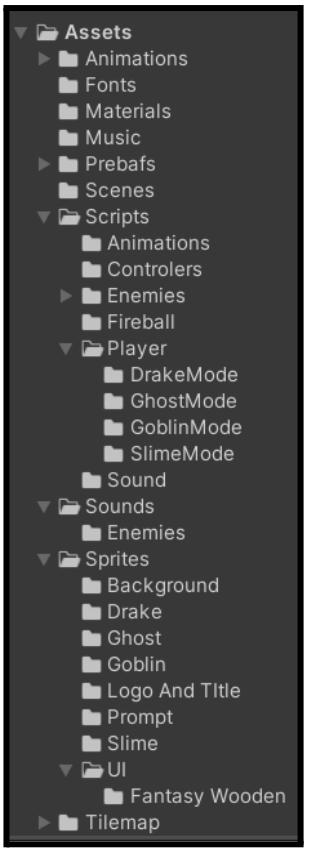

Figura 17: Organización de los Assets

4.4.2. Organización en Escena

En cuanto a la organización en escena, esta va a tener diferentes separadores para organizar los elementos diferentes. Por ejemplo, tendremos un empty Game Object para guardar dentro de él todos los elementos relacionados con el nivel y sus componentes (suelos, paredes, decoraciones) llamado *"Environment"*, otro elemento donde irá el jugador y todo lo relevante que tenga que ver con el llamado *"Character"*, otro objeto donde encontraremos a los diferentes enemigos, otro elemento para toda la interfaz gráfica y la UI, y elementos extra como el funcionamiento de la cámara, luces, etc… De esta manera es muy fácil lograr un orden y así localizar cada elemento dependiendo de las necesidades de cada momento. Un toque personal que se ha añadido a las escenas son elementos vacíos con guiones en su nombre, de modo que actúan de separadores entre los elementos, dando una organización mayor.

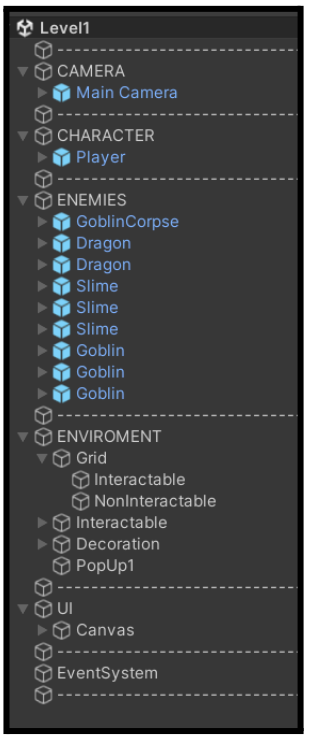

Figura 18: Organización de la escena

5. Implementación

5.1. Implementación primitiva del proyecto

Antes de comenzar a desarrollar el prototipo, se ha estudiado bien la idea, la competencia y estado del arte del proyecto, planteando cada punto de la implementación para realizar esta de manera dinámica. Después de todo el planteamiento y diseño, se ha dado comienzo a la implementación, en la cual los primeros pasos han sido crear la máquina de estados del jugador e implementar la mecánica principal que compone este juego. Junto a esto, esta fase ha sido la encargada de encaminar toda la programación del prototipo. Esta idea inicial se ha entregado en la PEC2 mostrando gráficamente la idea planteada para entender bien el proyecto. Al principio todos los elementos han sido representados por assets primitivos, permitiendo crear y ajustar la mecánica sin preocuparse por diseños de personajes, etc.

Para entender mejor la parte de la máquina de estados, aquí podemos ver como funciona con las líneas de código más relevantes:

- Script general o controlador de la máquina "**PlayerStateManager**". Este script es el que va unido al GameObject del jugador. Se puede ver que este script referencia un objeto del tipo "**PlayerBaseState**" que va a ser el objeto que referencie a cada estado de la máquina. Luego se pueden ver sus funciones más importantes como la función Start, el bucle Update y la función que se encarga del cambio de estados:

```
public class PlayerStateManager : MonoBehaviour
{
  PlayerBaseState currentState;

  public PlayerGhostMovingState ghostMovingState = new PlayerGhostMovingState();
  public PlayerGhostTransformState ghostTransformState = new PlayerGhostTransformState();

  public Rigidbody2D m_rigidbody;

  void Start()
  {
      currentState = ghostMovingState;
      currentState.EnterState(this);
  }
  void Update()
  {
     currentState.UpdateState(this);
  }
  public void SwitchStates(PlayerBaseState state)
  {
     currentState = state;
     currentState.EnterState(this);
  }
  …
}
```

- Script Base "**PlayerBaseState**" para los estados, con las funciones abstractas. Este script contiene la clase que va a definir a los estados y las funciones que van a compartir todos ellos. Luego, cada estado puede tener sus funciones diferentes:

```
public abstract class PlayerBaseState
{
  public abstract void EnterState(PlayerStateManager player);
  public abstract void UpdateState(PlayerStateManager player);
  …
}
```

- El ejemplo del estado "**PlayerGhostMovingState**", vemos las funciones del script base y el código del uso de esta clase concreta, que se encarga del movimiento del fantasma:

```
public class PlayerGhostMovingState : PlayerBaseState
{
  private float m_horizontal;
  private float m_vertical;
  private float speed = 15f;
  public override void EnterState(PlayerStateManager player)
  {
     //Init Ghost Movement Parameters
  }
  public abstract void UpdateState(PlayerStateManager player)
  {
      //Movement Example
      m_horizontal = Input.GetAxis("Horizontal");
      m_vertical = Input.GetAxis("Vertical");
      player.m_rigidbody.velocity = new Vector2(m_horizontal * speed, m_vertical * speed);
  }
  …
}
```

De esta manera, se puede ver mejor el funcionamiento de esta máquina de estados: Al inicio del juego, el GameObject del jugador tiene atribuido el script General "**PlayerStateManager**" de modo que este ejecuta su función del motor Start (ejecutada al inicio). Dentro de esta, está puesto que el estado inicial sea "**PlayerGhostMovingState**", estado que comparte las funciones de la clase base "**PlayerBaseState**". De esta manera, cuando se empiece a ejecutar el bucle Update de la clase "**PlayerStateManager**", esta ejecutara el codigo que haya dentro del Update del estado "**PlayerGhostMovingState**". Luego, basta con cambiar de estado cada vez que el jugador tenga que hacer una acción diferente, como transformarse. En ese momento, se hace el cambio de estado con la función `public void SwitchStates(PlayerBaseState state)` y se inicia el nuevo estado, del cual empezaremos a ejecutar su bucle Update después de iniciarlo con la función de entrada *EnterState(PlayerStateManager player)*. Toda la máquina de estados es mucho más extensa, compleja y completa, esto solo es un ejemplo de su programación y metodología de funcionamiento, tal y como se ha diseñado en el apartado anterior.

Esta parte de la implementación es muy laboriosa ya que lo más importante reside en crear la base de todos los elementos de manera correcta y ordenada, no hacer las cosas rápido y mal queriendo hacer todo el trabajo de golpe. Los primeros pasos dentro de la implementación de un juego son lentos, donde se deben crear las primeras clases, crear el orden de todos los elementos del juego como assets y elementos en el editor, etc. También hay que crear las bases de toda la programación. Es mucho trabajo para un resultado poco visual, pero es lo más importante de los primeros pasos del proyecto, al final la parte visual viene más tarde, cuando toda la parte interna del juego funciona correctamente. Si hacemos mal esta fase, esos errores se van a arrastrar durante toda la implementación, dificultando y obligando a invertir más tiempo que el necesario si se hubiese hecho bien la base. Aquí se puede ver cómo se ve esta fase del proyecto entregada en la PEC2:

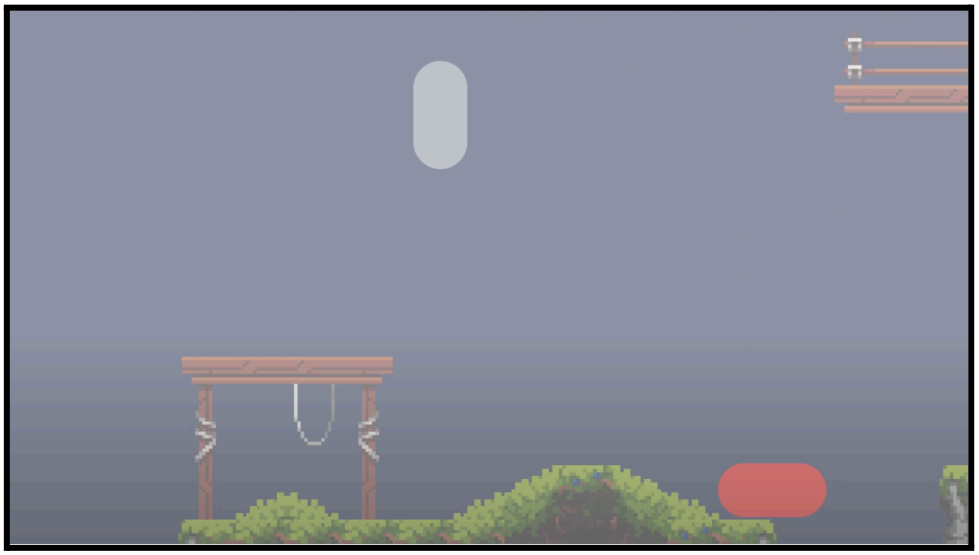

Figura 19: Primera fase de implementación del prototipo

5.2. Implementación de los diseños y animaciones básicas

5.2.1. Primeros Assets y Animaciones

Una vez ya está toda la programación y planteamiento de las mecánicas bien formadas, es hora de introducir los primeros assets. Esto ayuda a visualizar de mejor manera el juego, ajustar tamaños de los personajes, enemigos, escenario, calcular fuerzas de las físicas como la capacidad de saltos, movimiento, etc.. Estos assets no tienen porqué ser los definitivos, simplemente sirven de guía para empezar a ajustar todos los elementos mencionados anteriormente, es decir, sirven como un placeholder más avanzado y preciso que los assets primitivos. A parte de esto, viendo todos los assets juntos se puede ver de mejor manera si el juego contiene una buena armonía entre diseños y elementos, o si hay que seleccionar alguno diferente. En el caso de este prototipo, todos los assets seleccionados y creados desde el inicio han encajado muy bien con la idea y atmósfera del juego, respetando un grado de armonía entre ellos, por lo que se ha decidido mantenerlos como assets definitivos (por lo menos de cara a este prototipo, hay ciertos assets que no se podrían utilizar en caso de querer vender un producto ya que su licencia no lo permite).

Dentro de esta fase se introducen también las primeras animaciones importantes del juego, como movimientos del jugador y de los enemigos, ataques, saltos, muertes, etc. Esto le da movimiento al juego y le empieza a dar vida, viendo así cada vez mejor el punto al que se quiere llegar y el camino hasta llegar a él. Aun así, en esta fase solo se añaden las animaciones básicas e indispensables para el funcionamiento del juego, pero no es prioritario darles un acabado perfecto, ya que eso es la siguiente fase de la implementación.

5.2.2. Diseño de niveles

Otro punto importante de esta fase es comprobar si el diseño de los niveles está bien planteado, ya que una cosa es lo que se plantea en el diseño sobre papel y boli con un dibujo, y otra cosa muy diferente es si ese diseño es viable dentro del juego. En este caso, se ha respetado bastante la idea de ambos diseños (tutorial y nivel principal) simplemente modificando cosas puntuales para que el nivel esté bien adaptado a sus competencias (cambiando un poco la forma y la ubicación de un par de elementos). Para plasmar la idea del diseño de los niveles en el juego, se ha utilizado la herramienta del Grid y los Tiles de Unity, una herramienta muy cómoda que permite crear los niveles bloque a bloque. A parte de esto,se han creado varias capas para de esa manera separar elementos físicos y visuales dentro del nivel, consiguiendo profundidad.

Esta fase de la implementación es costosa porque hay que tener muchas cosas en cuenta, lo bueno es que ya existe la base sobre la que trabajar creada en la implementación primitiva. Otro punto positivo de esta fase es que el avance visual es mayor al de la fase anterior, viendo resultados tanto de esta fase como de los preparativos de la fase anterior. Una vez teniendo las mecánicas y los diseños implementados, el juego ya estaría implementado en una fase avanzada, con una versión completamente funcional. Aún así, aún le quedan los acabados para rematar todo y que esté en completa armonía. Aquí se pueden ver los resultado finales de ambos niveles en el editor de Unity:

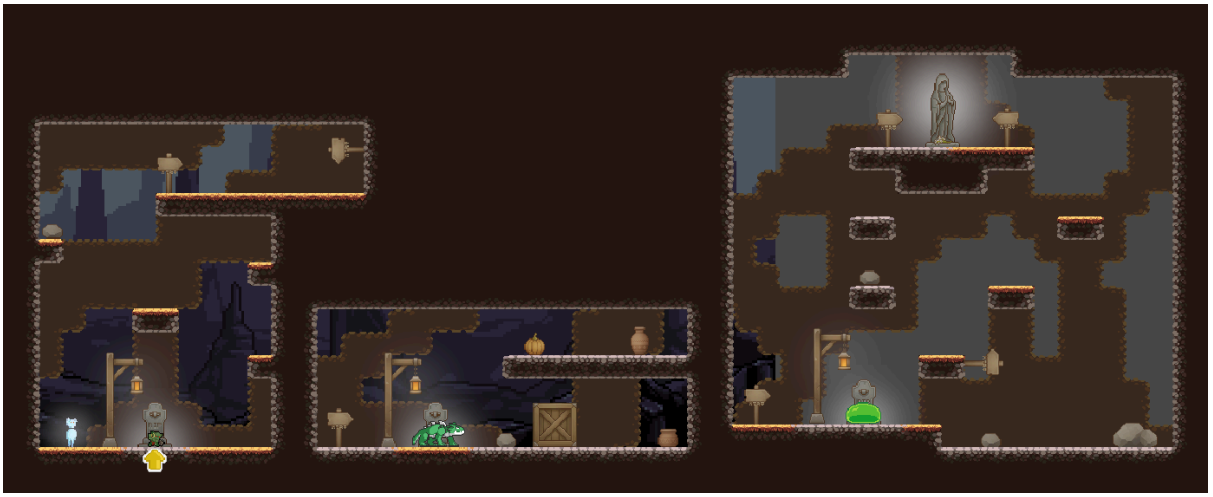

Figura 20: Implementación del nivel Tutorial

Figura 21: Implementación del nivel Principal

5.3. Implementación de las animaciones avanzadas y acabados

5.3.1. Animaciones avanzadas

Aunque en la fase anterior se hayan añadido las animaciones básicas, para que el juego tenga un buen acabado y sea visualmente agradable para el jugador hay que ajustar estas correctamente, por ejemplo, cuadrar los ataques cuando la animación de atacar golpea. Estos ataques no se pueden lanzar simplemente pulsando una tecla, sino que se debe ajustar al momento en el que la animación del jugador lanza el ataque. Hay ocasiones que esto ocurre en el primer frame de la animación, por lo que bastaría con lanzar la lógica de la programación del ataque en el momento que se pulsa la tecla de atacar, al mismo tiempo que se inicia la animación. Pero existe otro caso que es cuando la animación lanza el ataque en otro que no sea el primer frame de la animación, cosa que se ha tenido que tener en cuenta dentro del proyecto. En este segundo caso, es necesario sincronizar el momento de la animación en el que se lanza el ataque de verdad con el momento de lanzamiento de la lógica de la programación del ataque correspondiente. Esto se ha tenido que aplicar varias veces en el proyecto, no solo con los ataques, sino que también con el salto del Slime por ejemplo (ya que cuando se aprieta la tecla de saltar, el slime en primera instancia carga el salto y luego salta, y es en ese momento que hay que activar el salto). Por ello, el apartado de las animaciones es complejo y hay que sincronizarlo y prepararlo lentamente y cuidadosamente para lograr un buen acabado.

Continuando con las animaciones, Unity ofrece algo muy interesante que es activar eventos en los propios frames de las animaciones, una herramienta muy útil a la que se le ha dado uso. De esta manera, se puede hacer lo que se ha explicado previamente: activar la lógica de programación correspondiente a cada elemento en el frame exacto de la animación que este debería ocurrir (saltos, ataques). Además, no solo se pueden activar elementos, sino

que se pueden modificar componentes de los objetos en los frames de las animaciones. Esta herramienta también ha sido utilizada para ajustar por ejemplo el Collider del Slime con el cambio de forma que tiene a la hora de saltar, generando así colisiones reales y evitando que colisione donde no debería. Ya que esta herramienta te permite cambiar los parámetros que desees, también se puede utilizar para crear animaciones sin necesidad de diseños exteriores de sprites, como por ejemplo, animaciones de la cámara explicadas a continuación.

5.3.2. Acabados Visuales

Aunque en la fase anterior se hayan añadido las animaciones básicas, para que el juego tenga un buen acabado y sea visualmente agradable para el jugador hay que añadir ciertas animaciones extra más allá del ataque y salto de los personajes. Estas animaciones son las que le dan al juego un acabado competente para que sea visualmente atractivo. En este caso, se han incluido animaciones en la cámara para que acompañen acciones del jugador, por ejemplo, limitar la cámara para cuando el jugador está en el límite del nivel. De este modo la cámara no se desplaza más allá de donde no debería ya que no se ganaría información visual, sino que se perdería. Otras animaciones introducidas a la cámara son los efectos de zoom y luz que se aplican cuando el jugador entra y sale de los cadáveres, haciendo énfasis en esas acciones y potenciando la acción y su importancia.

Finalmente, hay que implementar efectos de luz y partículas, dando aún más vida y mejores acabados al juego. Estos efectos son muy importantes ya que acompañan muy bien las animaciones principales, por ejemplo, un efecto de partículas fantasmagóricas que acompañen al jugador en todo momento para que sea reconocible sobre los enemigos corrientes (Así el jugador sabe en todo momento quién es). A esto le acompaña una tenue luz, en el fondo del jugador, para reconocerlo aún más fácilmente. Otros efectos de luz y partículas sirven para exagerar las cosas importantes, como tumbas donde tenemos un cuerpo para poseer, objetivos como el final del nivel, peligros como un lago de agua ácida, etc. Tampoco hay que abusar de la colocación de estos elementos, ya que se podría saturar demasiado la escena consiguiendo un peor acabado.

Si bien esta fase tiene muchos elementos que tener en cuenta, es bastante más llevadera que las anteriores, ya que todo cambio que se introduce es sencillo de realizar porque no deja de ser una fase de prueba y error hasta conseguir los resultados deseados. Además, al ser cambios visuales constantes, parece que el avance del juego es mucho mayor que en las anteriores fases, al final se está trabajando de cara a la parte que se va a ver al finalizar nuestro juego. Si comparamos el juego con el cuerpo humano, se podría determinar que en las anteriores fases se ha construido gran parte del cuerpo interno como los órganos, los músculos y los huesos, y en esta fase es donde se empiezan a desarrollar las partes externas como la piel, la forma del cuerpo, etc. Por eso hay que tener cuidado en la implementación de todas las fases, ya que todas son primordiales y necesarias para construir bien el cuerpo, o en este caso el juego.

5.4. Implementación de la Interfaz Gráfica

Cada desarrollador tiene su manera y orden de trabajo. La manera de trabajo aplicada a este prototipo consiste en dejar la implementación de la interfaz gráfica para el final. Es cierto que si la interfaz gráfica tiene demasiada importancia en el juego, como puede ser un inventario en el que tengamos que utilizar objetos, se debe implementar mucho antes. En este caso, el prototipo tiene una interfaz muy sencilla que apenas tiene como propósito mostrar información al jugador, por lo que dejarla para el final de la implementación no ha dificultado el desarrollo de esta. La información que muestra esta interfaz si que ha sido programada en las primeras fases de la implementación, por lo que en esta fase solo resta mostrar esa programación visualmente. Esta información trata de la vida y la energía del jugador, así como los objetos recogidos (la llave en este caso).

5.4.1. Otros elementos gráficos

En esta fase también se ha introducido en modo de interfaz una parte primordial del nivel de tutorial: las explicaciones que muestran el funcionamiento del juego. Estos son simples Pop-Ups que se van a ir mostrando al jugador en el nivel del tutorial para explicarle objetivos, controles, y funcionamiento general del juego. De esa manera, se consigue ayudar al jugador al principio del juego para que no se frustre por no entender el funcionamiento del mismo (una posibilidad cuando faltan elementos como estos). De la mano con esto podemos encontrar otros elementos de acabado mencionados en la fase anterior, como la iluminación que acompaña a las explicaciones de estos Pop-Ups.

5.4.2. Pantalla de Título y Menú

Para finalizar con esta fase de la implementación quedaría la creación del Título y Menú principales. Esta parte no tiene demasiada complicación, pero hay que invertir tiempo en ella para cuadrar bien todos los elementos y diseños. En este caso se ha optado por una pantalla de Título con menú desplegable, haciendo una fusión entre estas dos implementaciones. Dentro del menú, se han configurado varias opciones: comenzar el juego, mostrar los créditos del mismo, ajustar el volumen general, el de la música y el de los efectos y finalmente un botón para cerrar el juego. Estos elementos básicos son lo mínimo que tendría que tener un juego como opciones dentro del menú principal. Se ha decidido añadir pequeños detalles y animaciones a ambas pantallas de modo que se ajusten más al juego y creen una buena atmósfera de entrada al juego.

5.4.3. Implementaciones extra

Para que la pantalla de Título principal no quedase demasiado vacía, se ha creado una imagen oficial del juego con su nombre, y un icono a raíz de esta (No se ha escrito simplemente el nombre del juego). El nombre del juego en este caso es **Iluna**, una palabra sacada del Euskera que significa “Oscuro”, referencia directa a la atmósfera del juego de este prototipo (atmósfera melancólica, oscura). Es una parte importante darle al juego una identidad, aunque en un futuro pudiese cambiar el nombre o el icono, es necesario entregar un prototipo con ello. Otra cosa que se ha decidido añadir es una marca sobre la que

producir el juego. En este caso la marca se llama **Garikasko Games** y es la desarrolladora oficial del producto de este proyecto, creada exclusivamente por el estudiante. Esta marca tiene un logotipo que creado manualmente y se muestra al inicio del juego, junto al logo de la Universidad y el de Unity (las 3 como *Splash-Images* de entrada al juego).

Figura 22: Logo oficial del juego

Figura 23: Logo de la marca Garikasko Games

5.5. Implementación de Musica y Sonidos

Para finalizar la implementación completa del juego, falta añadir efectos de sonido y música para ambientar el juego. Realizar esto teniendo toda la implementación anterior realizada resulta muy cómodo, ya que es más fácil acertar a la hora de crear y seleccionar los efectos y música ideales para el juego. Esta fase no tiene demasiada complicación más allá de seleccionar los mejores elementos para realizarla. Esto es porque toda la lógica y programación del control de volumen y efectos está realizada en fases anteriores, por lo que esta es exclusivamente la introducción de los elementos. Todos los elementos de sonido y música están libres de copyright y han sido adquiridos de páginas oficiales libres de copyright como Pixabay[18] .

5.6. Últimos acabados de la Implementación

Aunque el juego ya debería estar completo por parte de la implementación, siempre surgen imprevistos y retoques que hacer tras esta. Por ejemplo, algún elemento que no estaba del todo correcto y hay que reajustarlo. Lo que no se puede hacer es realizar cambios grandes de última hora ya que se debe cambiar tanto el producto. Un cambio grande habría que introducirlo fase a fase para no dañar la estructura del juego. Este punto está muy conectado con el siguiente apartado ya que muchos de los retoques o acabados realizados aquí surgen de las pruebas y test realizados sobre el producto final. Aparte de esto, una vez finalizada la implementación hay que añadir detalles finales, como limpieza de assets no utilizados o revisar el orden final del producto y en caso de haber algo mal reorganizarlo.

6. Demostración

6.1. Trailer

Se ha creado un trailer como tarea obligatoria del proyecto. En este trailer, no solo se enseña el prototipo sino que también se ha tratado de enfocar como trailer de una posible versión completa del juego, enseñando diferentes elementos tratando de generar interés y curiosidad por lo que tiene el juego que brindar a los usuarios. El trailer ha sido creado con diferentes escenarios y elementos que podemos encontrar dentro del prototipo, añadiendo música totalmente libre de copyright, en este caso, el tema *Epic Dramatic Action Trailer* de *QubeSounds* obtenido de Pixabay. Podemos encontrar el link al video en los anexos del documento, concretamente en el anexo de los entregables.

6.2. Descarga del Prototipo

El prototipo está subido a la página de GitLab. En ella podremos encontrar tanto el código fuente del proyecto, así como su primera versión 1.0.0 completamente funcional y siendo la versión definitiva de este proyecto. Depende de lo que se quiera descargar se encontrarán dos links. En uno se encuentra la descarga del código fuente en caso de que se quiera hacer revisión del mismo. En el otro link podremos encontrar la versión jugable del juego en una versión final completamente terminada. Ambos links podemos encontrarlos en los anexos del documento, concretamente en el anexo de los entregables.

6.3. Tests y pruebas realizadas

El prototipo ha sido probado y testeado por varias personas del círculo cercano. Es por ello que estas pruebas han sido bastante informales, pero suficientemente válidas como para recabar información para el juego. Se ha intentado que los perfiles de estas personas sean bastante diferentes entre sí para recibir diferentes feedbacks para realizar los últimos acabados del proyecto y comprobar resultados de su desarrollo. Dentro de estas personas se encuentran todo tipo de perfiles, desde fieles jugadores al género metroidvania, jugadores más casuales de videojuegos y personas que apenas han jugado un videojuego. En este caso, se han realizado diferentes pruebas y se han pedido feedbacks concretos a estas personas, para cubrir diferentes necesidades del juego.

6.3.1. Test de Rendimiento

Para comenzar, el primer test realizado ha sido el test de rendimiento, test necesario para comprobar la fiabilidad del juego bajo diferentes equipos u ordenadores. Este test demuestra si el código del juego y sus elementos están bien optimizados para su correcto funcionamiento, sin lag, perdidas de rendimiento u otros factores que dificulten su jugabilidad. Todo el feedback recibido por parte de los jugadores ha sido positivo, no ha habido ningún problema con el rendimiento del juego en ningún caso.

6.3.2. Test de Jugabilidad

De seguido, tenemos el test de jugabilidad. El objetivo de esta prueba es ver si los controles, las mecánicas del juego y su manejo general están bien implementados, son cómodos e intuitivos. Esta vez el feedback recibido ha sido variado, habiendo las mayores diferencias entre los distintos perfiles que han realizado las pruebas. El feedback más importante es el de los jugadores habituales de metroidvanias. Su respuesta a este test ha sido muy positiva, apoyando las mecánicas y los controles del personaje, opinando que su implementación es correcta y cómoda, siendo intuitivo el uso de los controles y claras las mecánicas. En el caso de los jugadores casuales de videojuegos el feedback ha sido prácticamente idéntico, estando acostumbrados a el uso de los mismos controles en otros juegos, así como los movimientos y acciones del jugador. En cambio, las personas que apenas han jugado un videojuego han tenido un feedback más negativo, alegando que el control es algo difícil. Esto es normal ya que como personas que no han jugado videojuegos se les hace algo complicado el uso de WASD como control de movimiento del personaje, y más añadiendo teclas vecinas para las acciones como la Q, E y R. En este test el feedback de este perfil de personas no es tan importante, ya que esto no es realmente un problema del juego, sino más bien de la falta de costumbre por parte de los testers.

6.3.3. Test de Diseño

Finalmente, tenemos la prueba de diseño. Una vez más, el feedback general de esta prueba ha sido positivo. Cabe destacar que parte del feedback de esta prueba han sido recomendaciones para la mejora del diseño, sobre todo por parte de los aficionados a los Metroidvania. La mayoría de personas de este perfil recomiendan añadir más elementos visuales al juego, alegando que el diseño pixel-art está muy bien, pero se puede explotar más a favor del juego. Lo que sí que les ha resultado extraño es que el juego consta de dos niveles en vez de un mundo, que tras la explicación de que el trabajo es un prototipo han llegado a entender que esto fuese así. En ese sentido sí que es cierto que han echado de menos un mundo cerrado con sus zonas, pero por razones obvias no se ha podido implementar algo de tales dimensiones. Por parte de los otros grupos el juego ha resultado cómodo y bonito visualmente, dando también sugerencias personales para la mejora de este en futuras versiones.

7. Conclusiones y líneas de futuro

7.1. Conclusiones

Para concluir con esta memoria, restaría la conclusión personal sobre el trabajo realizado. Esta conclusión es extensa, ya que el trabajo realizado ha sido muy grande, dando mucho valor personal.

Comenzando con la conclusión, creo que el trabajo realizado por mi parte ha sido muy bueno, al menos esa es la sensación que se me queda como realizador del mismo. Creo que el ímpetu, las ganas de aprender y las ganas de realizar un buen proyecto me han llevado a crear este gran trabajo. Las lecciones aprendidas por el camino son incontables, ya que hasta que no te pones a desarrollar un trabajo grande y serio (de un videojuego concretamente) no te das cuenta de las dificultades, las cosas a tener en cuenta, el tiempo de trabajo, etc. qué necesitas. Por ejemplo, una gran lección que he aprendido es que organizar los proyectos antes de realizarlos es muy importante, porque si decides trabajar sobre la marcha es muy fácil pisotear tus pasos anteriores, generando problemas que con una planificación se habrían evitado y teniendo que invertir más tiempo del necesario. Otra lección que he aprendido es que desarrollar un juego al completo requiere mucho mucho trabajo. Casi cualquier juego que podemos ver en el mercado tiene un gran trabajo detrás que quizás si no llegamos a vivir algo como lo de este proyecto no podremos llegar a ver nunca. He llegado a leer sobre juegos que realizan viajes largos solo para encontrar el instrumento perfecto para grabar un único sonido para su juego, como ciertos juegos realistas de la segunda guerra mundial que viajan en busca de un arma que se usó en aquel entonces simplemente para grabar el sonido de su disparo. Todo eso solamente por un efecto de sonido. Ese es el verdadero arte de crear un videojuego, el trabajo y cariño que tiene detrás cada uno de ellos.

Siguiendo con los objetivos del juego, podría dar mi cabeza por satisfecha ya que he logrado realizar el trabajo que planteé desde el principio. He logrado todos los objetivos propuestos, proponiendo soluciones y desarrollos adecuados para todos ellos. Cabe destacar que ciertas cosas podrían haber tenido quizás un mejor acabado, ir más allá, pero la realidad es que el tiempo para realizar el trabajo era limitado, por lo que era mejor centrarse en terminar todos los objetivos, y luego ya rematar estos para darles un acabado excepcional. Luego, en las líneas de futuro comento todas las cosas que me habría gustado incluir en el trabajo, así como las mejoras de cara a una segunda versión, o el desarrollo al completo del juego, nunca se sabe.

Haciendo una reflexión crítica sobre el seguimiento de la planificación y la metodología del proyecto, está ha estado un poco descalibrada en términos de tiempo. He de decir que ciertas circunstancias han dificultado el progreso del trabajo, como empezar a trabajar en una empresa a jornada completa, algo que no estaba previsto cuando inició el trabajo. Aun así, he intentado respetar la planificación inicial, teniendo algunos retrasos sobre la fecha prevista. No creo que esta planificación haya estado mal realizada, simplemente no tuve en cuenta la futura situación por razones obvias ya que no había planes por mi parte de trabajar en ese momento. En cambio, la situación lo requirió por lo que me tuve que adaptar

a ella e introducir cambios para garantizar el éxito del trabajo. Estos cambios fueron rebajar los objetivos para las fechas señaladas, o mejor dicho, cumplir los objetivos importantes una o dos semanas después de las fechas señaladas en la planificación para cumplirlos. Esto ha resultado en unas últimas semanas de mucho trabajo en el proyecto, pero el tiempo de las vacaciones de navidad han ayudado en poder sobrellevarlo. El resultado ha sido positivo ya que como he comentado antes, todos los objetivos han sido cumplidos y realizados correctamente, hasta el punto de estar orgulloso del trabajo realizado.

Para finalizar, tengo que agradecer toda la ayuda recibida por parte de amigos, pareja y familiares ya que sin su apoyo no habría sido posible realizar este proyecto. En el apartado de agradecimientos ya les he dedicado unas líneas, pero me parece importante hacerlo aquí también ya que al final puedo sacar una gran conclusión de esto: Por mucho que un trabajo sea individual, no siempre estamos solos en el camino, podemos apoyarnos sobre nuestros seres queridos. Esto no significa pedirles que nos realicen el trabajo, ni mucho menos, pero sí pedirles que estén ahí cuando tropezamos, para ayudarnos a levantarnos, o para apoyarnos mientras recorremos el camino. Este trabajo me ha servido para saber lo que es el trabajo duro, y también para saber que la industria de los videojuegos es fantástica y que sería un gran honor poder trabajar en ella y contribuir a crear lo que tantas horas de diversión nos ha brindado en nuestras vidas.

7.2. Líneas de futuro

Por último, tenemos las líneas de futuro del proyecto. Para comenzar con estas, hay ciertas cosas que me habría gustado incluir dentro de este prototipo pero por temas de disponibilidad y tiempo no me ha sido posible. Desde un inicio pensé en realizar la mayoría del arte del proyecto, creando los sprites y animaciones de estos. Es por ello que en un principio seleccioné el estilo pixelart, ya que me parecía un gran estilo artístico para comenzar a crear mis propias animaciones y sprites. Dada la situación apenas pude crear unos pocos elementos que forman el prototipo, de los cuales estoy muy orgulloso. Me habría gustado quizás añadir un segundo nivel, pero esto era más una ambición personal que un objetivo, ya que desde el principio se planteó que el prototipo constase de un nivel, y otro más estilo tutorial para enseñar a los jugadores cómo se juega realmente. Realmente no haber añadido estos elementos dentro del juego no han influenciado en el logro de los objetivos planteados, por lo que se puede quedar como una mejora a futuro, crear mis propios sprites, diseños y animaciones, y plantear nuevos niveles. Otras propuestas de futuro serían las siguientes:

- Mejorar el juego a lo que configuraciones y opciones se refiere: Añadir configuración de resolución para poder jugar a diferentes tamaños de pantalla. Mudar el juego al new input system para poder añadir opciones como una pausa del juego parando el tiempo interno del motor (sin influir en los inputs). Realizar esto con el proyecto actual es imposible ya que el modo de hacerlo sería con sentencias “If” siendo un código muy sucio y poco eficiente. Por último, traducir el juego a varios idiomas y poder jugarlo como el jugador prefiera, y dado que el juego no tiene mucho texto esto no sería realmente complicado.

- Añadir más contenido al juego: Una vez el juego tuviese una mejor base, añadirle más contenido de enemigos, biomas, quitar los niveles y plantear un escenario de mundo preestablecido, con sus zonas diferentes como un buen metroidvania. En general, esta parte es más el añadido de contenido que necesita el juego para empezar a parecer a un juego de verdad, al final se ha desarrollado solamente un prototipo muy básico, que enseña el concepto e idea del juego. Añadiendo más elementos podría plantearse un prototipo o una versión beta del juego, una demo.
- Contar con personas especialistas para la continuación del desarrollo: buscar personas que sepan de programación, diseño, marketing sería un gran punto para el futuro del juego, dando la oportunidad a un desarrollo sólido del mismo. Esto podría realizarse si se consigue una financiación para el proyecto, o se encuentra gente que confíe en el proyecto y quiera trabajar en él.
- Finalmente, la idea más ambiciosa de todas: Desarrollar el juego al completo y con intención de su lanzamiento al mercado con posibilidades de hacerse un sitio en él. Para llegar a este punto no solo haría falta cumplir todos los anteriores, sino que se requeriría de una cantidad de esfuerzo y trabajo increíble. El prototipo puede reflejar la idea y el concepto del juego, pero le falta mucho para parecerse a un metroidvania de verdad. Eso por otro lado es algo bueno, porque siempre que un proyecto tenga rango de mejora significa que este puede ser mejorado y aún le queda camino que avanzar.

Para concluir con las líneas de futuro, el trabajo tiene mucho margen de mejora. El punto positivo es que he agarrado cariño al proyecto, por lo que mi objetivo es no dejarlo de lado, sino que iré trabajando en él poco a poco para seguir sacando prototipos y propuestas y ver hasta dónde puede llegar. Sin lugar a dudas, un trabajo muy bonito y que me ha servido para aprender muchas cosas.

Bibliografía

[1] Team Cherry. Hollow Knight: Adelaide, South Australia; 2017.
[2] Nintendo. Saga The Legend Of Zelda: Japón; 1986-2023.
[3] Valve Corporation. Steam Store [Internet]. Steampowered.com [citado el 8 de noviembre de 2023]. Disponible en: https://store.steampowered.com/?l=spanish
[4] Reddit. Reddit [Internet]. [citado el 8 de noviembre de 2023] Disponible en: https://www.reddit.com/
[5] Sirradez. Survey about the demographic of Metroidvania players. [School project] [Internet]. [citado el 8 de noviembre de 2023]. Disponible en: https://www.reddit.com/r/metroidvania/comments/jj3igu/survey_about_the_demographic_of_metroidvania/
[6] Moon Studios. Ori and the Blind Forest: Viena, Austria; 2015.
[7] Moon Studios. Ori and the Will of the Wisps: Viena, Austria; 2020.
[8] Motion Twin, Playdigious, Motion Twin Scop ARL. Dead Cells; 2018.
[9] Metacritic. Metacritic - Movie Reviews, TV Reviews, Game Reviews, and Music Reviews [Internet]. Metacritic.com. [citado el 8 de noviembre de 2023]. Disponible en: https://www.metacritic.com/
[10] Kids With Sticks & 505 Games. Rogue Spirit: Milán, Italia; 2023.
[11] HAL Laboratory y Nintendo. Saga de Kirby: Japón; 1992-2023.
[12] Riot Games. League of Legends: West Los Angeles, California, Estados Unidos; 2009.
[13] Oak Woods - Brullov [Internet]. itch.io. [citado el 10 de enero de 2024]. Disponible en: https://brullov.itch.io/oak-woods
[14] Pixel Art Platformer - Village Props - Cainos [Internet]. Unity.com. [citado el 10 de enero de 2024]. Disponible en: https://assetstore.unity.com/packages/2d/environments/pixel-art-platformer-village-props-166114
[15] Fantasy enemies pack - Lilwillydesigns [Internet]. itch.io. [citado el 10 de enero de 2024]. Disponible en: https://lilwillydesigns.itch.io/fantasy-enemies-essential-pack
[16] High fantasy - Slime enemy - Warsvault. itch.io. [citado el 10 de enero de 2024]. Disponible en: https://warsvault.itch.io/high-fantasy-slime-enemy
[17] 2D pixel art Drake sprites - Elthen [Internet]. itch.io. [citado el 10 de enero de 2024]. Disponible en: https://elthen.itch.io/2d-pixel-art-drake-sprites
[18] Pixabay.com. [citado el 10 de enero de 2024]. Disponible en: https://pixabay.com/

Anexos

Entregables del proyecto:

Link al trailer del proyecto: https://www.youtube.com/watch?v=3vIR0IdFngA

Link al repositorio de gitlab con todos los archivos:

Gitlab: https://gitlab.com/gorkalabas/tmf-gorka-martinez

Release: https://gitlab.com/gorkalabas/tmf-gorka-martinez/-/releases/CodigoFuente

---

Link al ejecutable del juego del proyecto (Es la carpeta con la Build, basta con descargarla en formato .zip, descomprimir y ejecutar el ejecutable llamado ILUNA.exe):

Drive: https://drive.google.com/file/d/14LK1FMHLgxZ2YMkbgHOoUSAHuHZx2dBC/view

Gitlab:
https://gitlab.com/gorkalabas/tmf-gorka-martinez/-/tree/main/ILUNA%20Ejecutable%20juego

---

Link al repositorio de GitLab del proyecto que contiene el solamente código fuente:

Drive: https://drive.google.com/file/d/1DftkAzyBfkmMvV7fh3KcX1uROe3TAtoi/view

Gitlab:
https://gitlab.com/gorkalabas/tmf-gorka-martinez/-/tree/main/ILUNA%20Codigo%20Fuente

---

Link al video de la defensa:

https://youtu.be/By2xkQe7jks