UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Neyteria, un juego 2D de plataformas. Implementación de aspectos artísticos

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Garrido Palacios, David

Tutor/a: Abad Cerdá, Francisco José

CURSO ACADÉMICO: 2021/2022

Resumen

Para la realización de este proyecto se aplicarán principalmente los conocimientos obtenidos en la asignatura optativa de cuarto año "Desarrollo de Videojuegos 2D", y haciendo uso también de los conocimientos generales adquiridos a lo largo de la carrera de Ingeniería Informática.

Su estética es un videojuego de plataformas, acción y aventuras desarrollado para PC Pixel Art, con distintos tonos de color para cada nivel. Se trata de un juego offline y de un solo jugador. Tendrá dos modos de juego: normal y difícil. Los distintos niveles serán finitos, y su dificultad irá aumentando de forma que los enemigos tendrán más nivel y no dejarán tantas recompensas al vencerlos. Para finalizar el juego, se deberá pasar a través de distintos niveles saltando, golpeando, corriendo, esquivando y usando habilidades y artefactos, para enfrentarse a los distintos enemigos y matar al boss de cada nivel hasta pasarse el nivel final. El juego dispondrá de una variedad de más de 10 enemigos, con distintas características y habilidades.

A lo largo del juego se avanzará en la historia y se irá obteniendo distintos recolectables, como objetos activables (que afectarán a los atributos del personaje como vida, stamina, velocidad...), armas (espadas, mazos o pistolas), trajes (con ciertas características pasivas) y módulos (las habilidades principales del personaje). El personaje principal será un humano medio cyborg, modificado con tecnología avanzada, con un brazo robótico que le permite, no solo tener una fuerza sobrehumana, sino también reflejos mejorados a la hora de combatir.

La metodología de desarrollo se podría describir como una metodología ágil, que permite distribuir de forma permanente sistemas de software en funcionamiento diseñados con iteraciones rápidas. Dado a que tiene un enfoque flexible y se basa en el trabajo en equipo para ofrecer mejoras constantes, se realizará por un equipo de desarrollo reducido (2 personas) de forma que se proporcione en poco tiempo pequeñas piezas de software en funcionamiento.

Como se trata de un proyecto colaborativo, habrá que separar las tareas que deberá realizar y desarrollar cada alumno.

El alumno David Garrido se encargará principalmente de los aspectos más artísticos, sin dejar de lado su tiempo a realización de código. De una manera específica:

Se encargará de la realización del menú, el código de mecánicas básicas de combate. Elección y desarrollo de los principales enemigos. Desarrollo de la animación de estos enemigos y animaciones de las armas del protagonista. Desarrollo de los niveles y elección de sprites que concuerden con la temática del juego. Diseño del sistema de diálogo y la representación de la historia a lo largo del juego. Desarrollo artístico del inventario. Realización del sistema de sonido y selección de estos. Código de los eventos y puzzles de los niveles.

El alumno Gianfranco Pousa se encargará principalmente de los aspectos más técnicos y de código, pero también dedicará tiempo a la gestión y planificación del juego. De una manera específica:

Diseño de niveles, objetos y principales mecánicas del juego. Movimiento del personaje, así como los sistemas de vida, estamina y animaciones principales del protagonista. Código relativo a los módulos y habilidades del personaje. Desarrollo de la inteligencia de los enemigos y de la realización de los bosses. Diseño de la estética del juego y de la trama principal e historia del mismo. Planificación de las diferentes versiones y entregables, con fechas acordadas. Diseño de un sistema funcional de inventario. Gestión de carga de niveles. Mecánicas generales del juego.

Palabras clave: videojuego, Unity, desarrollo, proyecto, trabajo coordinado.

Abstract

The students will proceed to the realization of a Video Game in Unity using mainly the knowledge obtained in the Optional Subject of fourth year "Development of 2D Videogames", and also making use of the general knowledge taught throughout the career of Informatics Engineering.

The Video Game "Neyteria" will be a platform, action and adventure video game developed for PC. It will have a Pixel Art theme, with different color tones for each level. It is a game offline and single player. It will have two game modes: normal mode and difficult mode. The different levels will be finite, and its difficulty will increase so that the enemies will have a higher level and will not leave as much loot. To finish the game, you must go through different levels jumping, hitting, running, dodging and using skills and artifacts, to face the different enemies and kill the boss each level until you pass the final level. The game will have a variety of more than 10 enemies, with different characteristics and abilities.

Throughout the game, the story will progress and different collectibles will be obtained, such as activable objects (which will affect the character's attributes such as life, stamina, speed...), weapons (swords, maces or guns), suits (with certain passive characteristics), and modules (the main abilities of the character). The main character will be a half-cyborg human, modified with advanced technology, with a robotic arm that allows him to not only have superhuman strength, but also high reflexes when fighting.

The development methodology could be described as an agile methodology, which allows to distribute permanently working software systems designed with rapid iterations. Since it has a flexible approach and is based on teamwork to offer constant improvements, it will be carried out by a reduced development team (2 people) in such a way as to provide small pieces of working software in a short time.

As it is a collaborative project, it will be necessary to separate the tasks that each student must carry out and develop.

The student David Garrido will be in charge mainly of the most artistic aspects, without neglecting his time for code development. In a specific way:

He will be in charge of making the menu, the code of basic combat mechanics. Choice and development of the main enemies. Development of the animation of these enemies and animations of the protagonist's weapons. Development of the levels and choice of sprites that match the theme of the game. Dialogue system design and story representation throughout the game. Artistic development of the inventory. Realization of the sound system and selection of these. Code of the events and puzzles of the levels.

The student Gianfranco Pousa will be in charge mainly of the most technical and code aspects, but he will also dedicate time to the management and planning of the game. In a specific way:

Level design, objects and main game mechanics. Movement of the character, as well as the systems of life, stamina and main animations of the protagonist. Code related to the modules and abilities of the character. Development of the intelligence of the enemies and the realization of the bosses. Design of the aesthetics of the game and the main plot and history of the game. Planning of the different versions and deliverables, with agreed dates. Design of a functional inventory system. Level load management. General game mechanics.

Keywords : videogame unity, Development, project, coordinated work.

Índice General

Índice de Figuras

Índice de tablas

Índice de Listas

CAPÍTULO 1

Introducción

A lo largo de la historia, los juegos han servido a la humanidad y para ejercitar o pulir nuestros sentidos y capacidades competitivas, sociales e incluso de supervivencia, a la vez que nos ofrecían ocio y entretenimiento.

Los juegos (y videojuegos) son, además, herramientas muy útiles a la hora de aplicar pedagogía y de enseñar ideas y valores de una manera sencilla, eficaz y divertida. [1] A través de la dimensión *playworld* (mundo del juego), las mecánicas del juego y la *playperformance* (dinámica del juego), junto a los distintos planos podemos observar cómo penetra las ideas y la cultura, a la vez que emparenta estos juegos con otras disciplinas como la lectura, la literatura, el cine, la música, la escultura…

1.1 Motivación

En la actualidad, el desarrollo de la tecnología y la informática nos ha permitido crear tipos de ocios derivados, uno de los cuales hemos apodado como *videojuegos*. Esta digitalización ha permitido el acceso a este entretenimiento desde casi cualquier parte del mundo, y ha propiciado, por tanto, una mejora en la creación de estos, tanto en el diseño, como en el desarrollo y como en la implementación de los videojuegos.

Los videojuegos fomentan la experiencia y el aprendizaje constructivista [2], que busca la construcción del conocimiento por el usuario y demuestran el potencial de los videojuegos como una herramienta para la educación y formación de la persona.

Este sector de los videojuegos, hoy en día, se podría considerar como una de las principales fuentes de entretenimiento de las últimas generaciones. Esta fama ha desembocado en un desarrollo de videojuegos más complejos e impresionantes, haciendo que la creación de estos sea un gran reto. Ahora mismo, la creación de videojuegos de alta calidad se basa en enormes proyectos profesionales de software con distintos departamentos, equipos, y un gran número de personas cooperando.

Por ello, nuestra principal motivación para la realización de este proyecto de fin de grado se basa en el reto del desarrollo de un videojuego complejo, de forma que produzcamos entretenimiento y diversión a la vez que ayudamos al crecimiento y enriquecimiento personal de ideas y valores, obtenidos de una manera implícita y sencilla por los usuarios, y el desarrollo de sus capacidades motoras y competitivas, a la vez que nosotros mejoramos y evolucionamos nuestras habilidades e imaginación en el desarrollo de este proyecto.

La memoria de este proyecto contiene cierta terminología específica, que se podrá encontrar en el Glosario, también definido como Anexo 4.

1.2 Objetivos

Este Trabajo de fin de Grado tiene como finalidad el desarrollo de un videojuego indie [3] funcional y original de plataformas, acción y aventura que muestre nuestras habilidades y capacidades obtenidas a lo largo del estudio del grado de ingeniería informática.

Este videojuego se realizará en colaboración con otro alumno, será offline, de un solo jugador y estará diseñado para PC. A lo largo del juego se avanzará en la historia y se irá obteniendo distintos recolectables, que mejorarán la experiencia del jugador y le propiciarán una mejor inmersión.

El juego dispondrá de distintos niveles o zonas que el jugador irá explorando y recorriendo, a la vez que deberá resolver pequeños puzles en cada nivel para poder avanzar o conseguir ventajas que le ayudarán en su continuación del juego.

El videojuego poseerá una gran variedad de enemigos, con distintas mecánicas y habilidades, entre los cuales podemos distinguir a los bosses, que serán los guardianes a los que el jugador deberá vencer para cambiar de nivel de juego.

Se cuidará la parte artística del videojuego, así como una correcta realización de las interfaces y menús. El juego será de estilo pixel art y se tendrá cuidado en la selección de los tilesets y sprites a la vez que se buscará una correcta combinación de temáticas y colores. También se cuidará la parte musical del juego.

Los sprites son un conjunto de imágenes que representan a un personaje o a un objeto (o una parte de ellos) de manera gráfica y que se utiliza para poder crear cualquier efecto de movimiento o para cambiar su estado o posición en la escena. Un tileset es un conjunto de texturas (o tiles) reunidas en una misma imagen, una composición. Estas texturas forman las piezas gráficas que componen el escenario de un videojuego 2D: suelos, paredes, escaleras, techos, fondos, estructuras, etc.

A continuación, se enumerará un resumen de los objetivos a tener en cuenta en este TFG:

- Se encargará de la realización del menú principal, menú de pausa y pantalla de muerte
- Desarrollo del código de mecánicas básicas de combate y detalles del movimiento del personaje.
- Animación básica de las armas del protagonista.
- Diseño y elección de los enemigos del juego.
- Desarrollo de los enemigos del juego.
- Diseño y desarrollo de la animación de los enemigos.
- Diseño y desarrollo de los niveles del juego.
- Desarrollo e implementación del sistema de diálogo.
- Desarrollo de parte de la historia del juego.
- Representación de la historia a lo largo del juego.
- Desarrollo artístico del HUD del jugador.
- Diseño e implementación del sistema de sonido y selección de estos.

- Realización de elementos y eventos para puzles en los niveles del juego.
- Selección de los principales sprites, colores y fuentes del juego acordes a la temática de este.

1.3 Impacto esperado

El impacto esperado del videojuego es el siguiente:

- Como es un videojuego indie, no se espera una participación de un gran alcance por parte de los jugadores, ya que el mercado es inmenso y cada vez se crean mejores juegos, más complejos y más entretenidos, con lo cual se crea cada vez más competencia.
- Sin embargo, esto no implica que no busquemos conseguir un número decente de jugadores, dado que el juego que hemos desarrollado perfila y mezcla técnicas y mecánicas para obtener un juego original que sobresalga sobre la media.
- Mediante publicidad en redes sociales y la publicación de este en plataformas preparadas para publicitar juegos indie, también buscaremos aumentar el impacto sobre el público y abarcar de este modo más mercado de jugadores.

Se espera que el videojuego atraiga a clientes de las zonas de habla hispana, aunque no se descartan potenciales jugadores de otras regiones y otros idiomas.

No se descarta, en un futuro, crear traducciones, ya sean amateur [4] o profesional [5], a otros idiomas populares como inglés, alemán y chino para conseguir un mayor alcance de mercado.

1.4 Metodología

Este proyecto fue iniciado en la asignatura optativa *Desarrollo de Videojuegos 2D* del grado de Ingeniería Informática y ha sido continuado para la realización del Trabajo de Fin de Grado.

Como se ha estudiado en las asignaturas de software de la carrera, las metodologías tradicionales establecen que los proyectos se ejecutan en un ciclo secuencial. Sigue una secuencia fija de Iniciación, Planificación, Ejecución y Medición. Además, este enfoque tradicional, pone especial énfasis en los procesos lineales, la documentación, la planificación por adelantado y la priorización. Con el método tradicional, el tiempo y el presupuesto son variables y los requerimientos (aquello que tenemos que lograr) son fijos, debido a esto a menudo existen problemas de presupuesto y plazos. Para cada paso hay herramientas y técnicas definidas por el estándar que marca la metodología PMBOK [8].

Este tipo de metodología tiene grandes beneficios, como objetivos claramente definidos, procesos controlables, documentación clara, mayor responsabilidad… Sin embargo, la rigidez y

claridad que necesita el sistema puede suponer un problema cuando estamos tratando con entornos no claros ni estables, en los cuales los requisitos están cambiando frecuentemente.

Por otro lado, tenemos las metodologías ágiles, que se basan en gran medida en el trabajo en equipo, la colaboración, las tareas y la flexibilidad para responder al cambio lo más rápido posible. Este tipo de metodologías tienen características tentadoras como un mayor enfoque en individuos e interacciones que procesos y herramientas. Se entiende que en una metodología ágil el software funcionando es más importante que una documentación extensa y la colaboración con el cliente es más importante que la negociación contractual. Por lo tanto, es mucho mejor para responder al cambio en lugar de seguir ciegamente un plan.

Gracias a este tipo de metodología ágil se tiene que las prioridades establecidas pueden ser flexibles, que se empieza a entregar antes, mayor conocimiento sobre costes y plazos. Por lo tanto, esto desemboca en una mejora de la calidad final y en una mayor transparencia en el proyecto.

La metodología utilizada en el desarrollo del proyecto es una metodología de tipo ágil, en la cual a lo largo del desarrollo del juego se obtuvieron sistemáticamente pequeños entregables.

Utilizamos una metodología basada en SCRUM [6], en la cual se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

Realizábamos entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, una metodología basada en Scrum está especialmente indicada para estos proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos dado que el GDD es un documento variable, y donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

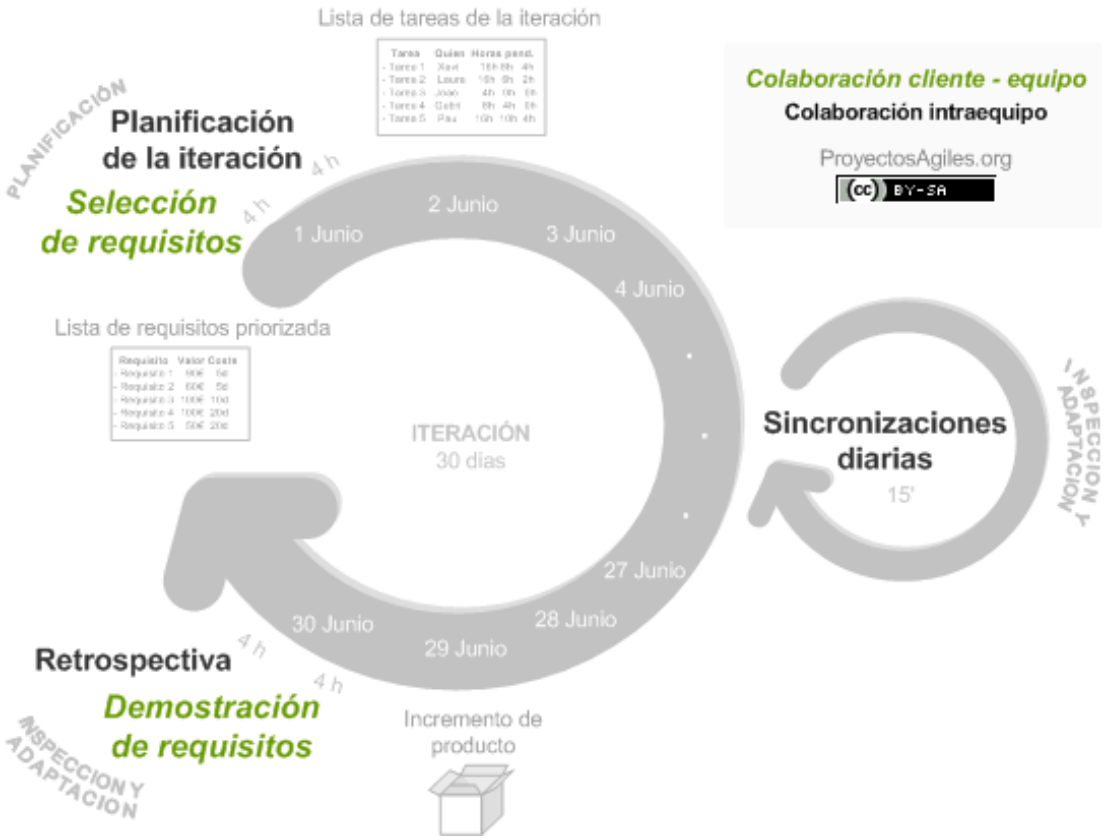


Figura 1. Proceso de Scrum (imagen obtenida de la página proyecto agiles: https://proyectosagiles.org/que-es-scrum/)

De manera específica en nuestro proyecto podemos ver dos partes:

Parte 1 (asignatura optativa)

- La metodología de desarrollo utilizada durante la asignatura optativa se basa en reuniones.
- En las clases nos encontrábamos ambos compañeros y hablábamos sobre los avances y actualizaciones que realizábamos en el juego, junto a unas cuantas reuniones por Discord varios días sueltos semanalmente.
- Al final de algunas clases también hablábamos con el profesor para contarle los detalles y avances de nuestro proyecto.
- Además, la propia asignatura disponía de una serie de entregables que ayudaban a la correcta fluidez y desarrollo del proyecto.
- En cada iteración se proporcionaba un resultado completo, un incremento del producto final preparado para su visualización por parte del profesor, ya fuese un acto evaluable de la asignatura o no.
- Coordinación del código y del proyecto mediante un sistema de administración de versiones online cooperativo como es GitHub [7].

Parte 2 (después de la asignatura)

- Al acabar la asignatura, los compañeros seguíamos en contacto mediante Discord, con reuniones frecuentes casi diarias.

- También contactábamos con el profesor cada dos semanas para contarle las mejoras y actualizaciones en el proyecto de software, y donde planteábamos los requisitos y funcionalidades a realizar en la iteración correspondiente
- Aún así, en esta fase del proyecto la coordinación de tareas era más compleja, por lo que empezamos a usar la aplicación HacknPlan recomendada por el profesor.
- Probamos con algunas otras, como Trello y Monday, pero la completitud y facilidad para proyectos de software que ofrecía HacknPlan nos hizo decidirnos por esta.
- Al igual que en la asignatura, en cada iteración se proporcionaba el incremento del producto final preparado para su visualización por parte del profesor.
- Se continuó con el uso de GitHub como gestor de versiones.

1.5 Estructura de la memoria

Esta memoria está estructurada en varias partes, donde se documentará el proceso de desarrollo del proyecto. Estas partes se resumirán a continuación:

- **Capítulo 1. Introducción:** Describe de forma general los objetivos y motivación en los cuales se basa el proyecto, así como la metodología y colaboraciones realizadas en el mismo.
- **Capítulo 2. Estado del arte:** También llamado Estudio Estratégico, este capítulo se basa en el estudio y comparación de soluciones similares y de la competencia actual, que nos ayudarán a identificar y extraer requisitos de una forma sencilla.
- **Capítulo 3. Análisis del problema:** tal como indica el nombre del capítulo, se incidirá en las necesidades y requisitos necesarios para la realización del proyecto, así como aquellos que puedan ser negativos para el mismo, y el plan de trabajo a desarrollar.
- **Capítulo 4. Diseño de la solución:** en este capítulo se hablará sobre las decisiones tomadas acerca de cómo se ha de llevar a cabo la solución del problema, destacando por qué se descartaron las opciones no seleccionadas. También se hablará de la tecnología utilizada para la solución.
- **Capítulo 5. Desarrollo de la solución propuesta:** en este capítulo se describirá el desarrollo del proyecto. Se incluirán los problemas y decisiones tomadas para transformar la propuesta en la solución final.
- **Capítulo 6. Implementación:** en este capítulo se indica cómo se instala y se pone en marcha el videojuego para probarlo y obtener resultados.
- **Capítulo 7. Pruebas:** en este capítulo se describen las pruebas realizadas para cerciorarnos de que la solución funciona correctamente y está listo para salir a mercado.
- **Capítulo 8. Conclusiones:** en este capítulo encontraremos una reflexión sobre los resultados y conclusiones comparado con los objetivos presentados anteriormente.
- **Capítulo 9. Trabajos futuros:** Describe una serie de ampliaciones, mejoras y detalles que no han podido ser realizados dada la limitación de tiempo de este TFG, y cuáles serían interesantes añadir en un futuro.

- **Capítulo 10. Referencias:** aquí encontraremos las referencias y bibliografías usadas en el desarrollo del TFG y de la memoria.

1.6 Colaboraciones

Este Trabajo de Fin de Grado trata sobre el desarrollo de un videojuego de forma cooperativa, como antes se indicó en el resumen, objetivos y metodología. Por lo tanto, debemos dejar claro cuál es el equipo de trabajo del proyecto y qué parte del proyecto ha realizado cada uno. De aquí en adelante, las partes del proyecto que no sean realizadas en este TFG serán explicadas brevemente siempre que sean interesantes y útiles para el desarrollo del mismo.

Este proyecto se realiza en una colaboración de dos alumnos de cuarto curso del grado de ingeniería informática: David Garrido Palacios y Gianfranco Pousa Barros.

De una forma general, se puede separar el trabajo realizado por cada alumno refiriéndonos a la parte más artística y de diseño al alumno David, y la parte más de código y gestión a Gianfranco.

No cabe olvidar, que este es un proyecto de ingeniería informática, en el que se deberá realizar software funcional y útil para los objetivos requeridos. Por lo tanto, aunque los trabajos estén separados de esta manera tan general, ambos deberán demostrar sus aptitudes en el desarrollo de software y código con la competencia de un alumno que va a graduarse de la carrera de ingeniería informática.

De manera más específica podemos enumerar las tareas realizadas por cada alumno.

David deberá encargarse de los siguientes aspectos:

- Se encargará de la realización del menú principal, menú de pausa y pantalla de muerte
- Desarrollo del código de mecánicas básicas de combate y detalles del movimiento del personaje.
- Animación básica de las armas del protagonista.
- Diseño y elección de los enemigos del juego.
- Desarrollo de los enemigos del juego.
- Diseño y desarrollo de la animación de los enemigos.
- Diseño y desarrollo de los niveles del juego.
- Desarrollo e implementación del sistema de diálogo.
- Desarrollo de parte de la historia del juego.
- Representación de la historia a lo largo del juego.
- Desarrollo artístico del HUD del jugador.
- Diseño e implementación del sistema de sonido y selección de estos.
- Realización de elementos y eventos para puzles en los niveles del juego.
- Selección de los principales sprites colores y fuentes del juego acordes a la temática.

Gianfranco deberá encargarse de los siguientes aspectos:

- Diseño y desarrollo de los objetos y armas del juego.
- Diseño e implementación de las principales mecánicas del juego.
- Diseño e implementación del movimiento del personaje.
- Desarrollo de los sistemas de vida y estamina del protagonista.
- Animaciones principales del protagonista.
- Realización del código de las habilidades del personaje.
- Desarrollo del sistema de módulos.
- Diseño de la estética principal del juego.
- Diseño de la trama principal e historia del juego
- Planificación de las distintas versiones y entregables de este, con fechas acordadas.
- Diseño y desarrollo del sistema de inventario.
- Diseño del sistema de tienda del juego.
- Diseño e implementación del sistema de estadísticas del protagonista.
- Diseño e implementación del sistema de niveles de los enemigos.
- Gestión de carga de niveles para optimización del juego.
- Diseño y desarrollo del sistema y movimiento de la cámara dentro del juego.

CAPÍTULO 2

Estado del arte

2.1 Crítica al estado del arte

A lo largo de la historia de los videojuegos, se observa que el gran avance permite tener una mejor resolución a la hora de desarrollar videojuegos, y por ello se vuelven de mejor calidad y más complicados. Desde “Tennis for Two” y “Space War” [9] llegamos a las máquinas recreativas, y un tiempo después vendría la industria de las consolas para el mercado doméstico. En poco tiempo se empezaron a desarrollar los videojuegos en 3D y rápidamente se empezó a comercializar las consolas portátiles. Unas cuantas consolas más de las marcas Nintendo, Sony y Xbox dejaron abierta la puerta a lo que conocemos ahora como el estado del arte de los videojuegos.

En PC este desarrollo fue más específico, ya que se centró más en los juegos de tipo FPS, RTS, y los MMORPG. Aunque últimamente, gracias a las nuevas tecnologías como la VR y a la gran potencia conseguida, los videojuegos en PC son actualmente los más complejos y pesados del mercado.

Volviendo a nuestro juego, queremos analizar la competencia que podrá tener nuestro proyecto. Para ello primero se definirá sus características principales:

- Videojuego en 2D
- Offline
- Un jugador
- Varios modos de dificultad
- Plataformas, acción y aventura
- PC

Como podemos observar, las características más importantes de nuestro proyecto es el género principal de plataformas, el tipo de videojuego en 2D y el modo offline con un jugador.

Pensando en estas características, no es complicado encontrar juegos de cultura que cumplan estas características:

Super Mario Bros

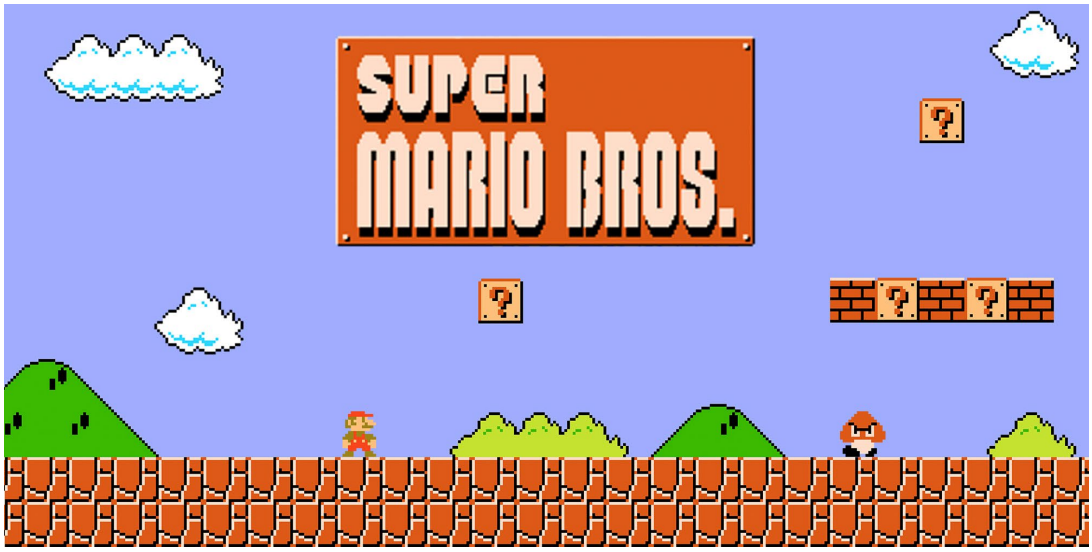


Figura 2. Super Mario Bros.

Como podemos observar, el Super Mario Bros es una importante referencia para nuestro videojuego, y por lo tanto deberemos obtener información útil analizando sus puntos fuertes y qué lo hace tan entretenido y adictivo. En la figura 2 vemos que estas características consistían en su sencillez y su género. Esta sencillez es algo que se ha buscado plasmar en el videojuego Neyteria como característica principal.

Hollow knight y Ori and the Blind Forest

Figura 3. Hollow Knight

Figura 4. Ori and the Blind Forest

Ambos videojuegos son de los más populares en este momento y están disponible en una gran cantidad de plataformas. Como se observan en las figuras 3 y 4, el secreto de su popularidad reside en sus hermosos dibujos y trazados, así como en su género *Metroidvania*[10]. Este género es un acción-aventura basado en plataformas, por lo cual es muy parecido al de nuestro juego.

Gris

Figura 5. Gris

Este juego de aventuras y plataformas está dando mucha guerra en el mercado gracias a sus dibujos y sus divertidas y simples mecánicas. El punto más destacable del juego es cómo se va actualizando la estética y las mecánicas conforme se avanza en el videojuego. En la figura 5 se puede observar un ejemplo del diseño artístico utilizado. Hemos utilizado la idea del cambio de estética conforme avanza el juego y la historia, para que sea más inmersivo y coherente.

Dark Souls

Figura 6. Dark Souls

Las mecánicas y dificultad lo han llevado a la fama a esta serie de videojuegos de Dark Souls. Nuestro juego ha bebido estas mecánicas para darle un toque más complejo y dinámico sin perder la simplicidad que debe tener un juego plataformas. Un ejemplo sería el uso de la stamina en el sistema de combate, la posibilidad de realizar habilidades como la evasión y el uso de objetos rápidos mediante determinadas teclas.

Comparativa con nuestro juego

Nombre	Género Plataformas	Mecánicas Elaboradas	Simpleza	Dificultad	Actualización estética	Disponible en PC
Neyteria	X	X	X	X	X	X
Mario Bros	X		X			X
Hollow Knight y	X	X			X	X

Ori						
Gris	X		X		X	X
Dark Souls		X		X		X

Tabla 1. Comparativa de la competencia con nuestro juego

2.2 Análisis DAFO del proyecto

El análisis DAFO [15] es una sencilla herramienta de análisis estratégico muy extendida en la toma de decisiones de todo tipo de organizaciones. Ayuda a analizar y poner de relieve la situación actual de una organización o proyecto, y así poder tomar las decisiones estratégicas adecuadas. Es una herramienta habitual tanto en un plan estratégico como en un plan de negocio o un estudio de mercado.

Mediante un análisis del entorno externo y las características internas de la organización, esta herramienta de gestión empresarial permite obtener una representación gráfica de las Debilidades, Amenazas, Oportunidades y Fortalezas del proyecto.

Figura 7. Análisis DAFO del proyecto

2.3 Propuesta

Lo que hemos observado en el mercado es una gran cantidad de juegos de alta calidad visual y mecánica, pero que dejan un poco de lado detalles importantes que hemos sabido aprovechar en nuestro juego. La mezcla de unas mecánicas elaboradas junto a cierta simpleza y dificultad es lo que hace a nuestro juego destacar sobre la competencia en el mercado.

Por lo tanto, nuestra propuesta de videojuego intenta unir y aplicar una mezcla de características que tienen los juegos populares actualmente, de manera que se obtenga lo mejor de cada juego y estilo, para crear un videojuego divertido, entretenido y adictivo.

A continuación, se presentará las características (actualizadas con el mercado actual) más importantes que tiene nuestro videojuego:

- **Videojuego en 2D**

  Como ya hemos comentado anteriormente, nuestro juego se realizará en 2D, dada la cantidad de sprites en 2D gratuitos disponibles en internet, las técnicas que hemos aprendido y desarrollado en Unity, y a que es el estilo de juego que estamos buscando para poder penetrar y destacar en el mercado.

- **Offline**

  El juego será offline. Al ser un juego donde hay que seguir la historia y explorar y descubrir los niveles, tiene sentido que el juego no sea on-line.

- **Un jugador**

  Hemos decidido hacer el juego de un solo jugador, para dar una experiencia más clásica e inmersiva a los usuarios.

- **Varios modos de dificultad**

  Esto permitirá establecer la dificultad y dinámica buscada para los jugadores más avanzados, sin dejar de lado a los nuevos usuarios que quieran adentrarse en el mundo de los videojuegos de géneros plataforma o *Metroidvania*.

- **Plataformas, acción y aventura**

  Estos géneros son los más populares en el mercado de juegos 2D offline, por lo tanto, tiene sentido explotarlos y mezclarlos para obtener el resultado óptimo buscado.

- **PC**

  El videojuego será creado para ordenador principalmente por sencillez y limitación de tiempo. En un futuro se podría actualizar y mejorar para plataformas y dispositivos, abarcando así mayor mercado y mejorando nuestra competitividad.

- **Simple**

  Tal como hemos comentado en la crítica al estado del arte, la sencillez de un videojuego puede ser una característica muy importante en los juegos de géneros plataformas.

- **Mecánicas elaboradas**

  Aunque esta sencillez nunca debería permitir que el juego se volviera aburrido y monótono debido a unas pobres mecánicas. Encontrar el punto medio entre estas dos características será clave para el éxito del videojuego.

- **Actualización estética**

  Los videojuegos son obras artísticas, visuales y musicales. Por ello, se debe explotar en tanto sea posible estas áreas. Con el cambio de estética conforme avanza el juego y la historia, podemos hacer que sea más intuitivo y coherente, lo que ayudará a una mejor inmersión del jugador.

CAPÍTULO 3

Análisis del problema

El proyecto a realizar es un videojuego, pero al fin y al cabo sigue siendo un proyecto software y debemos tratarlo como tal. Por ello, será necesario pasar por una fase de análisis de requisitos, para obtener una especificación de los requisitos software, ya sean funcionales [11] o no funcionales [12].

Aquí deberemos identificar, definir y especificar los requisitos del proyecto. Estos requisites deberán definirse correctamente, ya que el producto final deberá cumplirlos y marcarán las pautas de desarrollo del proyecto.

3.1 Análisis del marco legal y ético

El marco legal es de obligado cumplimiento por todo profesional, y debe ser considerado desde diversos aspectos como algo prioritario. A continuación hablaremos sobre algunos aspectos que afectan al juego.

Análisis de la protección de datos

El videojuego hace uso y guarda datos del usuario como su preferencia de volumen y brillo. En un futuro queremos tener un sistema de guardado más complejo que guarde el estado actual del personaje y la partida, al igual que hacer que este guardado sea por cifrado binario, evitando que los datos pudiesen ser modificados de una forma directa para obtener alguna ventaja.

Propiedad intellectual

Los sprites, tiles y sonidos usados en el desarrollo del proyecto han sido obtenidos de plataformas legales, y obtenidos de forma gratis y legal. Al final del documento hay un anexo (anexo 3) donde se enumerará la referencia de los materiales obtenidos que soliciten una mención para su uso.

3.2 Elicitación de requisitos

Para obtener los requisitos deberemos pasar por la fase de análisis o elicitación de requisitos. Esta actividad se basa en la comunicación de los desarrolladores con los stakeholders [13].

En un proyecto como este, generalmente, existe una diferenciación entre desarrollador y clientes, los cuales se comunican mediante distintos métodos para crear el software de acuerdo a los requisitos y estándares del cliente. Sin embargo, en nuestro caso que es un videojuego indie,

los desarrolladores ocupan también el rol de cliente, lo que facilita esta comunicación y definición de los requisitos del proyecto.

Se dispone de variadas técnicas para entender e identificar los requisitos necesarios que debe tener el proyecto para cumplir los objetivos. En nuestro caso usaremos tres tipos de técnicas debido a su eficacia y simpleza: brainstorming [14], entrevistas y reuniones.

BrainStorming

Mediante esta técnica recopilaremos información de los stakeholders. Mediante esta técnica generaremos una gran cantidad de ideas, lo cual viene muy bien para empezar. Además, mejorará la creatividad y nos aportará soluciones innovadoras que podremos usar en las siguientes fases de entrevistas y reuniones.

Para llevar a cabo este proceso, primero deberemos realizar el paso 1, la preparación de la tormenta de ideas. Esta preparación consiste en usar cuatro reglas básicas:

- No realizar valoraciones ni críticas sobre las ideas.
- Pensar libremente, aunque la idea no sea posible o viable.
- Intentar conseguir una gran cantidad de ideas.
- Ayudar o complementar las ideas de los otros aumenta el número de ideas obtenidas al final.

Con estas reglas básicas pasamos a la fase de desarrollo o generación de ideas, donde se define el problema y se buscan soluciones. Finalmente, se recogen estas ideas y se realiza un proceso de selección.

Este proceso fue realizado en una reunión presencial en la universidad, con un tiempo límite que marcaba el fin de la generación de ideas. Se decidió que se iba a realizar un videojuego en 2D, y al partir de ahí se obtuvo una lista de soluciones e ideas. De este resultado, mediante un descarte obtuvimos información útil que estableció una dirección en el proyecto en el pitch doc y en el GDD, dando un empujón temprano al proyecto.

A continuación, se mostrará una figura con las ideas obtenidas en el brainstorming:

Figura 8. Ideas obtenidas del Brainstorming. Las azules pasaron el proceso de selección.

Esta información se convirtió en las características principales nombradas anteriormente en el estado del arte. Además, obtuvimos información del personaje principal protagonista, del diseño de los niveles y de los enemigos, así como la estética y temática del videojuego.

Entrevistas y reuniones

Al ser nosotros mismos los desarrolladores y los clientes, las funciones de entrevistas y reuniones se solapaban bastante, de tal forma que una reunión se convertía en una entrevista y viceversa. Estos procesos fueron realizados en las distintas clases y reuniones on-line realizadas a lo largo de la asignatura.

Aparte de los desarrolladores, el punto de vista del profesor, más externo y experimentado, nos permitía centrarnos en detalles que se habían dejado poco perfilados o eran demasiado detallados.

También se realizaron una serie de entrevistas con personas externas, como familiares amigos, que aportaban ideas, consejos o dudas sobre las ideas que se tenían sobre el videojuego en el momento, y a lo largo de su desarrollo.

3.3 Definición de requisitos

A continuación, usando los datos obtenidos en la elicitación de requisitos, se listarán una serie de requisitos funcionales y no funcionales necesarios para el proyecto realizado. Los listaremos de manera que se distinga entre requisitos funcionales (servicios que prestará el sistema) y requisitos no funcionales (propiedades del sistema).

Requisitos Funcionales del Sistema

Requisito nº	Referente a	Descripción
F1	Elección modo de Juego. Menú principal	El juego permitirá al jugador elegir entre dos modos de juego, modo normal y modo difícil.

Tabla 2. Requisito F1

Requisito nº	Referente a	Descripción
F2	Menú principal	El juego permitirá al jugador el inicio de la partida desde el menú de inicio

Tabla 3. Requisito F2

Requisito nº	Referente a	Descripción
F3	Menú principal	El juego permitirá al jugador cambiar las opciones desde el menú de inicio

Tabla 4. Requisito F3

Requisito nº	Referente a	Descripción
F4	Menú principal	El juego permitirá al jugador a acceder a los créditos desde el menú de inicio

Tabla 5. Requisito F4

Requisito nº	Referente a	Descripción
F5	Menú principal	El juego permitirá al jugador salir del juego desde el menú de inicio

Tabla 6. Requisito F5

Requisito nº	Referente a	Descripción
F6	Menú principal. Submenús del Menú principal	El juego permitirá volver al menú principal desde cualquier submenú

Tabla 7. Requisito F6

Requisito nº	Referente a	Descripción
F7	Menú de pausa	El juego permitirá al jugador pausar el juego y acceder al menú de pausa en cualquier momento, siempre que haya iniciado la partida

Tabla 8. Requisito F7

Requisito nº	Referente a	Descripción
F8	Movimiento básico del personaje	El juego permitirá al jugador mover el personaje, siempre que haya iniciado la partida

Tabla 9. Requisito F8

Requisito nº	Referente a	Descripción
F9	Mecánicas básicas del personaje. Realización de elementos y eventos para puzles en los niveles del juego.	El juego permitirá al jugador activar los elementos necesarios del nivel, pulsando la tecla correspondiente

Tabla 10. Requisito F9

Requisito nº	Referente a	Descripción
F10	Mecánicas de enemigos y del personaje	El juego permitirá que el personaje atraviese los enemigos y no reciba daño, tras haber sufrido daño por contacto con ellos

Tabla 11. Requisito F10

Requisito nº	Referente a	Descripción
F11	Realización de elementos y eventos para puzles en los niveles del juego.	El juego permitirá al protagonista romper determinadas paredes seleccionadas

Tabla 12. Requisito F11

Requisito nº	Referente a	Descripción
F12	Realización de elementos y eventos para puzles en los niveles del juego.	El juego permitirá al protagonista a desplazarse correctamente por las plataformas móviles

Tabla 13. Requisito F12

Requisito nº	Referente a	Descripción
F13	Desarrollo e implementación del sistema de diálogo.	El juego permitirá al protagonista activar elementos de diálogo

Tabla 14. Requisito F13

Requisito nº	Referente a	Descripción
F14	Mecánicas básicas de combate	El juego permitirá al protagonista a realizar uso de los ataques, dash y módulos a su disposición pulsando la tecla o botón correspondiente

Tabla 15. Requisito F14

Requisito nº	Referente a	Descripción
F15	Mecánicas de los enemigos y de combate del personaje	El juego permitirá al protagonista realizar daño a los enemigos mediante su sistema de combate

Tabla 16. Requisito F15

Requisito nº	Referente a	Descripción
F16	Menú principal. Diseño e implementación del sistema de sonido.	El juego deberá enviar feedback visual y sonoro de las acciones en el menú principal

Tabla 17. Requisito F16

Requisito nº	Referente a	Descripción
F17	Mecánicas de combate y movimiento básico del personaje. Animación básica de las armas del protagonista. Diseño e implementación del sistema de sonido.	El juego deberá enviar feedback visual y sonoro de las acciones de movimiento y combate del protagonista, siempre que haya iniciado la partida

Tabla 18. Requisito F17

Requisito nº	Referente a	Descripción
F18	Mecánica de enemigos	El juego deberá enviar feedback visual del estado de vida de los enemigos del juego

Tabla 19. Requisito F18

Requisitos No Funcionales del Sistema

Requisito nº	Referente a	Descripción
NF1	Sistema de intefaz y HUD	El juego se deberá adaptar a distintas resoluciones y tamaños de pantalla.

Tabla 20. Requisito NF1

Requisito nº	Referente a	Descripción
NF2	Mecánicas generales del juego	El juego deberá funcionar en PC con el sistema debidamente instalado

Tabla 21. Requisito NF2

Requisito nº	Referente a	Descripción
NF3	Propiedad intelectual	El videojuego no guardará ningún dato personal de relevancia del usuario

Tabla 22. Requisito NF3

Requisito nº	Referente a	Descripción
NF4	Mecánicas generales del juego	No deberá haber pantallas de carga entre niveles

Tabla 23. Requisito NF4

Requisito nº	Referente a	Descripción
NF5	Mecánicas generales del juego. Menú principal.	La duración de carga de inicio del juego desde el menú principal no deberá pasar de 5 segundos

Tabla 24. Requisito NF5

3.4 Identificación y análisis de soluciones posibles

Dado los requisitos obtenidos en los apartados anteriores, se nos presentan varias opciones a la hora de crear una solución a nuestro proyecto.

Sistema de niveles

Una solución sería dividir el mapa del juego entero en varios niveles. En específico 4 niveles, donde cada uno tenga una temática distinta siguiendo la historia.

Otra solución al sistema de niveles sería hacer un único nivel, y cambiar la temática cuando sea necesario para cuadrar con la estética y la historia del juego.

Incluso se podrían hacer más o menos divisiones de niveles, habiendo varios niveles con la misma temática.

La solución de usar un único nivel parece la menos viable, ya que compartimentar el trabajo ayuda a una mejor fluidez. Por otro lado, la solución de hacer muchos niveles puede ser contraproducente, pudiendo perder el sentido y la cohesión global de la estética y la historia del juego.

Enemigos del juego

Una solución sería dividir el diseño de enemigos por cada nivel del juego. De tal manera que se realizarán enemigos de misma temática más fácilmente mientras nos quedemos en un mismo nivel. De tal manera se realizarían los enemigos necesarios para cada nivel a la vez que diseñamos el nivel.

Otra solución para el diseño de enemigos sería hacer una lista de enemigos global, y empezar a hacer uno detrás de otro. Después, se acabarían repartiendo en los distintos niveles de forma equitativa. Esta solución puede ser peligrosa porque se puede perder la cohesión y el sentido de la temática en el desarrollo de los distintos enemigos a lo largo del videojuego.

Incluso se podrían hacer enemigos a la carta, de manera que los desarrollaríamos cuando los fuéramos necesitando, por lo tanto, no se produciría un problema de cohesión ni de temática, haciendo que el desarrollo de niveles esté muy separado del de los enemigos. Esta independencia podría no ser lo que buscamos a la hora de aplicar la cohesión y el correcto avance de la historia.

Menús del juego

Una solución sería dividir cada menú por una serie de submenús y prefabs, de manera que un menú se realice de forma modular y repetitiva. De esta manera, cuando se consiga hacer un menú o submenú, simplemente será replicar lo que ya está realizado y modificar los pocos elementos necesarios.

Otra solución sería realizar un menú desde cero con todos los elementos y enlaces que tiene, y lo que eso conlleva. Esta opción puede ser más cómoda al principio, pero cuando el número de menús empieza a aumentar o se vuelven más complicados, los enlaces y saltos se convierten en una serie de fallos y bugs que son muy complicados de solucionar a posteriori.

Sistema de diálogo del juego

Una solución sería crear un elemento que pueda ser duplicable, y de tal manera ser replicado por todo el juego, contando la historia y avances. Es un método sencillo y eficaz, pero no parece ser muy eficiente a la larga ni óptimo para conseguir unos correctos fps en el juego.

Otra solución sería diseñar un sistema de diálogo no duplicable, y que no se eliminase cada vez que es usado. Simplemente se escondería y se iría reutilizando en cada momento que fuese necesario a lo largo de los niveles.

Historia del juego y su representación

Una solución sería crear la historia del juego de forma progresiva, según se vaya avanzando en el desarrollo del mismo. Esta opción no tiene por qué ser negativa, pero el hecho de no tener la historia completa no permite tener una tierra firme en la que apoyarse cuando surgen dudas sobre el desarrollo.

Otra solución sería crear la historia del juego de golpe y con anterioridad al desarrollo del videojuego y sus niveles. De esta manera, la historia puede guiar al desarrollador y al diseñador a lo largo del proyecto. Sobre todo, ayuda a la estética y cohesión del juego.

La historia se puede representar mediante el uso del sistema de diálogo que tenemos implementado en el juego.

Elementos y eventos para puzles del juego

Una solución sería dividir el diseño de elementos por cada nivel del juego. De tal manera que se realizarán elementos y eventos de misma temática más fácilmente mientras nos quedemos en un mismo nivel. De tal manera se realizarían los elementos necesarios para cada nivel a la vez que diseñamos el nivel.

Otra solución sería hacer una lista de elementos y eventos global, y empezar a hacer uno detrás de otro. Después se acabarían repartiendo en los distintos niveles de forma equitativa. Esta solución puede ser peligrosa porque se puede perder la cohesión y el sentido de la temática en el desarrollo de los distintos elementos a lo largo del videojuego.

Incluso se podrían hacer elementos y eventos a la carta, de manera que los desarrollaríamos cuando los fuéramos necesitando, por lo tanto, no se produciría un problema de cohesión ni de temática. Sin embargo, esta solución no parece la más conveniente dado que el desarrollo de niveles estaría muy separado y aislado de los elementos y eventos para puzles de los niveles, cosa que puede hacer peligrar la cohesión y avance correcto de la historia.

Diseño e Implementación del sistema de sonido del juego

Una solución sería crear un elemento de sonido que pudiese ser duplicable, y de tal manera ser replicado por todo el juego, haciendo los sonidos oportunos. Es un método sencillo y eficaz, pero no parece ser muy eficiente a la larga ni óptimo para conseguir unos correctos fps en el juego.

Otra solución sería diseñar un sistema de sonido no duplicable, y que no se eliminase cada vez que es usado. Sería general y tendría acceso a todos los audios del juego, los cuales iría reproduciendo en cada momento que fuese necesario a lo largo de los niveles.

Mecánicas y animación básicas de movimiento y combate del personaje

Una solución sería crear método general que permita al personaje moverse por todo el mapa de forma automática, o general bajo unas condiciones. De esta manera la creación del mapa no estaría ensuciada o limitada con elementos innecesarios.

Otra solución sería tener muchos elementos repartidos por el mapa que permitiesen al personaje hacer o no hacer ciertas acciones. Aunque parece sencillo, la verdad no lo es y además

conlleva un desarrollo lioso e ineficiente, por lo cual esta opción debería ser descartada rápidamente para no tener problemas mayores.

3.5 Solución propuesta

A lo largo del punto anterior se han descrito las distintas alternativas para alcanzar los objetivos. A continuación, se presentarán las soluciones escogidas.

- Para el sistema de niveles se ha optado por dividir el mapa del juego en 4 niveles, donde cada uno tenga una temática distinta siguiendo la historia.
- Para el sistema de diseño de enemigos se ha optado por dividir el diseño de enemigos por cada nivel del juego. De tal manera que se realizarán enemigos de la misma temática más fácilmente mientras nos quedemos en un mismo nivel. Se realizarán los enemigos necesarios para cada nivel a la vez que diseñamos el nivel.
- Para la creación de los menús del juego se ha optado por dividir cada menú en una serie de submenús y prefabs, de manera que se realice de forma modular y repetitiva. De esta manera, cuando se consiga hacer un menú o submenú, simplemente se podrá replicar lo que ya está realizado y modificar los elementos necesarios
- Para la creación del sistema de diálogo se ha optado por diseñar un sistema de diálogo no duplicable, y que no se elimine cada vez que se usa. Simplemente se esconderá y se irá reutilizando en cada momento a lo largo de los niveles.
- Para la creación y desarrollo de la historia se ha optado por crear la historia del juego de golpe y con anterioridad al desarrollo del videojuego y sus niveles. De esta manera, la historia puede guiar al desarrollador y al diseñador a lo largo del proyecto. Sobre todo, ayuda a la estética y cohesión del juego. Se representará la historia mediante el sistema de diálogo implementado en el juego.
- Para el desarrollo e implementación de los elementos y eventos del puzle del juego se ha optado por dividir el diseño de elementos por cada nivel del juego. Se realizarán elementos y eventos con misma temática más fácilmente mientras nos quedemos en un mismo nivel. Se realizarán los elementos necesarios para cada nivel a la vez que diseñamos el nivel.
- Para el sistema de sonido se ha optado por diseñar un sistema de sonido no duplicable, y que no se elimine cada vez que se usa. Será general y tendrá acceso a todos los audios del juego, los cuales iría reproduciendo en cada momento que sea necesario a lo largo de los niveles.
- Para las mecánicas básicas y animación del movimiento y combate del personaje se ha optado por crear un método general que permita al personaje moverse por todo el mapa. De esta manera la creación del mapa no estaría ensuciada o limitada con elementos innecesarios.

3.6 Plan de trabajo

Para el correcto diseño, desarrollo e implementación del proyecto ha sido necesaria cierta organización entre ambos compañeros. Como ya se ha comentado antes en la Metodología, el uso de una metodología ágil y el uso de distintas aplicaciones como Discord, GitHub y HacknPlan han permitido la correcta coordinación para la realización del diseño, desarrollo e implementación del proyecto.

A continuación, hablaremos de las planificaciones y estimaciones de horas y entregables, así como las fases en las que se ha dividido el desarrollo del TFG.

Fases:

- Diseño de niveles
- Mecánicas básicas personaje y animación del mismo
- Enemigos
- Elementos y eventos para puzles del juego
- Menús e Interfaces
- Miscelánea (Donde entran cosas como el desarrollo de la historia, el sistema de diálogo, selección de fuentes del juego)
- Selección de sprites, tiles y sonidos del juego

Dadas estas fases podemos estimar una serie de entregables y horas de trabajo, de manera que se repartan las horas de forma que se distribuyan más hacia tareas más extensas y menos hacia tareas más ligeras.

Disponemos de 7 fases. Si se dejan 60 horas para la realización de la memoria del TFG, quedarían unas 300 horas a repartir entre esas 7 fases.

Reparto de horas por fases:

Fase	Horas estimadas
Diseño de niveles	100h
Mecánicas básicas del personaje y su animación	20h
Creación de los enemigos	80h
Elementos y eventos para puzles del juego	30h
Menús e Interfaces	30h
Miscelánea (Donde entran cosas como el desarrollo de la historia, el sistema de diálogo, selección de fuentes del juego)	25h
Selección de sprites, tiles y sonidos del juego	15h

Tabla 25. Horas estimadas para cada fase

CAPÍTULO 4

Diseño de la solución

En este capítulo procederemos a definir el Proyecto de dos formas, mediante la arquitectura del Sistema y mediante el Diseño detallado del sistema.

Primero se contará cómo se realizará la solución de forma general, y a continuación se explicará con profundidad la solución.

4.1 Arquitectura del Sistema

Neyteria es un videojuego 2D de plataformas con cuatro niveles principales y uno secundario, que es el que permite el cambio entre niveles. Estos cuatro niveles están conectados mediante el nivel secundario ZoneLoader.

Para el sistema de niveles se ha optado por dividir el mapa del juego entero en varios niveles donde cada uno tenga una temática distinta siguiendo la historia. Además, encontramos nuevos enemigos y también nuevos elementos.

Los enemigos que encontraremos a lo largo del juego se pueden clasificar de varias formas. Podríamos clasificarlos por nivel, por funcionalidad similar, por herencia o incluso por sus estadísticas de daño y vida.

Estos enemigos tienen una inteligencia que les permite esperar al jugador, e ir a por él en cuanto lo detecten. Tienen implementado una serie de ataques y habilidades que les permiten atacar al jugador y ser una amenaza.

4.2 Diseño detallado

En esta sección hablaremos sobre los dos sistemas más grandes a implementar. El sistema de niveles y el sistema de enemigos. Dado que ambos abarcan una gran extensión, se explicarán de forma separada.

Tras la explicación de estos dos sistemas, se procederá a explicar el resto de sistemas.

Niveles del juego

El sistema de niveles se basa en 4 niveles conectados con distintas características y eventos. A continuación, se mostrará un resumen de la posición y conexión de los cuatro niveles:

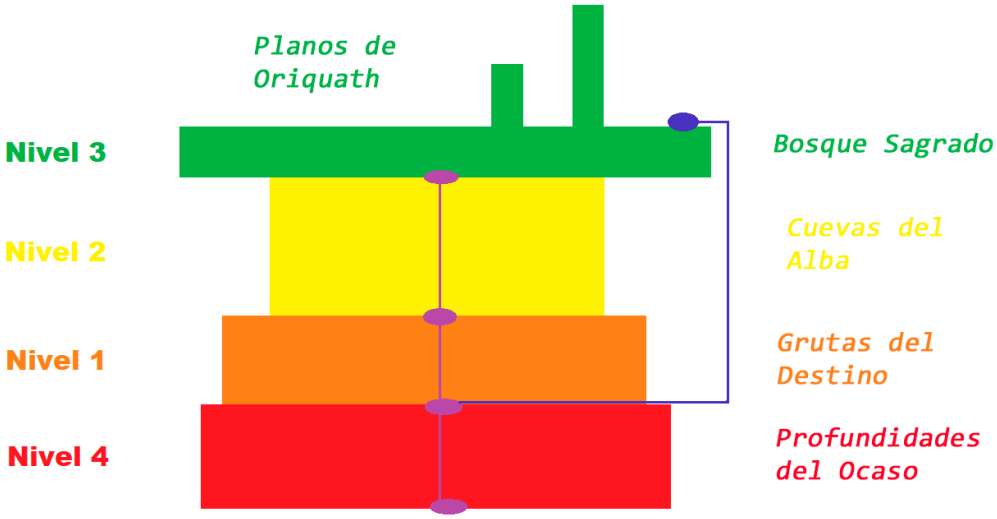


Figura 9. Boceto de los niveles del juego

En el primer nivel, el protagonista explorará un lugar amplio, con una iluminación deficiente y una temática de cuevas. Allí descubrirá qué elementos puede obtener, qué son las armas, qué son los trajes, para qué sirven los módulos y cómo se usan. Aprenderá a realizar los movimientos básicos, y también los de combate. El jugador estará bastante libre, para que pueda adaptarse y habituarse al juego cómodamente. La dificultad de los enemigos es bastante baja, por lo que no tendrá problemas en pasarse el nivel con tranquilidad.

En este nivel, además, aprenderá a usar su interfaz, para poder mirar su estado y poder usar la tienda, donde podrá comprar objetos, módulos o armas y trajes. Para pasarse el nivel deberá acabar con el boss del nivel, que espera la llegada del protagonista.

En el segundo nivel, el jugador explorará un lugar un poco más cerrado, con más saltos y plataformas más complicadas. Este nivel dispondrá de mejor iluminación. Aquí los enemigos serán un poco más fuertes, y veremos nuevos enemigos con nuevas mecánicas. También nos encontraremos nuevos elementos para puzles puestos por el nivel. En este nivel, se da por hecho que el jugador ya se ha acostumbrado a los controles del juego, pero que aún no los ha interiorizado, por lo que la dificultad aumentará, pero no tanto.

El jugador deberá usar la tienda y/o el cambio de armas si quiere pasarse el nivel cómodamente, aunque podría no hacerlo y pasar el nivel como si fuera un reto más difícil. Al igual que el nivel anterior, derrotar al boss final del nivel desbloquea el paso al nivel siguiente.

En el tercer nivel, se da por hecho que el jugador ya tiene control total de los controles del juego, y por tanto la dificultad de este nivel sube bastante. Este nivel será al aire libre, por lo que habrá más luz. También encontrará elementos nuevos para puzles, y nuevos enemigos.

Este tercer nivel será más largo que los demás y, por ello, dispondrá de dos bosses de nivel. El jugador deberá ser capaz de usar todos los recursos a su alcance, porque simplemente con buena técnica y controles es muy difícil pasarse el nivel. Al haber dos bosses, habrá dos zonas

de tienda en el nivel, lo cual ayudará al usuario a actualizarse y ganar herramientas y mejorarse las estadísticas para prepararse para el cuarto y último nivel.

En el cuarto nivel, se asume que el jugador domina los controles y el sistema de equipamiento, por lo tanto, la dificultad de este nivel sube drásticamente. Este nivel es en el subsuelo, debajo del primer nivel, por lo tanto, la iluminación será deficiente. Encontraremos nuevos elementos para puzles, y bastantes nuevos enemigos.

Este cuarto nivel será complicado, por lo cuál parecerá más largo que todos los demás al tener que ir con cuidado todo el rato. No es posible pasarse este nivel sin buena equipación, aunque se disponga de una técnica maravillosa. Aunque no hay boss a mitad de nivel, sí que habrá una zona de tiendas para recuperarse antes del tramo final.

El boss del cuarto nivel es el más complicado de los cinco, por lo que es recomendable que el usuario consiga todas las ventajas que tenga a su alcance, como por ejemplo encender los ventiladores para poder acceder a las plataformas móviles superiores.

El nivel ZoneLoader se encuentra entre los niveles principales, y es en el cuál se le permite al jugador realizar cambios de equipamientos, comprar en la tienda, e incluso el cambio de niveles.

A continuación, se muestra el diagrama de flujo de las escenas que tiene el juego.

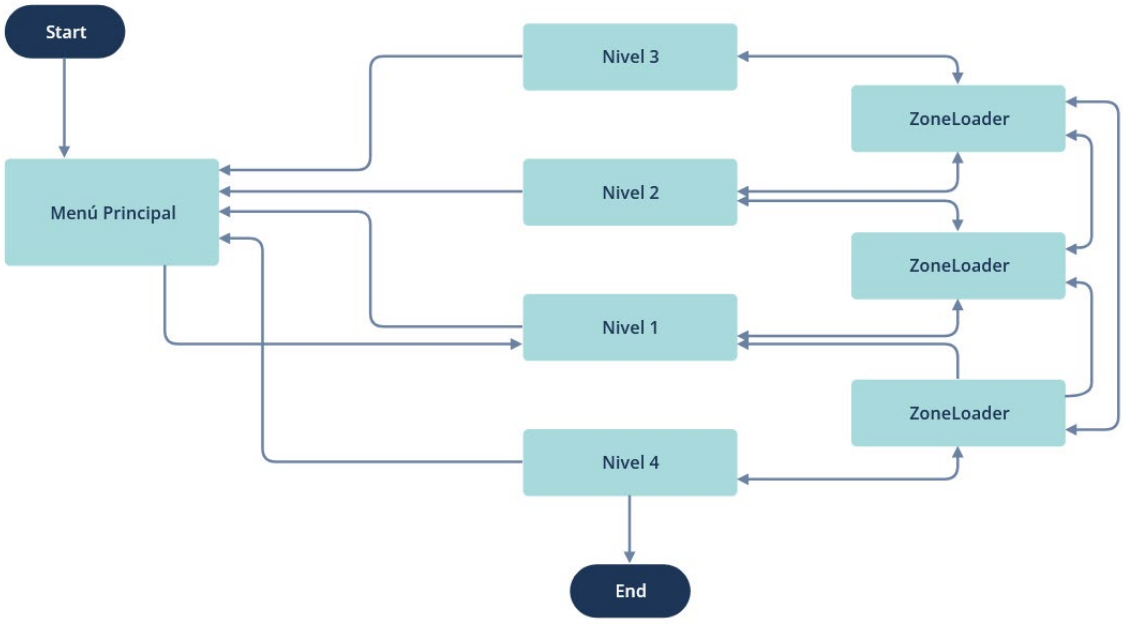


Figura 10. Flujo de escenas del juego Neyteria

Enemigos del juego

Los enemigos del juego cuentan con distintas inteligencias que le permiten perseguir y atacar de una forma efectiva.

Los enemigos parten de la base de que tienen la capacidad de hacer guardia esperando a que entre el jugador en su rango de visión. Y cuando eso sucede, deciden ir a por él y usar sus ataques.

Esto es una gran amenaza hacia el jugador, ya que va a estar siempre expuesto a ataques de enemigos en cuanto se acerque a ellos más de lo que debe.

Distintos enemigos tienen distintos tipos de ataques. Por ejemplo, el enemigo NightBorne se encarga de realizar ataques de cerca, mientras que el enemigo Archer realiza ataques de lejos (proyectiles). Al partir de ahí podemos encontrar variaciones en su movimiento. Por ejemplo, el enemigo Bat1 tiene el sistema de ataque del Archer, pero su movimiento es distinto (vuela y mantiene la distancia con el protagonista). Algo parecido pasa con el enemigo TinyFly, que hereda de NightBorne, pero persigue al enemigo volando.

Modificando trozos de código de los anteriores enemigos, podemos crear la inteligencia del enemigo ImpMaze, que le permite, mediante una probabilidad, no morir y volver a levantarse (aunque con una penalización de vida).

Para los bosses el código se vuelve un poco más complejo. La máquina de estados se empieza a agrandar ya que no tiene un simple ataque. Ahora tiene un ataque de cerca y una habilidad a distancia. Además, realiza una serie de invocaciones cuando la vida le baja cierta cantidad.

Usando el código de los anteriores enemigos, podemos añadir funcionalidades para hacer que los bosses tengan varias habilidades, e incluso anden diferente cuando les baje la vida de cierto punto. Este es el caso de ElementalWater, donde el jugador ve por primera vez el uso de varias habilidades por parte del boss.

Finalmente, mediante la inclusión de unas funcionalidades más al código, se puede conseguir que al morir se invoque algún enemigo. Este es el caso del boss final DemonOfAbyss que invoca al SlimeBoss tras su muerte.

Los enemigos que encontraremos a lo largo del juego se pueden clasificar de varias formas. Podríamos clasificarlos por nivel, por funcionalidad similar, por herencia o incluso por sus estadísticas de daño y vida.

Antes de entrar en destalles, definiremos qué son los prefabs.

Los prefabs [16] son objetos reutilizables, y creados con una serie de características dentro de la vista proyecto, que serán instanciados en el videojuego cada vez que se estime oportuno y tantas veces como sea necesario. El prefab actúa como una plantilla a partir de la cual se pueden crear nuevas instancias del objeto en la escena. Cualquier edición hecha a un prefab asset será inmediatamente reflejado en todas las instancias producidas por él.

A partir de estos prefabs, se pueden crear los prefabs variants [17], los cuales permiten usar un prefab como base, y al partir de él, realizar cambios sin influir en el prefab original. La modificación de elementos del prefab original, producirá la actualización en los prefabs variants hechos al partir de él, siempre que esos elementos no hayan sido modificados por el prefab variant.

A continuación, se describirá en listas, la clasificación de los enemigos por nivel en el que aparecen.

Enemigos Nivel 1

- Slime
- Slime Berserker
- Archer
- FireWorm
- NightBorne
- HAO4
- HAO4Berserker
- BringerOfDeath

Lista 1. Enemigos del Nivel 1

Enemigos Nivel 2

- SlimeNvl2
- Slime Berserker
- ArcherNvl2
- Bat1
- Bat2
- NightBorneNvl2
- NightBorneBerserker
- ElementalWater

Lista 2. Enemigos del Nivel 2

Enemigos Nivel 3

- HAO4Lvl3
- ArcherHero
- Bat2
- TinyFly
- NightBorneNvl3
- NightBorneBerserker
- EyeBallMonster
- ElectricDefendant
- EvilWizard

Lista 3. Enemigos del Nivel 3

Enemigos Nivel 4

- SlimeNvl4
- Slime Berserker
- HAO4Lvl3
- ArcherNvl2
- NightBorneNvl2
- Bat2
- FireWormLvl4
- TinyFlyLvl4

- NightBorneNvl4
- NightBorneBerserker
- ImpMaze
- ImpAxe
- ImpAxeBerserker
- EyeBallMonster
- DemonOfAbyss
- SlimeBoss

Lista 4. Enemigos del Nivel 4

A continuación, se describirá en una lista una clasificación de los enemigos por función similar. Se asume que los prefabs variants tienen la misma funcionalidad que el prefab original.

Funcionalidad Similar

- Slime
- Archer – ArcherHero – FireWorm
- NightBorne – HAO4 - EyeBallMonster
- Bat1
- Bat2
- TinyFly
- ImpMaze – ImpAxe
- BringerOfDeath
- ElementalWater – EvilWizard – ElectricDefendant - SlimeBoss
- DemonOfAbyss

Lista 5. Enemigos con Funcionalidad Similar

A continuación, se mostrará en un diagrama una clasificación de los enemigos por herencia de clases.

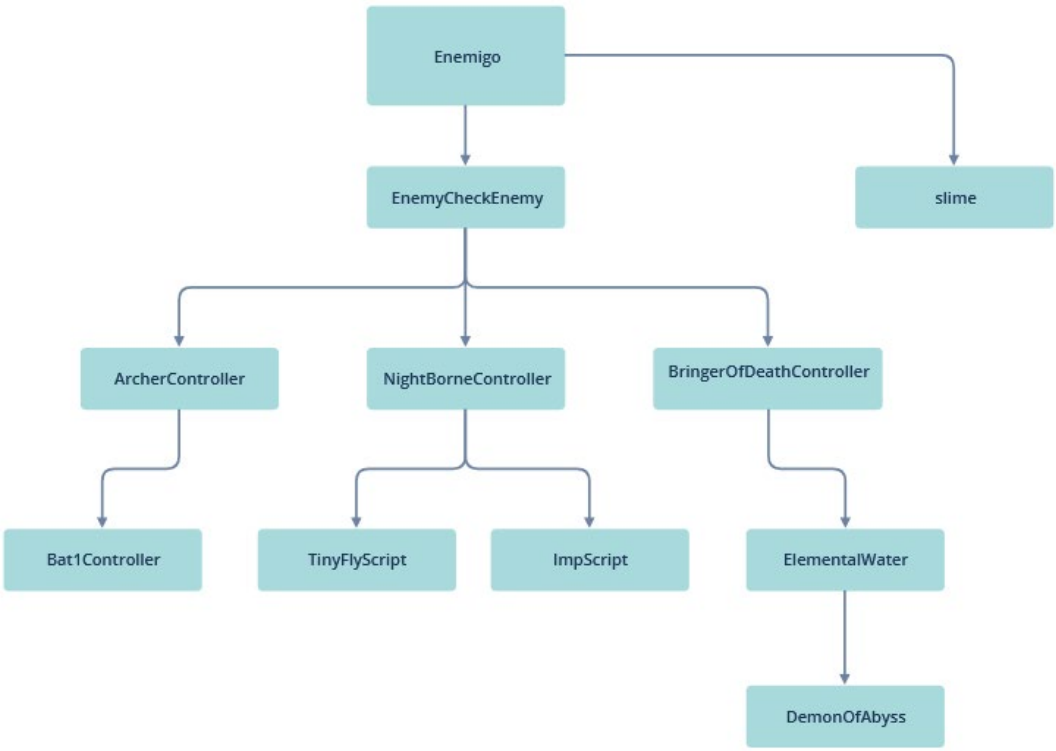


Figura 11. Herencia de clases de los Scripts de los enemigos

Resto de sistemas

En este subapartado nos encargaremos de definir y explicar cómo será la realización y funcionamiento del resto de sistemas cuyos objetivos hemos de cumplir.

A continuación, pasaremos a explicar el sistema de menú principal.

Para la creación del menú principal se dividirá cada menú por una serie de submenús y prefabs, de manera que un menú se realice de forma modular y sistemática. De esta manera, al hacer un menú o submenú, simplemente será replicar y modificar los elementos necesarios

El menú principal se basará en la creación de una escena en la cual habrá un canvas que contendrá los submenús correspondientes a las partes de este.

A continuación, se muestra unos mockups con el resultado esperado para la interfaz de usuario del menú principal:

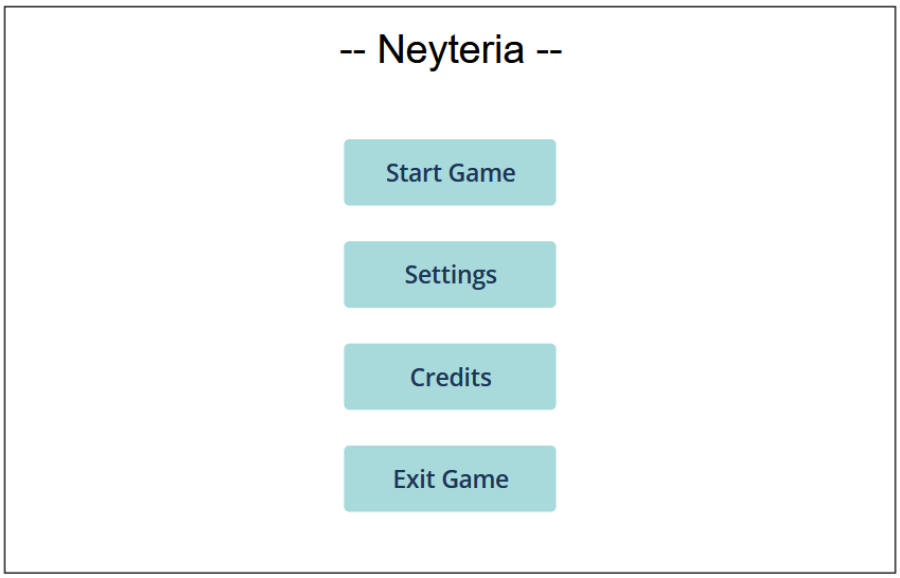


Figura 12. mockup del menú principal

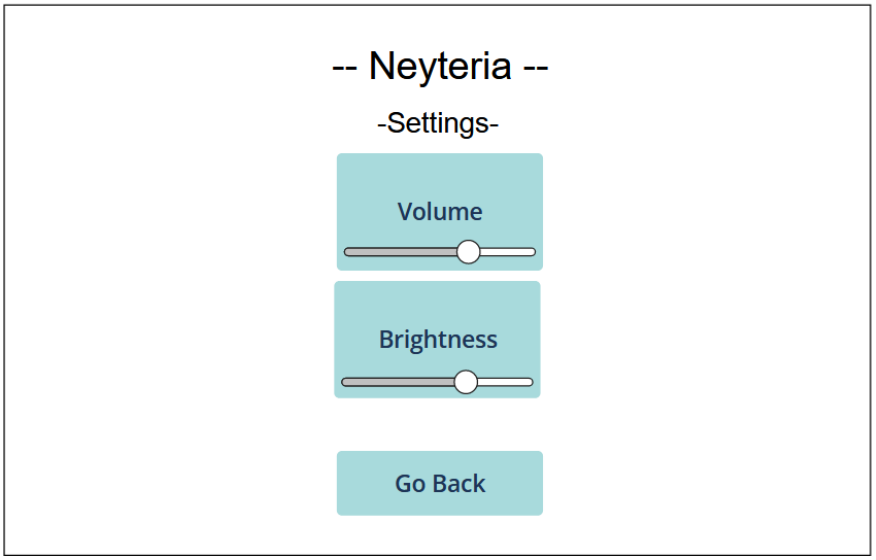


Figura 13. mockup del submenú opciones

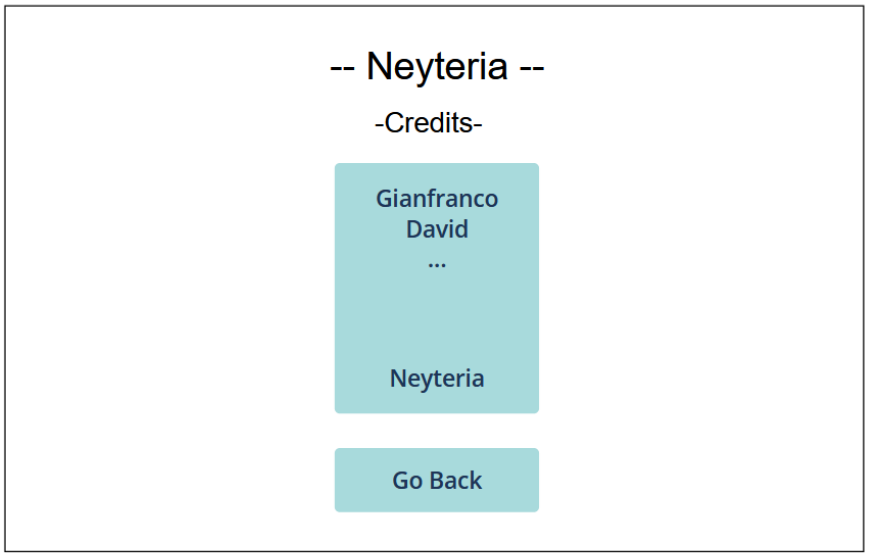


Figura 14. mockup del submenú créditos

El menú de pausa se realizará mediante un canvas y no estará en una escena aparte, ya que debe ser invocado cuando el jugador ha iniciado el juego.

A continuación, mostraremos un mockup del menú de pausa:

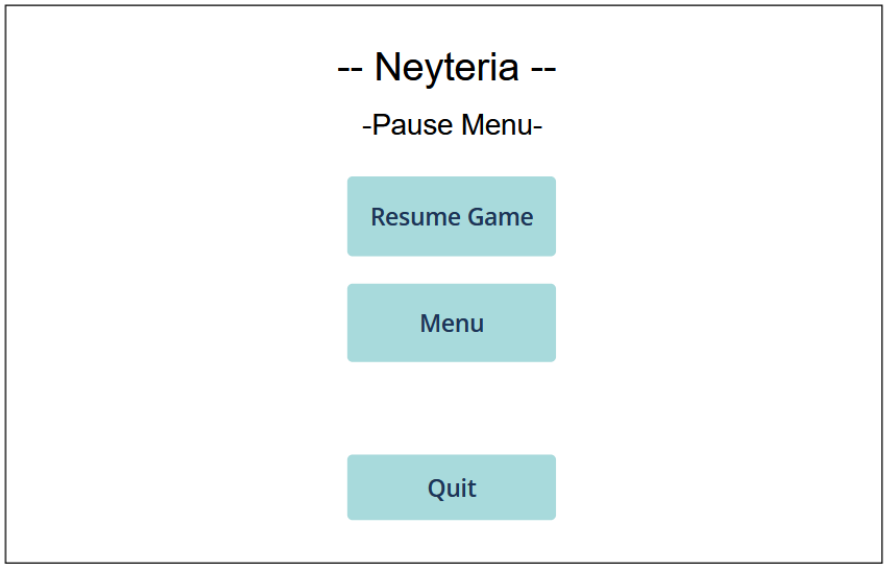


Figura 15. Moqup del menú de pausa

Ahora mostraremos un diagrama del flujo de interfaces en el juego:

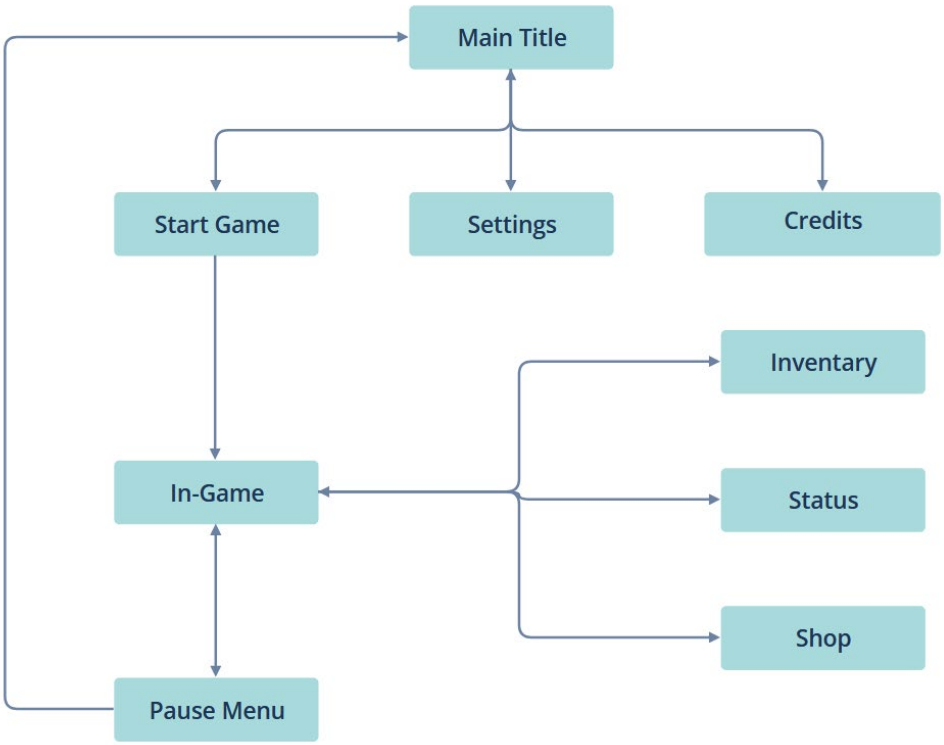


Figura 16. Flujo de interfaces en el videojuego Neyteria

La implementación artística del HUD se muestra, también, usando un mockup como guía para su desarrollo.

Figura 17. Diseño artístico del HUD

Para la pantalla de muerte simplemente se usará un canvas con un texto indicando que el jugador ha perdido.

Las mecánicas de combate se realizarán mediante la implementación de tres métodos, uno para cada tipo de arma. Estos métodos se llamarán SwordAttack, GunAttack y MazeAttack, los cuales representan el ataque con los respectivos tipos de arma del juego. El salto mejorado se realizará mediante el uso de un método jump, que contendrá Coyote Time y Time Buffering, haciendo que el juego sea más suave y fluido a la hora de usar los controles.

El Coyote Time y el Time Buffering son dos técnicas usadas en los videojuegos para mejorar la jugabilidad. El Coyote Time se basa en dar al jugador un tiempo de margen para saltar una vez que el personaje ha abandonado una plataforma y ha empezado a estar en el aire. Por otro lado, el Time Buffering se basa en permitir al jugador un margen de tiempo para que este pueda pulsar el botón de saltar antes de haber caído a la plataforma correspondiente, y se ejecute el salto en el instante que el personaje toque la plataforma.

El sistema de diálogo se realizará mediante un canvas que se esconda cuando no sea usado, y un sistema de triggers que llamen a este canvas cuando sea necesario mostrar diálogo en el juego.

El sistema de sonido se basará en un objeto invisible, que sigue al jugador y que, dada una serie de sonidos asociados, se le pida que reproduzca un sonido de los que tiene asociados en un momento dado.

4.3 Tecnología utilizada

Para el desarrollo de este TFG han sido necesario una serie de programas y aplicaciones. Algunos de estos programas sirvieron para coordinación, otros para la realización del código, otros para la ejecución del código y otros la gestión de versiones.

A continuación, se irá explicando cada uno con detalle:

Unity

Unity es uno de los motores de videojuegos más populares actualmente. Cuenta con una base de usuarios tan inmensa, que la mayoría de dudas que puedan aparecer en un proyecto está resulta por alguien en un blog o incluso en la propia documentación de la página oficial Unity [18].

Unity es capaz de realizar juegos tanto en 2D como en 3D, cubriendo así una gran variedad de videojuegos. Además, Unity incluye una versión personal que es gratis para el usuario. De esta manera, fomenta que los pequeños creadores se animen a usar este motor frente a otros.

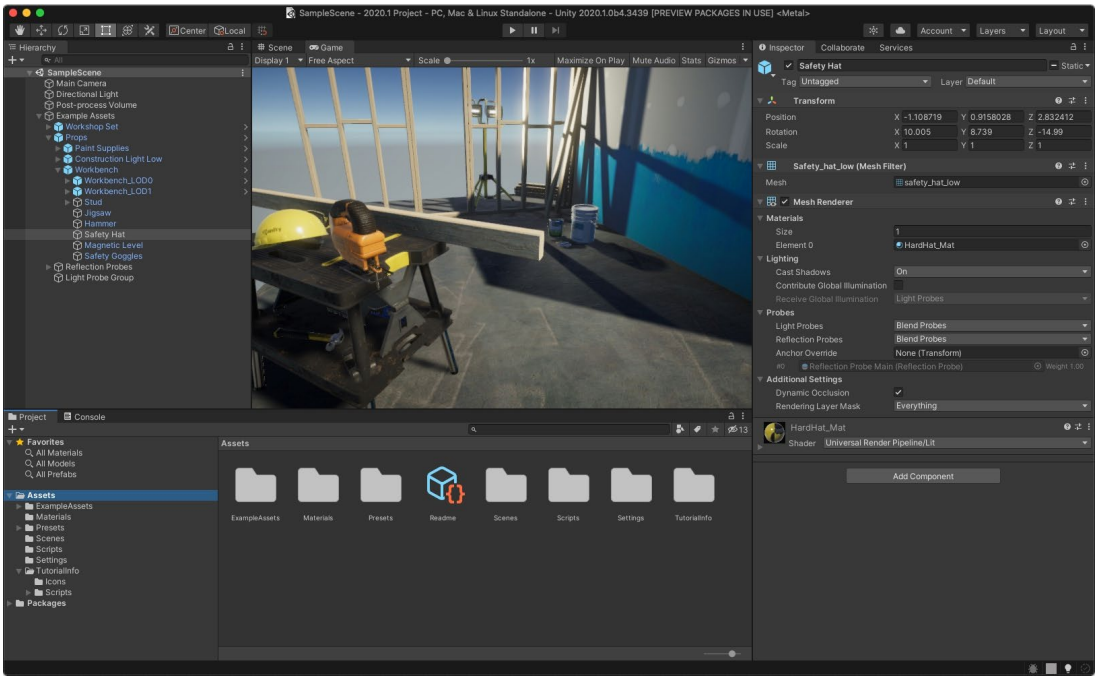


Figura 18. Editor de Unity

La interfaz y editor de Unity son de los más sencillo de usar en el mercado, por lo que se vuelve muy atractivo para usuarios nóveles. De esta forma, puedes hacer productos profesionales gratis y con no demasiada dificultad. Todo esto explica la gran base de fans y usuarios que dispone el motor.

Unity cuenta con características que le mantienen a flote en el mercado de los videojuegos. Una de ellas es la capacidad de crear ports a distintos dispositivos de manera sencilla. Con crear una vez un juego, Unity es capaz de construir dicho juego en diversas plataformas y dispositivos.

Otra cosa atractiva de Unity es el uso del lenguaje C#, el cual es potente y bastante sencillo de leer incluso para los que están empezando.

Como guinda del pastel, Unity tiene la capacidad de importar y exportar assets externos de la asset store. De esta manera, se pueden encontrar materiales con los que partir y tener una base con la que realizar los proyectos.

Además, tiene facilidades para realizar colaboraciones o simplemente guardar los proyectos en git, porque es compatible con la aplicación de GitHub Desktop.

Por último, remarcar que Unity cuenta con la aplicación de escritorio llamada Unity Hub, que permite gestionar las distintas versiones y proyectos realizados en unity.

Git y GitHub

El uso de GitHub [19] como repositorio resulta en una gran facilidad y adaptabilidad a proyectos ya sean en solitario o cooperativos. Mediante el uso de Git, GitHub [20] puede acceder al sistema de gestión de versiones y ofrecerte servicios, tal como antiguamente hacía Apache [21], pero de forma descentralizada y más cómoda.

GitHub ofrece que los desarrolladores alojen proyectos creando repositorios de forma gratuita, aunque el portal obliga a los usuarios que lo usan gratis a tener código abierto.

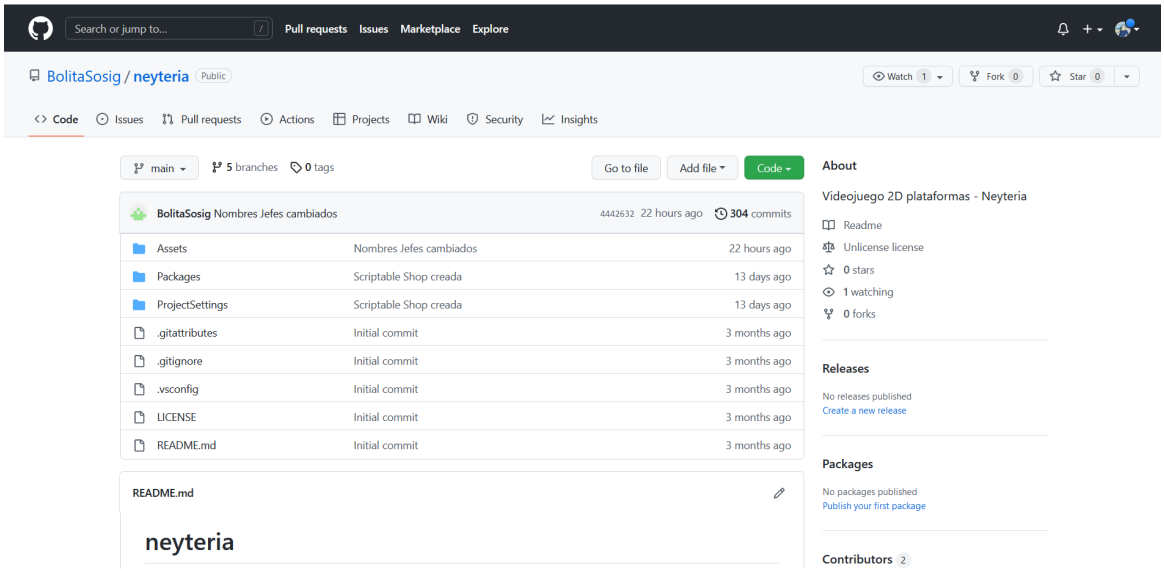


Figura 19. Repositorio neyteria en GitHub

GitHub también ofrece una serie de herramientas propias con las que complementar las ventajas que ya tiene el sistema Git de por sí. Por ejemplo, puedes crear una Wiki para cada proyecto, de forma que puedas ofrecer toda la información sobre él y anotar todos los cambios de las diferentes versiones. También goza de un sistema de seguimiento de problemas, y de herramientas de revisión de código.

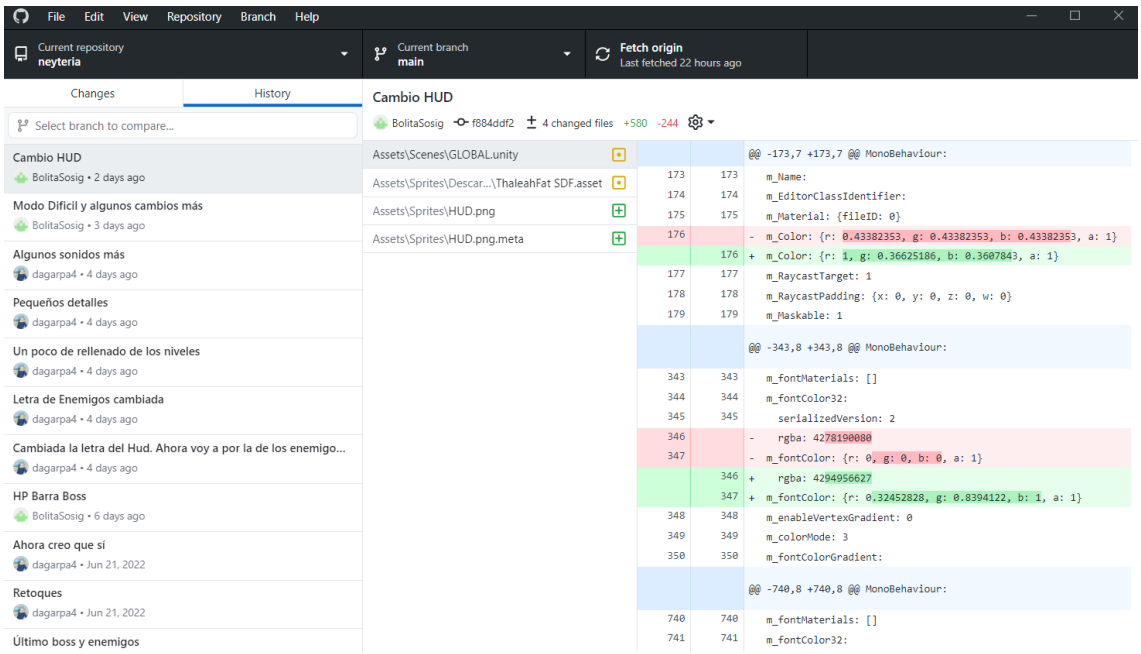


Figura 20. Interfaz de GitHub Desktop con el proyecto neyteria

Lo más importante para este proyecto es la aplicación de GitHub Desktop, que se puede utilizar para subir, descargar y gestionar proyectos de Unity. Esta compatibilidad ha propiciado

una gran facilidad para la cooperación en el proyecto entre compañeros, añadiendo, además, herramientas extra para mejorar el desarrollo de los proyectos Unity como, por ejemplo, crear, agregar y clonar repositorios, colaborar con compañeros a través de informes o solicitudes y mantener el repositorio local sincronizado con los datos de la nube.

Microsoft Visual Studio

El IDE de Visual Studio en entorno de desarrollo integrado que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación. Un entorno de desarrollo integrado (IDE) es un programa con numerosas características que respalda muchos aspectos del desarrollo de software.

Aparte del editor y el depurador estándar que proporcionan la mayoría de IDE, Visual Studio incluye compiladores, herramientas de finalización de código, diseñadores gráficos y muchas más características para facilitar el proceso de desarrollo de software.

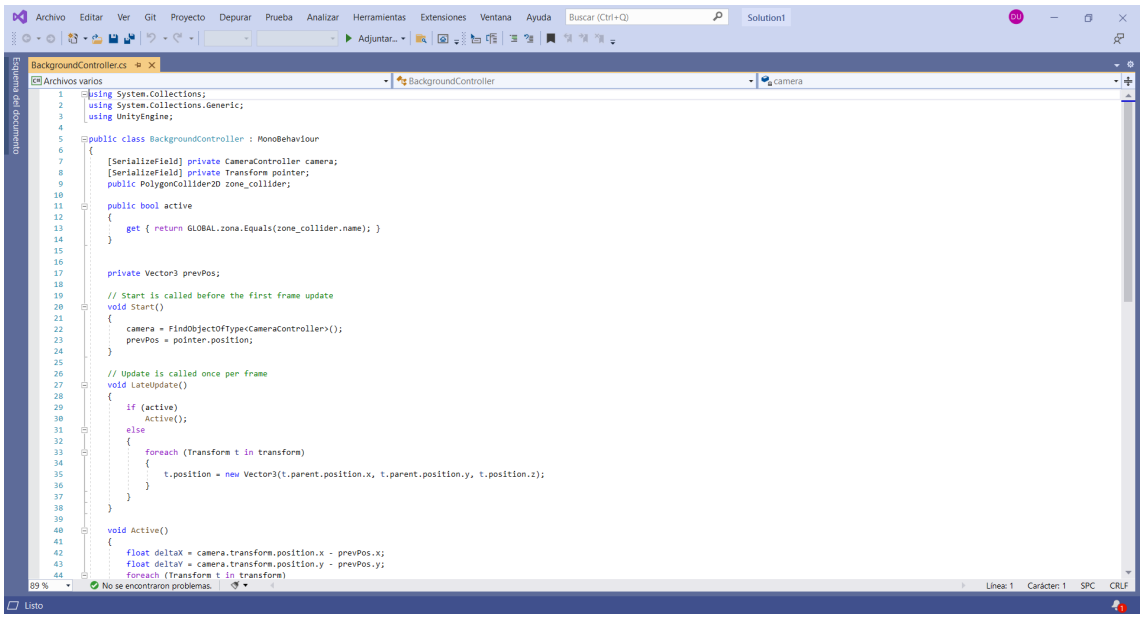


Figura 21. Editor e Interfaz de Visual Studio

Visual Studio está disponible para Windows y Mac. Existen tres ediciones de Visual Studio: Community, Professional y Enterprise.

Visual Studio cuenta con características útiles y populares, que mejoran la productividad al desarrollar software. Estas características son: Subrayados ondulados y Acciones rápidas, Limpieza de código, Refactorización (que incluye operaciones como el cambio de nombre inteligente de variables, la extracción de una o más líneas de código en un nuevo método y el cambio del orden de los parámetros de método), IntelliSense, Live share, ….

HacknPlan

HacknPlan reúne la documentación de diseño de juegos y la gestión de proyectos en una sola herramienta de producción de juegos, y proporciona una forma semántica de organización, planificación y seguimiento del progreso de tu juego.

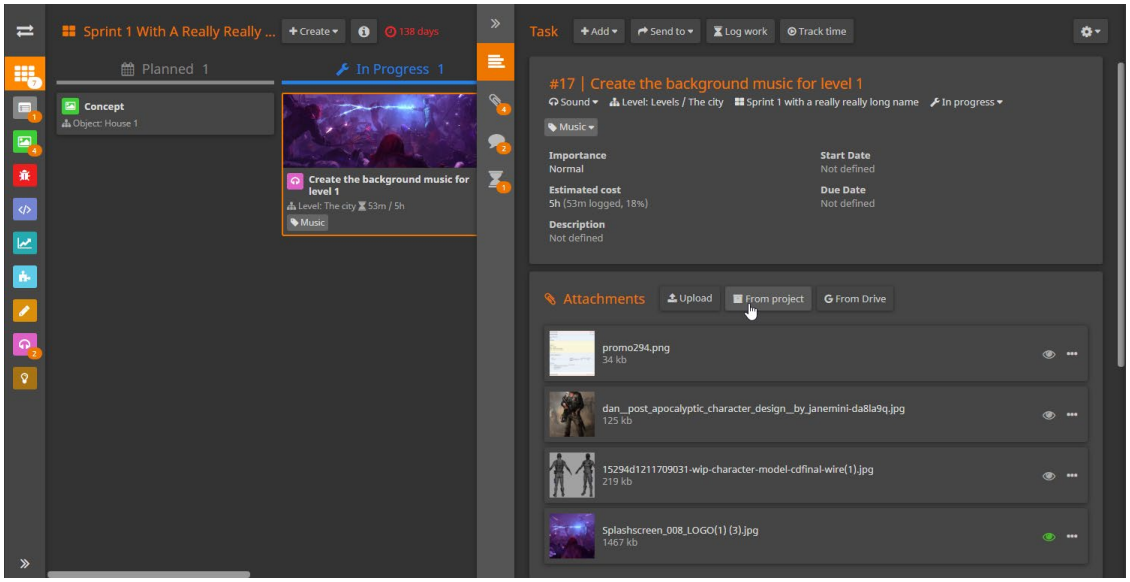


Figura 22. Interfaz de HacknPlan

Es la aplicación recomendada por el profesor. Tras haber probado Trello y Monday, hemos decidido usar HacknPlan por su adaptación a los videojuegos. Aunque es más complicada que las dos anteriores nombradas, tiene más libertad y opciones que permiten una mejor descripción y fluidez a la hora de plasmar la información y entregables del proyecto.

CAPÍTULO 5

Desarrollo de la solución propuesta

En este capítulo se mostrará el resultado de la solución propuesta, de forma que se explique cómo se ha pasado de la propuesta a la solución final.

El capítulo se basa en apartados donde se muestra el resultado del juego y donde se describe cómo ha sido el desarrollo de la solución.

Finalmente encontraremos otro apartado donde se muestra información sobre la organización y el tiempo usado en el desarrollo de la solución (fases del plan)

5.1 Desarrollo de los niveles del videojuego

El desarrollo de los niveles del juego ha sido un proceso lento y tedioso, sobre todo al principio. Al principio no había referencias sobre a qué distancias se podían colocar plataformas y enemigos, por lo que este trabajo prematuro se basó en un prueba y error buscando un mapa cómodo e interesante.

Por suerte, no se tardó mucho en encontrar una forma eficiente y cómoda de construir los niveles sin tener que recurrir a pruebas frecuentes. Esta forma permitió la construcción de niveles directamente desde los bocetos, y haciendo así que este desarrollo fuese más fluido y sistemático.

Los niveles del videojuego se basan en un conjunto de tiles pintados en la escena mediante un grid. Esta forma de rellenar el mapa usando tiles hace que la realización del mismo sea mucho más eficiente. Sin embargo, el tamaño de los niveles y la complejidad que tienen algunos saltos entre plataformas hacen que esta construcción de niveles lleve más tiempo del esperado.

Tras haber construido los niveles, había que lograr la forma en la que estos tiles tuviesen colliders para que el personaje no los atravesara. La solución es añadir componentes al tilemap, en específico los siguientes tres componentes: Tilemap Collider, Composite Collider 2D y Platform Effector 2D. Estos componentes, junto a un rigidbody permiten la correcta interacción del personaje con el nivel hecho por tiles.

Tilemap Collider se encarga de ponerle un collider a cada tile. Composite Collider 2D se encarga de unir estos colliders en unos colliders generales más grande, reduciendo y optimizando así el número final de colliders. Platform Effector 2D es el que se encarga de que el personaje principal interactúe correctamente con estos colliders, de forma que se produzca de una forma física y natural. Rigidbody permite al objeto tener un comportamiento físico y realista en el videojuego.

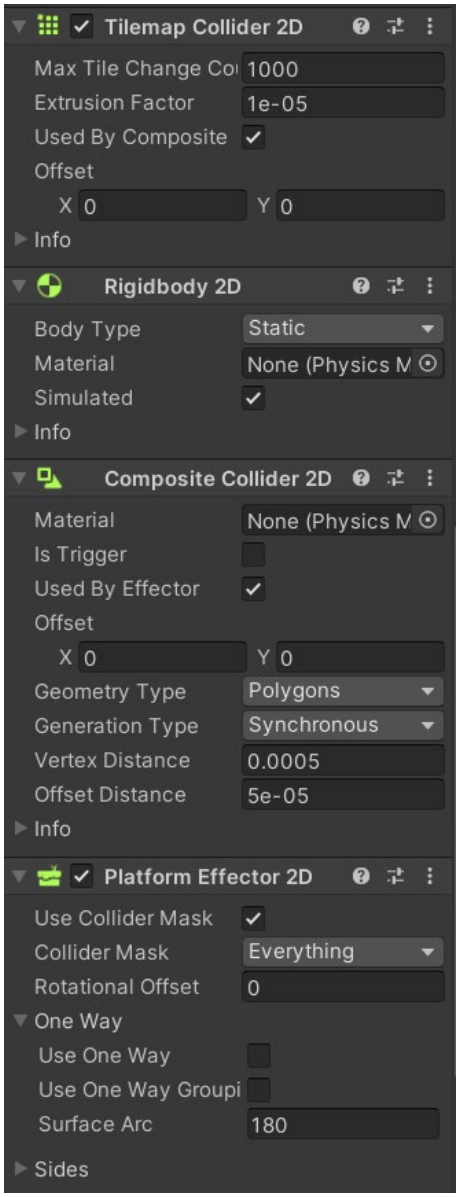


Figura 23. Componentes del objeto tilemap

De esta manera, los niveles ya eran funcionales y el personaje se podía mover libremente por este. A partir de este momento, realizar los niveles no era más que dedicarle horas y horas hasta que se plasmase el boceto a la práctica.

A continuación, se mostrará el resultado de los niveles desarrollados en el proyecto Neyteria:

Figura 24. Nivel 1 del videojuego Neyteria

Siguiendo lo que dice la historia, el protagonista se encuentra solo en el subsuelo y lucha para poder sobrevivir. Por lo tanto, la temática del nivel ha sido de cueva con poca iluminación, y con colores más oscuros.

Figura 25. Nivel 2 del videojuego Neyteria

Cuando el jugador atraviesa el primer nivel, y por tanto llega al segundo, se encuentra que hay una mejor iluminación y cierta vegetación, lo que le indica que ese camino lo llevará a la superficie y por tanto será libre al final del camino.

Figura 26. Nivel 3 del videojuego Neyteria

Este nivel le permite al protagonista llegar a la superficie y sentir el aire puro y la luz del sol por primera vez en mucho tiempo. La vegetación y la iluminación abundan en esa zona. Este nivel, además, se divide en dos subniveles: uno en el que el protagonista escapa del subsuelo y visita el árbol Ymrafihl, y otro en el que se encuentra con un bosque denso y recto, que tras un largo camino de árboles y escombros se encuentra con una catedral.

Figura 27. Nivel 4 del videojuego Neyteria

Tras ver algo en la catedral, el protagonista decide meterse por un atajo que se hallaba en lo más profundo del edificio, y que le acaba devolviendo al subsuelo, esta vez más profundo que nunca. En esta zona destacan los tonos oscuros y rojizos, a la vez que la iluminación vuelve a ser escasa.

Los niveles tienen un tamaño considerable y están llenos de pequeños detalles que son interesantes desde el punto de vista del diseño de niveles, como por ejemplo las zonas de plataformeo donde hay saltos complicados sin presencia de enemigos, y plataformas preparadas para elementos activables o desactivables que limitan o determinan el movimiento del jugador por el mapa. A continuación, se mostrará una visión global de los cuatro niveles juntos, y como quedaría el mapa del juego completo:

Figura 28. Niveles unidos del videojuego Neyteria

5.2 Elementos y eventos para puzzles en los niveles del juego

Dentro de los niveles podemos ver elementos que hacen que estos sean más interactivos y menos monótonos. Propician una mejor inmersión y entretenimiento dentro del juego. Estos elementos podemos separarlos en dos grupos: los interactuables mediante una tecla, y los que directamente actúan con el protagonista.

En el lado de los interactuables disponemos de dos elementos: toggleButton y toggleButtonBlue. El primero se encarga de generar u ocultar plataformas por el mapa, ya sea de forma temporal o permanente. En cambio, el segundo se encarga de activar o desactivar otros elementos de los niveles del juego.

Por ejemplo, veamos el caso del toggleButton:

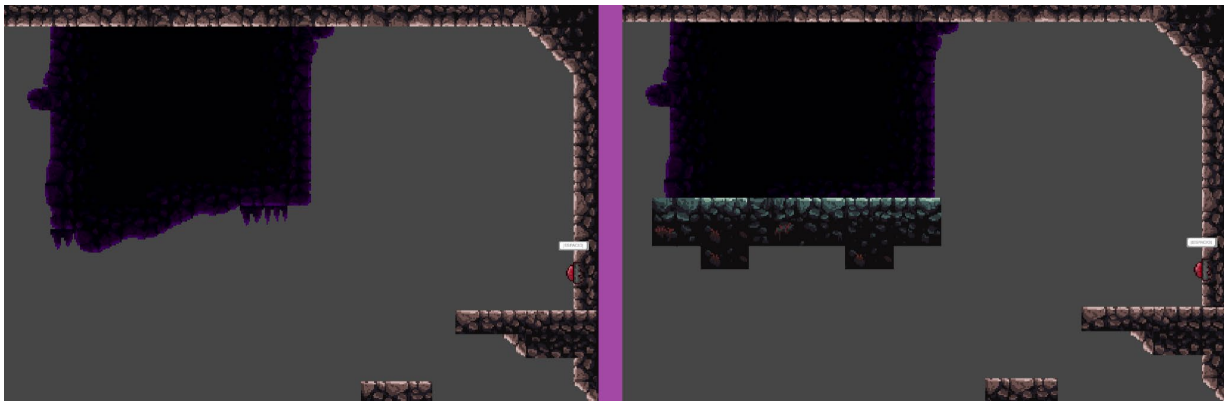


Figura 29. ToggleButton y una plataforma desactivada

Como se puede suponer, la activación del botón llevará a cabo la aparición de la plataforma que yace escondida tras la roca morada.

En el lado de los elementos que actúan directamente con el protagonista corrientes de lava, agua, plataformas destruibles, ventiladores ascendentes, plataformas móviles y plataformas que se disuelven al tocarlas.

A continuación, se comentarán las características principales de cada uno de estos elementos:

- El contacto con una corriente de lava, significa la muerte instantánea del personaje
- La corriente de agua implica un desplazamiento lateral en la dirección de la corriente, pudiendo permitir el movimiento del personaje o no (determinado por un boolean)
- Como indica su nombre, las plataformas destruibles pueden derrumbarse si el jugador le da un número suficiente de golpes o ataques con el arma.

- Los ventiladores ascendentes producen un desplazamiento ascendente del personaje cuando este se coloca encima. Estos ventiladores pueden estar encendidos o apagados.
- Las plataformas móviles permiten al personaje subirse en ellas y ser transportado. El código de estas plataformas es un poco complejo debido a que, por la forma de haber creado el videojuego, no es viable hacer al protagonista hijo de la plataforma. Por lo tanto, el movimiento de la plataforma al personaje ha sido proporcionado de otra manera, específicamente mediante una modificación del transform del protagonista cuando se sube a la plataforma y activa el trigger correspondiente.
- Por último, las plataformas que se disuelven solas al tocarlas el personaje, son específicamente eso. Mediante el sistema de trigger comentado en la plataforma anterior, el script produce la difuminación y desaparición temporal de la plataforma cuando tiene contacto con el protagonista.

5.3 Desarrollo de los enemigos del juego

El desarrollo de los enemigos, al igual que el de los niveles, ha sido un proceso lento y tedioso, en el que ha primado un poco la prueba y error. Sin embargo, al ser un desarrollo de código más que un desarrollo artístico, se ha notado una fluidez y sencillez para poder realizar y continuar la creación de los distintos enemigos del juego.

Como se ha explicado en el apartado de Diseño Detallado, el desarrollo de los enemigos ha sido realizado de una forma eficiente gracias al uso de la herencia. De esta manera, la creación de nuevas variantes de enemigos se ha vuelto entretenida y más eficiente.

Desgraciadamente, aunque la herencia nos ahorre trabajo en el código, aún queda mucho trabajo de animación y búsqueda de sprites de enemigos que no puede ser acelerada.

En el videojuego, hay treinta y un enemigos distintos a lo largo del juego. Aunque, gracias a Unity, se ha podido realizar de una manera sencilla usando los prefabs y los prefabs variants.

La creación de prefabs variants, acelera vertiginosamente la creación de nuevos enemigos, siempre que sean bastante similares al original.

Solo se han creado once scripts de enemigos, por lo cual, para la realización de los enemigos del juego, ha habido una gran reutilización de ciertos scripts.

Antes de explicar los enemigos, cabe destacar que los prefabs variants, o sea, los enemigos creados a partir del prefab enemigo original, disponen de la misma inteligencia y características, con la salvedad de que tienen más tamaño, vida y daño que el original.

A continuación, se comentará uno por uno las distintas características de los treinta y un enemigos disponibles en el juego. Aclarar que en este apartado no se mostrarán imágenes de todos los enemigos dado a que en el anexo cuatro se encuentra el GDD, donde están los enemigos con más detalle y con sus imágenes.

Primero procederemos a explicar los enemigos normales y dejaremos para el final los bosses.

Enemigos del juego:

- Slime: este enemigo contiene un mínimo de inteligencia para poder desplazarse lateralmente, pero de forma ininterrumpida.

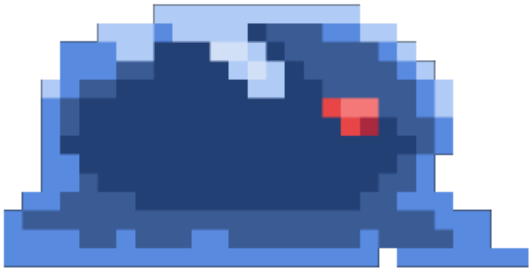


Figura 30. Sprite del enemigo slime

- Archer: este enemigo contiene la inteligencia para hacer guardia desplazándose lateralmente, y tiene la capacidad de detectar al jugador. Cuando el jugador es detectado, o sea, ha entrado dentro de su rango de reconocimiento, este enemigo perseguirá al jugador e irá disparándole agresivamente. Sus disparos invocan prefabs, gracias al método que proporciona Unity llamado Instantiate.

Figura 31. Sprite del enemigo Archer

- FireWorm: este enemigo es una copia del enemigo Archer, solo que le ha sido cambiado el componente animator y el proyectil para que parezca un enemigo nuevo.
- ArcherHero: lo mismo que FireWorm.
- NightBorne: este enemigo contiene la inteligencia para hacer guardia desplazándose lateralmente, y tiene la capacidad de detectar al jugador. Cuando el jugador es detectado, o sea, ha entrado dentro de su rango de reconocimiento, este enemigo

perseguirá al jugador e irá atacándole agresivamente. Sus ataques se basan en la función proporcionada por Unity llamada Physics2D.OverlapCircleAll que detecta si el jugador ha recibido el impacto en esa zona..

- HAO4: este enemigo es una copia del enemigo NightBorne, solo que le ha sido cambiado el componente animator y el transform del ataque para que parezca un enemigo nuevo.
- EyeBallMonster: lo mismo que HAO4.
- Bat1: este enemigo hereda sus capacidades del enemigo Archer, pero sobreescribe la función de persecución del protagonista. En vez de correr directo a por el player, se mantiene a una distancia adecuada para estar más seguro.
- Bat2: este enemigo es un prefab variant de Bat1, pero dado la modificación del boolean follow, este enemigo dispara proyectiles con seguimiento. Es más grande y duro que Bat1.
- TinyFly: este enemigo hereda sus capacidades del enemigo NightBorne, pero sobreescribe la función de persecución del protagonista, dado que este enemigo puede volar, irá directo hacia el protagonista flotando.
- ImpMaze: este enemigo hereda sus capacidades del enemigo NightBorne, pero sobreescribe la función de muerte. Este enemigo cuando muere tiene una probabilidad de revivir.
- ImpAxe: este enemigo es una copia del enemigo ImpAxe, solo que le ha sido cambiado el componente animator y el transform del ataque para que parezca un enemigo nuevo.

Bosses del Juego

- BringerOfDeath: este enemigo contiene la inteligencia para hacer guardia desplazándose lateralmente, y tiene la capacidad de detectar al jugador. Cuando el jugador es detectado, o sea, ha entrado dentro de su rango de reconocimiento, este enemigo perseguirá al jugador e irá atacándole agresivamente. Este enemigo dispone de un ataque cercano y un ataque lejano de proyectil. La elección de uno u otro es aleatoria. El ataque de proyectil tiene una baja probabilidad de invocar a un enemigo del nivel. Cuando su barra de vida baja de la mitad y del quinto, en ambos casos se cambia el color de su barra de vida y produce una invocación masiva de enemigos en el campo de batalla. Cuando muere, desbloquea la puerta que estaba cerrada al final de la sala del boss.

Figura 32. Sprite del enemigo BringerOfDeath

- ElementalWater: este enemigo hereda sus capacidades del enemigo BringerOfDeath, pero sobrescribe las funciones de ataque y daño para añadir dos habilidades adicionales, que se activan para su uso cuando baja de media vida. Además, cuando baja de media vida su forma de caminar también cambia.
- ElectricDefendant: este enemigo es una copia del enemigo ElementalWater, solo que le ha sido cambiado el componente animator, los transform de ataque y habilidades y el proyectil para que parezca un enemigo nuevo.
- EvilWizard: lo mismo que ElectricDefendant.
- SlimeBoss: lo mismo que ElectricDefendant.
- DemonOfAbyss: este enemigo hereda sus capacidades del enemigo ElementalWater, pero sobreescribe habilidades para poder ejecutar defensas, y también sobreescribe la función de muerte para invocar a SlimeBoss antes de morir.

Antes de finalizar con los enemigos, es conveniente la visualización de la máquina de estados del Boss ElementalWater:

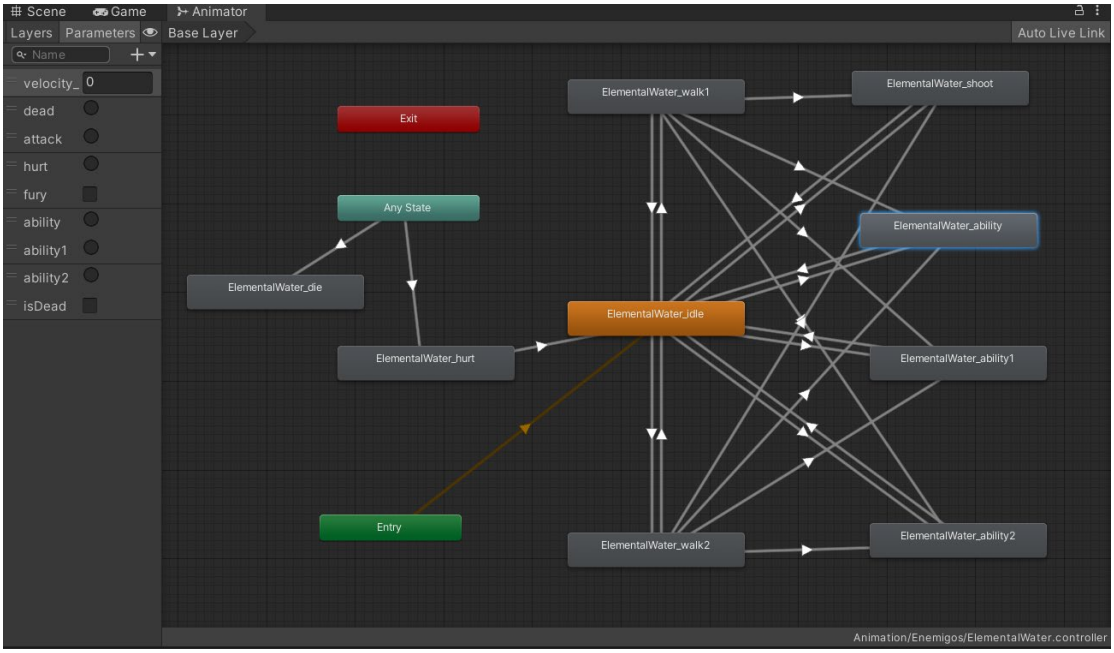


Figura 33. Máquina de estados del enemigo ElementalWater

La máquina de estados se puede entender como una red de nodos conectada y dirigida. Podemos observar que desde algunos estados se pueden realizar ciertas acciones, y desde otros no. Los estados Exit, Entry y Any State son comodines para realizar transiciones especiales. El estado entry indica cuál será el estado en el que iniciará la máquina. El estado exit indica que, una vez alcanzado, la máquina se terminará o apagará. Y el estado Any State indica que esas transiciones a los estados die y hurt pueden hacerse desde cualquier otro estado.

5.4 Mecánicas básicas de combate del protagonista

Nuestro héroe debe ser capaz de pelear y luchar para sobrevivir en ese mundo hostil. Por lo tanto, deberá tener un sistema de combate.

Tal como se comentó en el apartado de Diseño Detallado, las mecánicas de combate se realizarán mediante la implementación de tres métodos, uno para cada tipo de arma. Estos métodos se llamarán SwordAttack, GunAttack y MazeAttack y realizarán las acciones de ataque de espada, disparo del pistola y golpe de maza, respectivamente.

Estas funciones hacen uso de las mismas características que las que hemos observado en los enemigos. Usan Instantiate para los golpes a distancia (proyectiles) con la pistola láser y usan Physics2D.OverlapCircleAll para los golpes de cerca.

La animación de combate y movimiento del protagonista se basan en una máquina de estados no muy compleja que se mostrará a continuación:

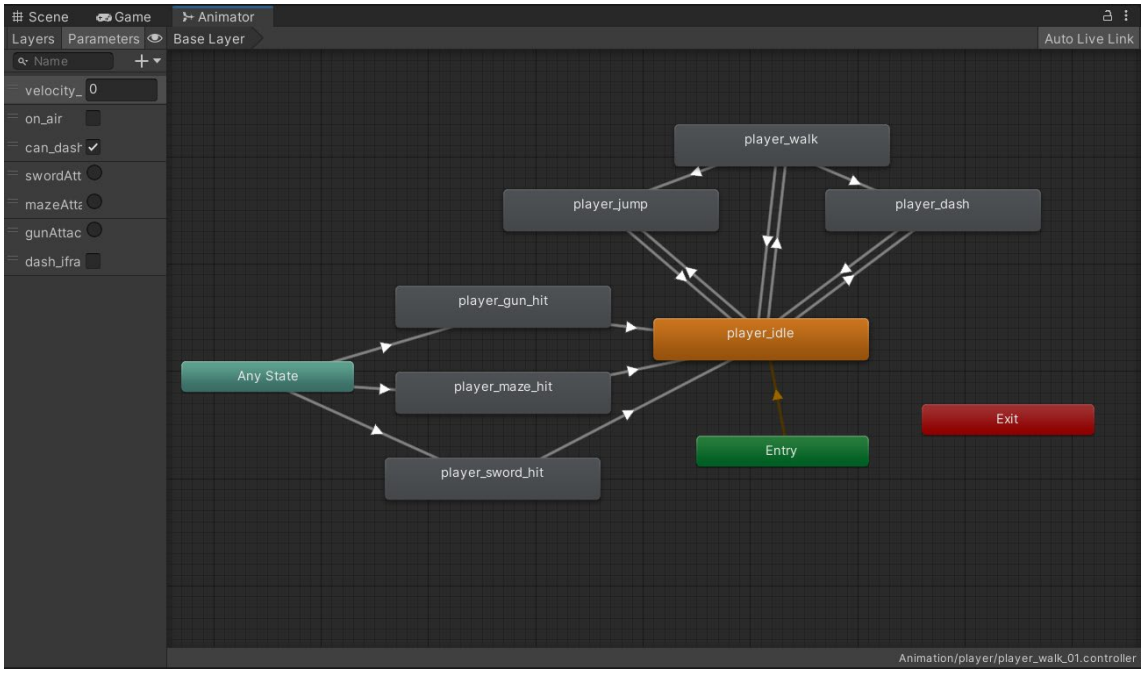


Figura 34. Máquina de estados del protagonista

Y el salto mejorado se realizará mediante el uso de un método jump, que contendrá Coyote Time y Time Buffering, haciendo que el juego sea más suave y fluido a la hora de usar los controles. También permite hacer saltos más pequeños, dependiendo de cuándo el usuario suelte la tecla de salto. Se mostrará el código a continuación:

```
void Jump()
{
    if (grounded) { coyoteTimeCounter = coyoteTime; }       //Coyote time
    else { coyoteTimeCounter -= Time.deltaTime; }

    if (Input.GetButtonDown("Jump")) { jumpBufferTimeCounter = jumpBufferTime; }    //Jump buffer
    else { jumpBufferTimeCounter -= Time.deltaTime; }

    if (jumpBufferTimeCounter > 0f && coyoteTimeCounter > 0f) // comprueba que puede saltar
    {
        _rigidbody2D.AddForce(Vector2.up * JUMP_FORCE * Mathf.Sqrt(JumpCap), ForceMode2D.Impulse); // impulsa al personaje hacia arriba
        _audioSource.PlayAudioOneShot(6);

        jumpBufferTimeCounter = 0f;
    }

    if (Input.GetButtonUp("Jump") && _rigidbody2D.velocity.y > 0f)      // Saltos variables
    {
        _rigidbody2D.velocity = new Vector2(_rigidbody2D.velocity.x, _rigidbody2D.velocity.y * 0.7f);

        coyoteTimeCounter = 0f;
    }

    _animator.SetBool("on_air", onAir);
}
```

Figura 35. Código de la implementación de Coyote Time, Time Buffering y saltos variables

Esta imagen representa el código necesario para la función de salto del personaje principal. Consta de 4 condiciones, que determinarán si se puede realizar el salto y de cómo será este salto.

Los dos primeros ifs (condiciones) son los encargados de registrar información sobre si el salto puede ser efectivo o no. Las dos segundas condiciones comprobarán si este salto finalmente es efectivo, y de la duración y altura de este.

5.5 Desarrollo artístico del HUD

El HUD, es un elemento artístico que influye en la percepción del videojuego. Por lo tanto, es necesario que cuadre con la temática y tenga una calidad suficiente, para que no se pixele con el cambio de dispositivos o de resoluciones.

Figura 36. Vista del HUD en el videojuego Neyteria

5.6 Sistema de sonido

El sistema de sonido se basará en un Gameobject empty, que seguirá al player, y que tiene un script donde se define la gestión y reproducción de sonido en forma de funciones públicas accesibles por los demás objetos del juego:

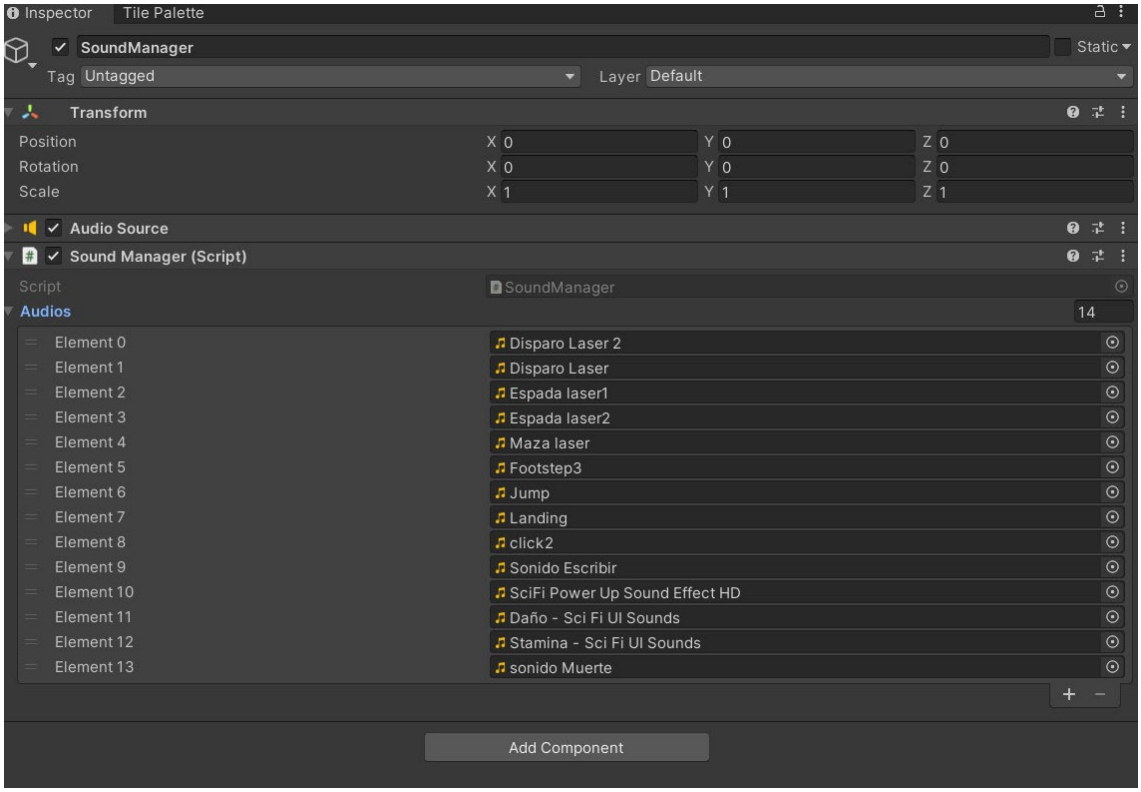


Figura 37. Sistema de Sonido del Videojuego Neyteria

5.7 Menús del videojuego

Los menús del videojuego son una parte artística importante, dado que son la primera impresión que se lleva el jugador de nuestro proyecto. Por lo tanto, deben ser funcionales y estéticos, de acuerdo a la temática del juego.

A continuación, veremos el resultado final de los distintos menús del juego así como de la pantalla de muerte:

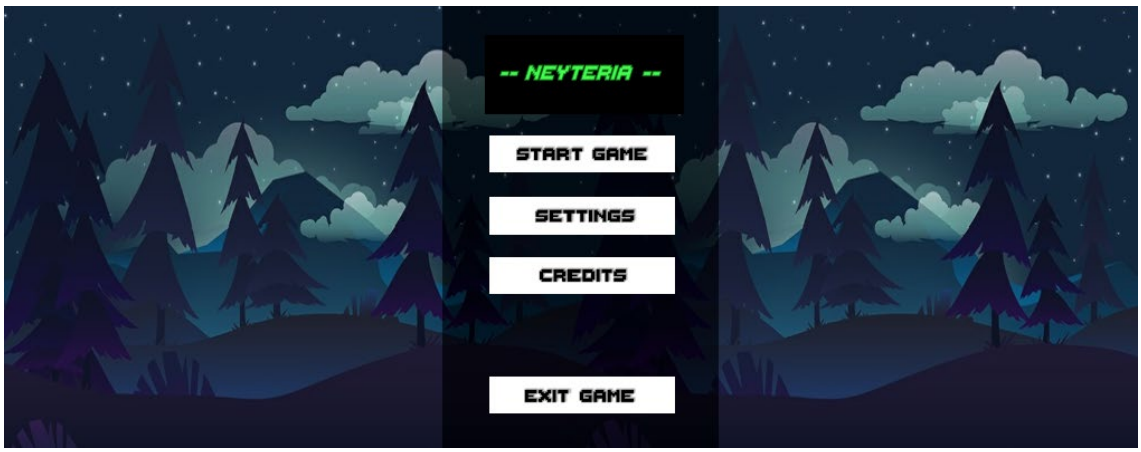


Figura 38. Menú principal del videojuego Neyteria

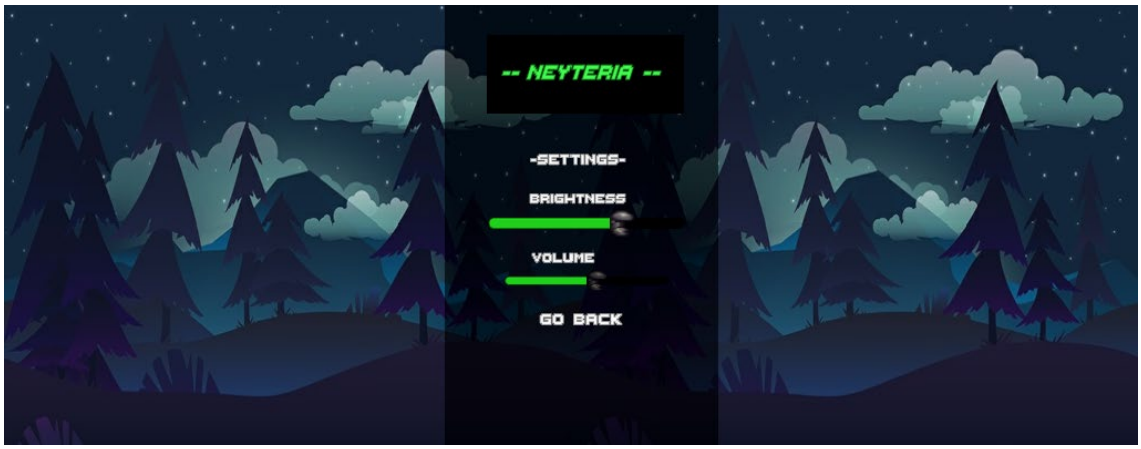


Figura 39. Submenú Principal Opciones del videojuego Neyteria

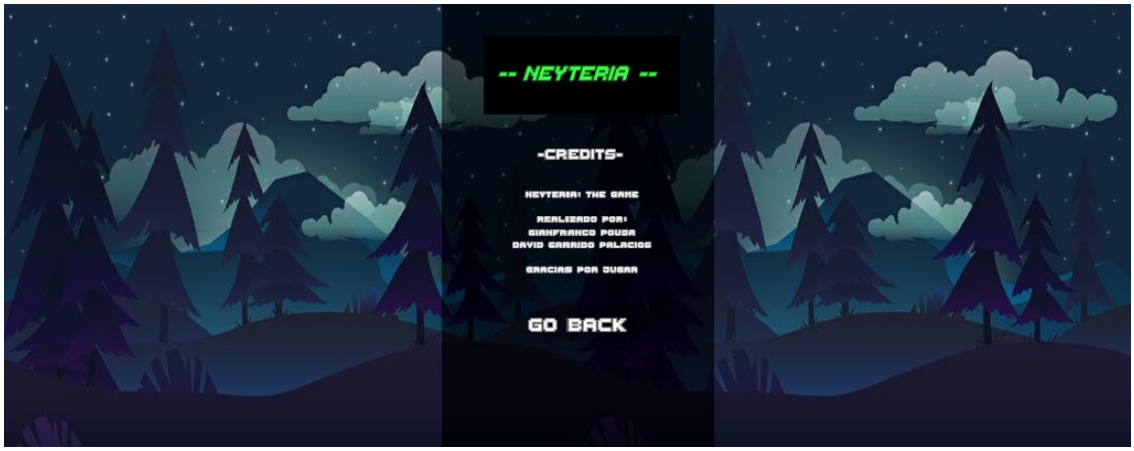


Figura 40. Submenú Principal Créditos del videojuego Neyteria

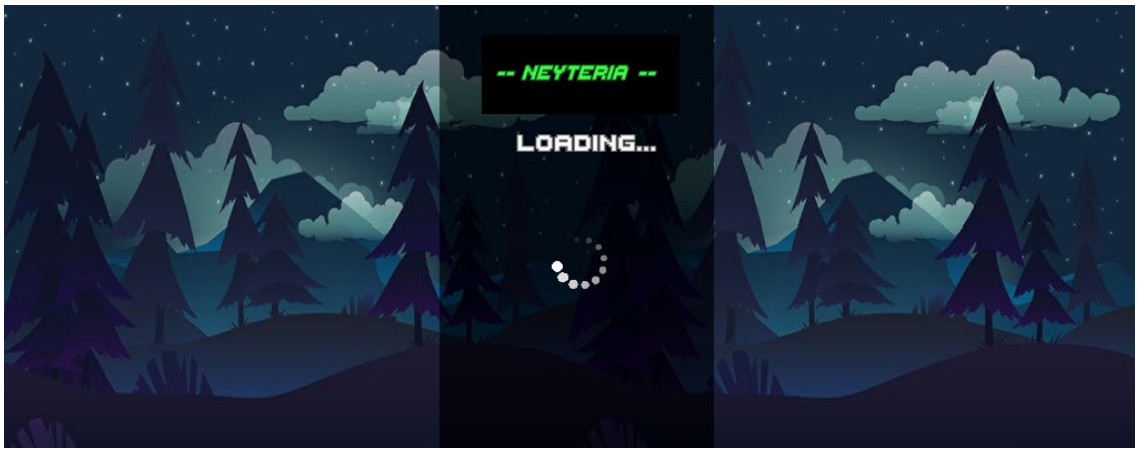


Figura 41. Submenú Principal Cargando del videojuego Neyteria

A continuación, veremos el menú de pausa:

Figura 42. Menú de Pausa del videojuego Neyteria

A continuación, veremos la pantalla de muerte:

Figura 43. Pantalla de Muerte del videojuego Neyteria

5.8 Sistema de diálogo e historia

El sistema de diálogo está implementado mediante un empty DialogueManager que tiene un script con el mismo nombre, y es el que se encarga de escribir el texto en la caja de diálogo, así como esconder o recuperar la caja de diálogo cuando sea necesario.

DialogueBox es el canvas que contiene las imágenes y textos pertenecientes a la caja de diálogo.

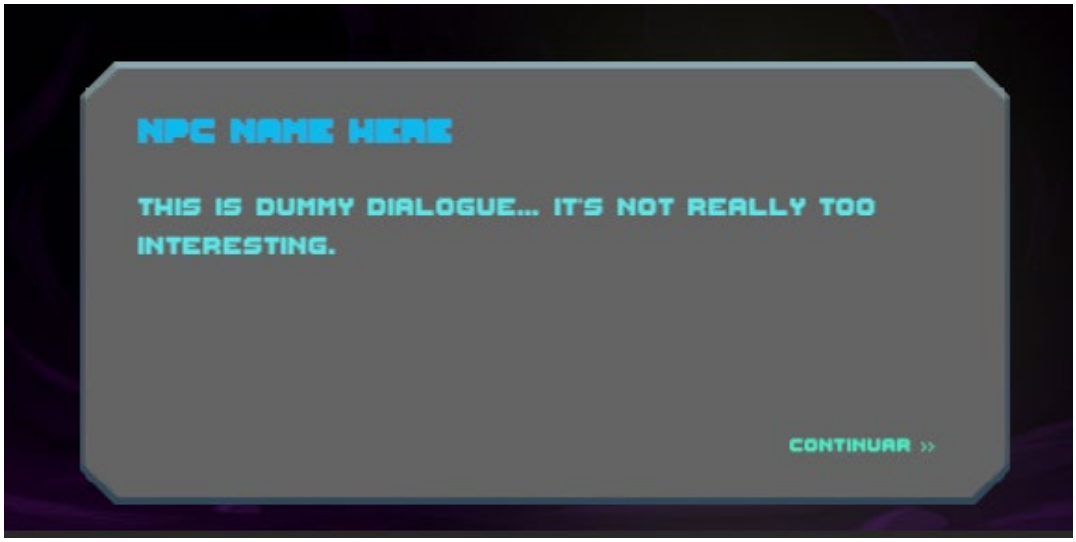


Figura 44. Sistema de diálogo del videojuego Neyteria

5.9 Organización y fases del plan

En este apartado se mostrará información sobre la organización y el tiempo usado en el desarrollo de la solución (fases del plan).

Recordamos que las fases eran:

- Diseño de niveles
- Mecánicas básicas personaje y animación del mismo
- Enemigos
- Elementos y eventos para puzles del juego
- Menús e Interfaces
- Miscelánea (Donde entran cosas como el desarrollo de la historia, el sistema de diálogo, selección de fuentes del juego)
- Selección de sprites, tiles y sonidos del juego

A continuación, veremos capturas de pantalla de la aplicación HacknPlan para observar cómo se ha llevado a cabo el desarrollo del plan.

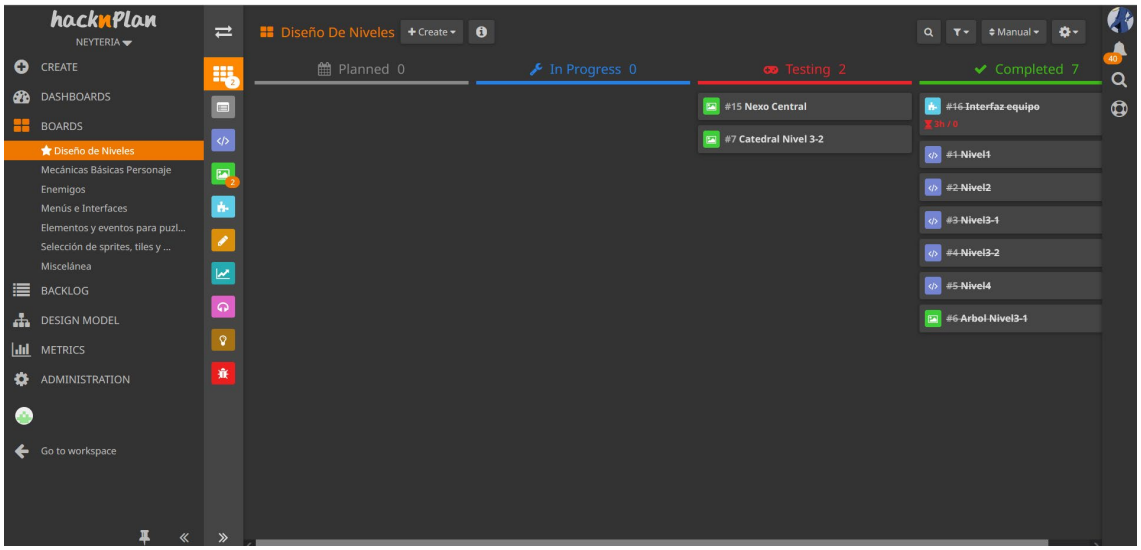


Figura 45. HacknPlan Diseño de niveles del videojuego Neyteria

Aquí observamos unas listas de tares a realizar. Estas tareas se colocan en las diferentes listas, las cuales significan, por orden de izquierda a derecha, que la tarea ha sido creada y planeada, que la tarea está siendo desarrollada, que la tarea está en fase de pruebas, y finalmente que la tarea ha sido completada correctamente.

En las distintas imágenes de HacknPlan se observarán estas listas de cada una de las fases propuestas, y mostrarán cómo han sido o están siendo completadas.

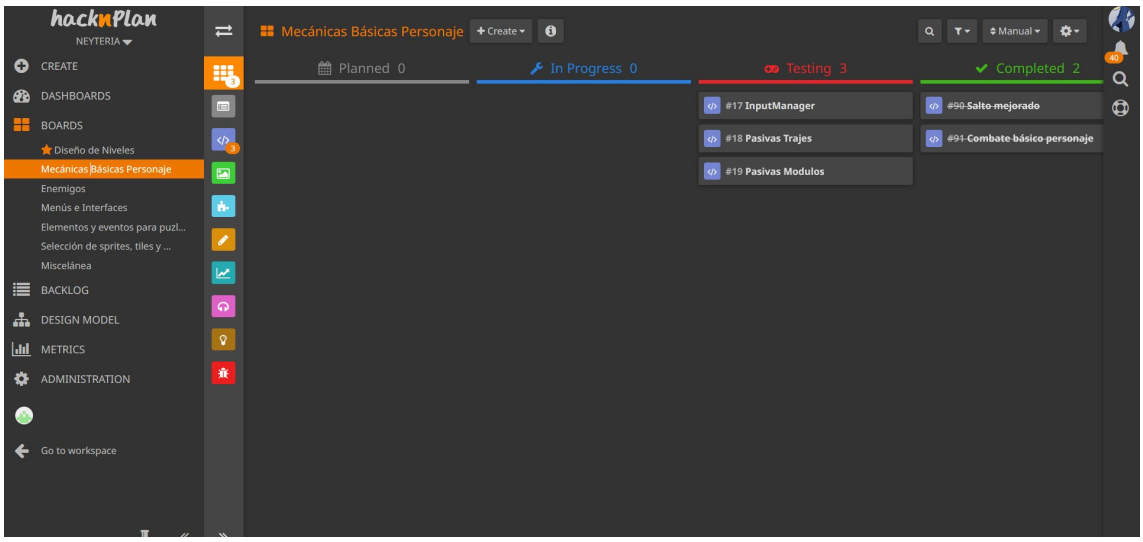


Figura 46. HacknPlan Mecánicas Base Personaje del videojuego Neyteria

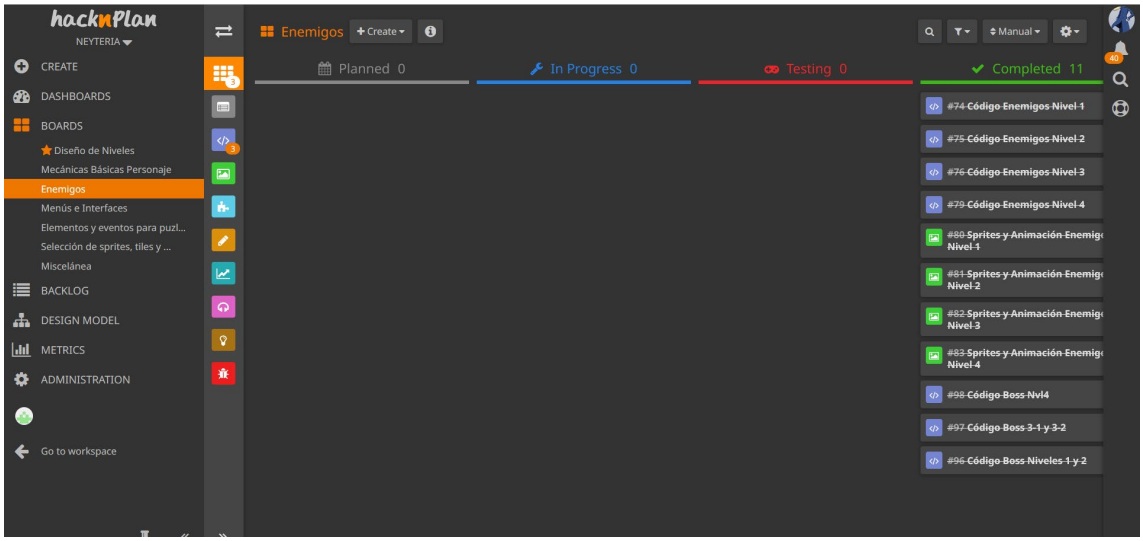


Figura 47. HacknPlan Enemigos del videojuego Neyteria

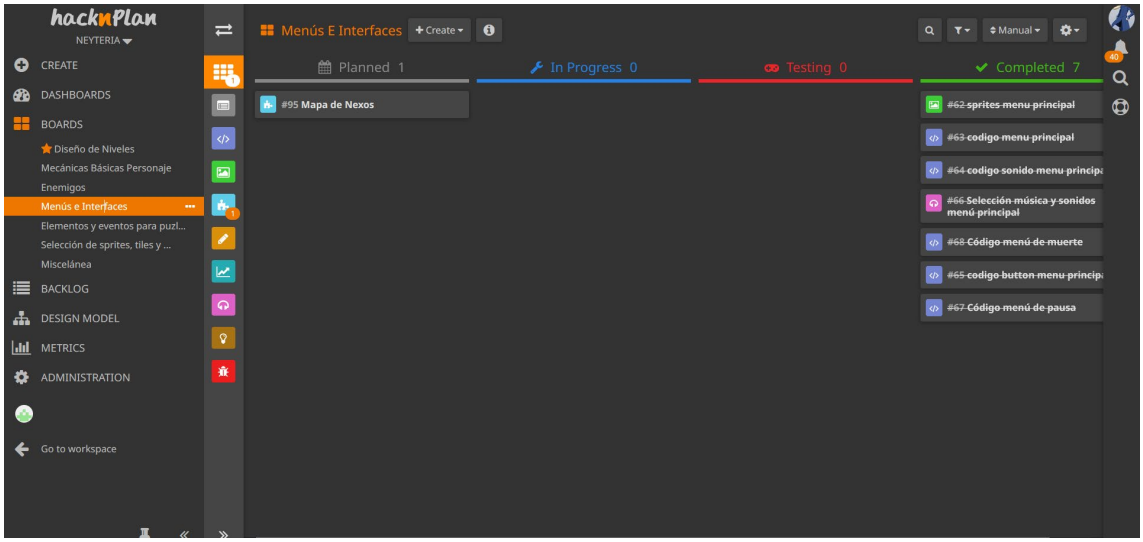


Figura 48. HacknPlan Menús e Interfaces del videojuego Neyteria

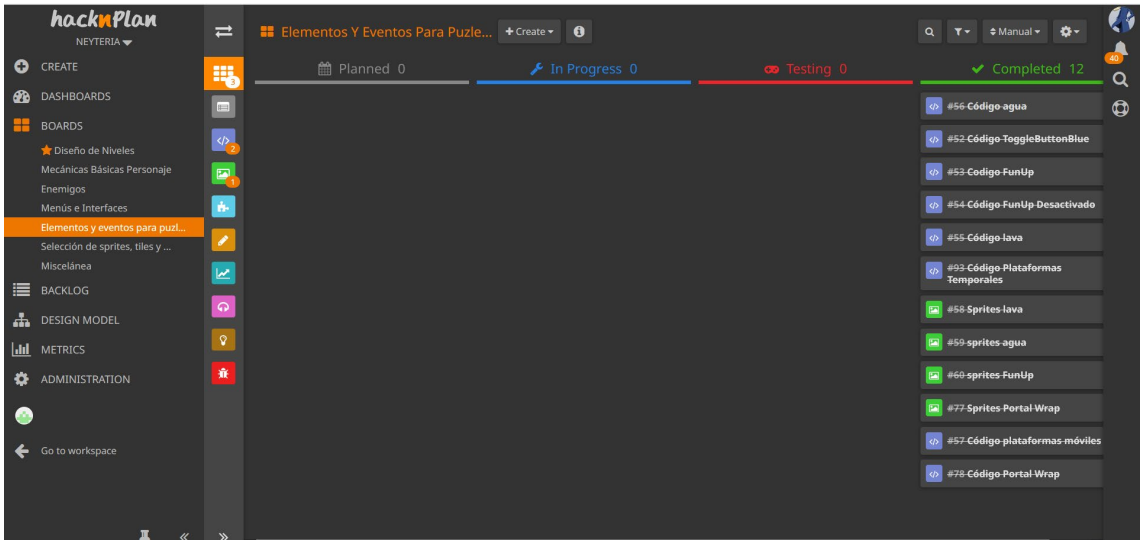


Figura 49. HacknPlan Elementos Puzles del videojuego Neyteria

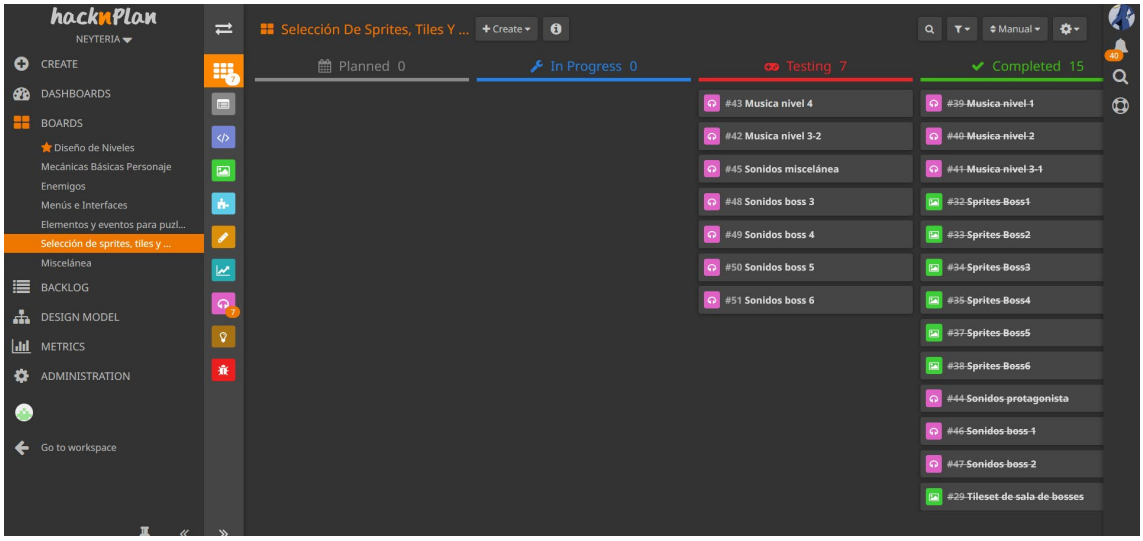


Figura 50. HacknPlan Selección de Sprites, Tiles y sonidos del videojuego Neyteria

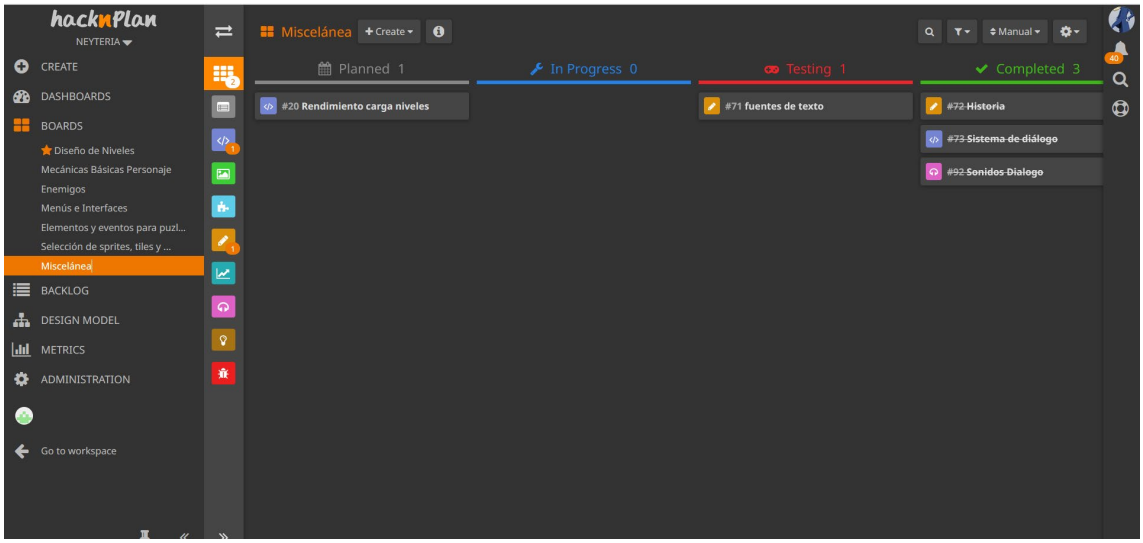


Figura 51. HacknPlan Miscelánea del videojuego Neyteria

Como se puede observar, las fases están casi completadas. Solo falta finalizar las pruebas de los elementos sonoros y de un par de scripts y de sprites de la fase de niveles y de la fase de movimiento básico del personaje.

CAPÍTULO 6

Implantación

6.1 Puesta en marcha del Sistema desarrollado

Para la puesta en marcha del sistema desarrollado en Unity se puede probar directamente desde el editor. Pero no es esto lo que se busca en este capítulo, sino la explotación del sistema en un caso real, para poder probar y obtener resultados fiables de la solución final del proyecto.

Por suerte, Unity permite la creación del build de un proyecto de una manera sencilla.

Además, cualquier persona que quiera probar la build en su ordenador no necesitará tener las aplicaciones que hemos usado para el desarrollo de este proyecto.

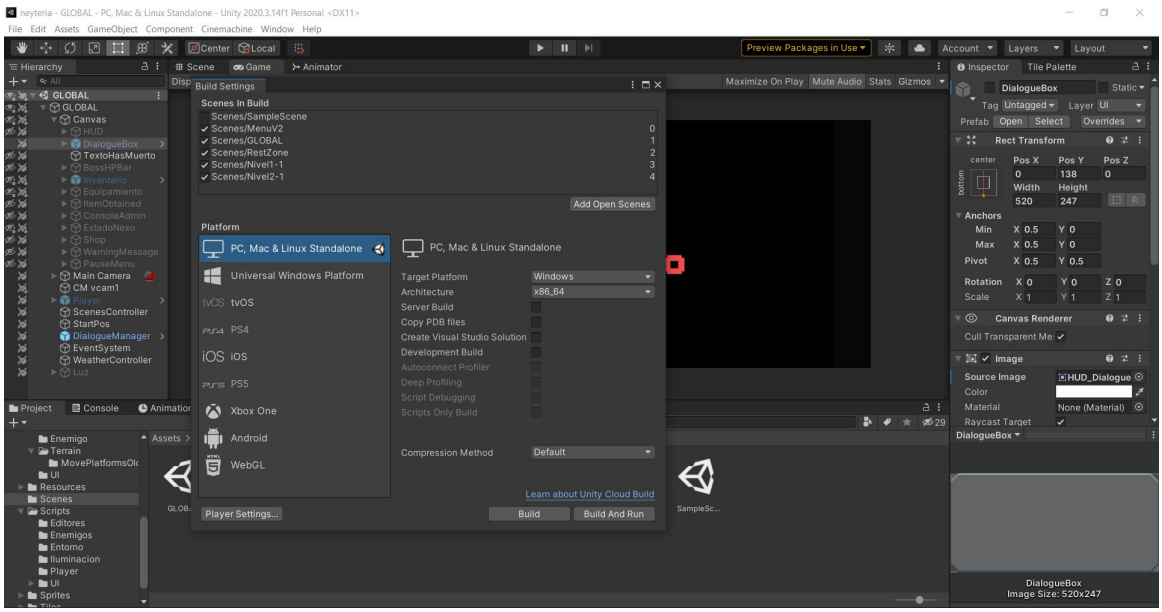


Figura 52. Creación de la build del videojuego Neyteria

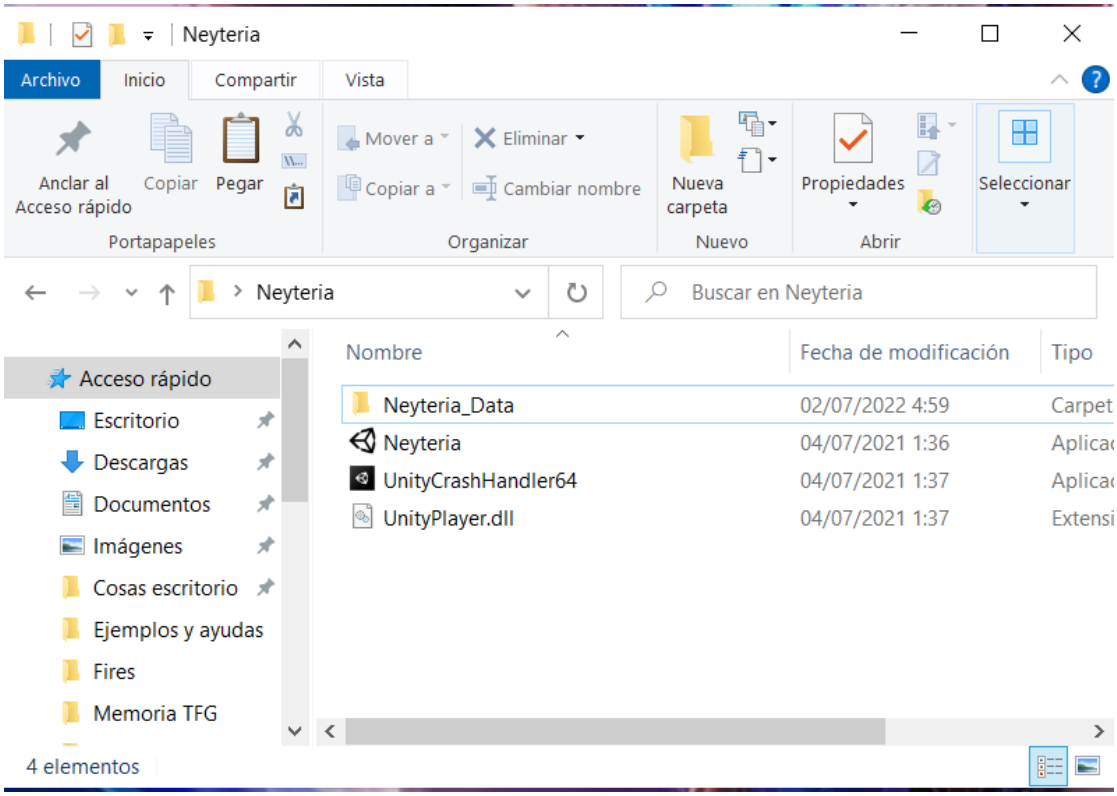


Figura 53. Build del videojuego Neyteria

La Figura 51 muestra el resultado en el explorador de archivos de una Build de Neyteria. El directorio contiene los ficheros necesarios para la ejecución del Videojuego, junto a su ejecutable.

CAPÍTULO 7

Pruebas

7.1 Pruebas y testing

Los requisitos implementados se testeaban usando pruebas manuales que eran ejecutadas directamente por los desarrolladores, simulando las acciones del usuario final del videojuego. Mediante la creación de escenas específicas, y gracias al sistema de comandos elaborado por Gianfranco, ha sido facilitada la creación y proceso las pruebas.

En las pruebas se han tratado aspectos como el correcto movimiento del personaje principal, y sus mecánicas de combate y de salud. Estas pruebas fueron realizadas mediante la creación de una escena con plataformas básicas y un par de enemigos, llamada pruebas:

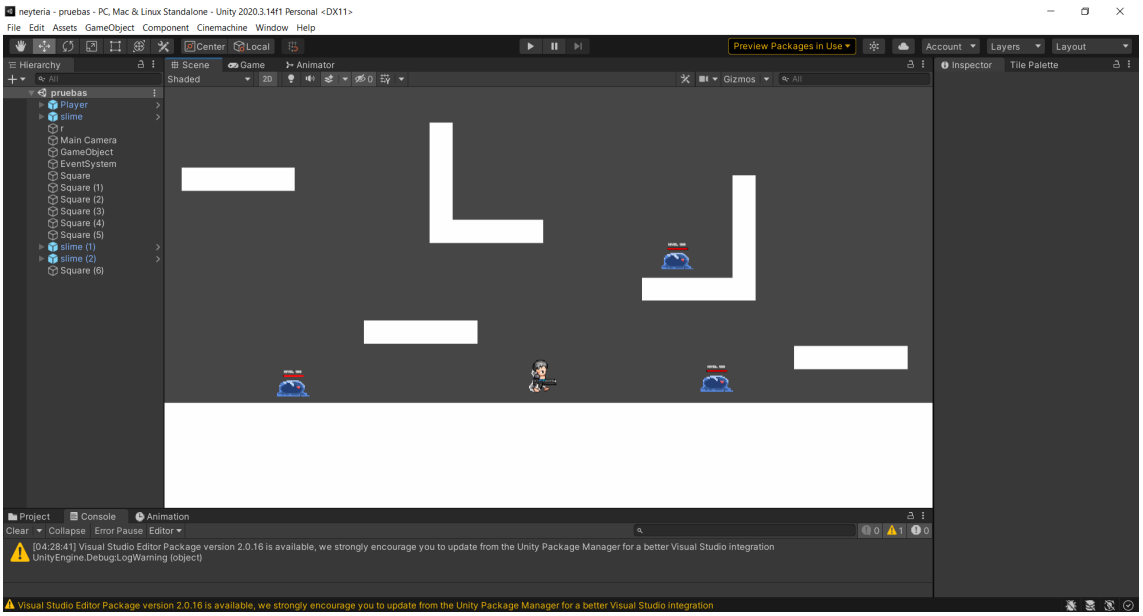


Figura 54. Escena pruebas para mecánicas básicas del personaje

Para comprobar el correcto funcionamiento de los enemigos, las pruebas de movimiento, de combate y salud se realizaron en la escena de pruebas pruebas 1:

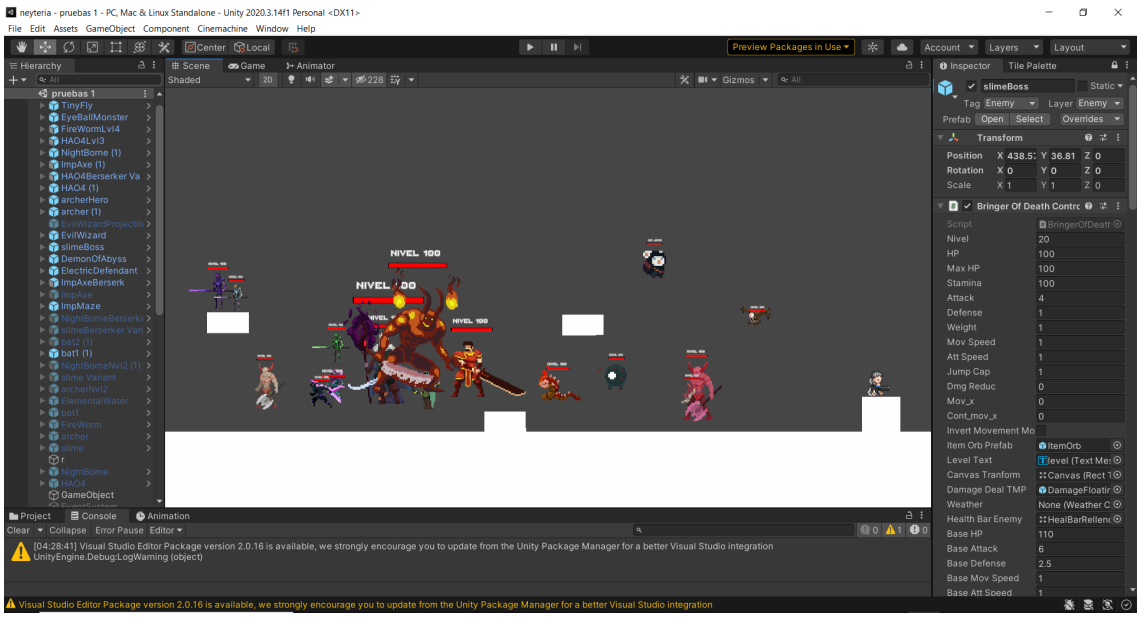


Figura 55. Escena pruebas 1 para mecánicas de enemigos

En esta escena se han ido instanciando los distintos enemigos del juego. Primero se han comprobado individualmente, y luego en grupo. De tal manera primero no aseguramos de que el enemigo se comporte correctamente hacia el protagonista, y después se comprueba que su comportamiento sigue siendo correcto junto a otros enemigos (los enemigos no deben bloquearse el paso ni causarse daño entre ellos).

Las pruebas para el funcionamiento de los niveles se han basado en el uso del protagonista a través de la escena del nivel correspondiente, recorriéndola de un lado a otro e interactuando con todos los elementos del nivel (primero se realiza la prueba sin enemigos. Después se añaden y se vuelve a realizar la prueba).

La prueba del funcionamiento del menú ha sido la simple ejecución de la escena correspondiente al menú. De este modo se ha comprobado las funcionalidades de transición entre menús, y de carga y descarga de niveles.

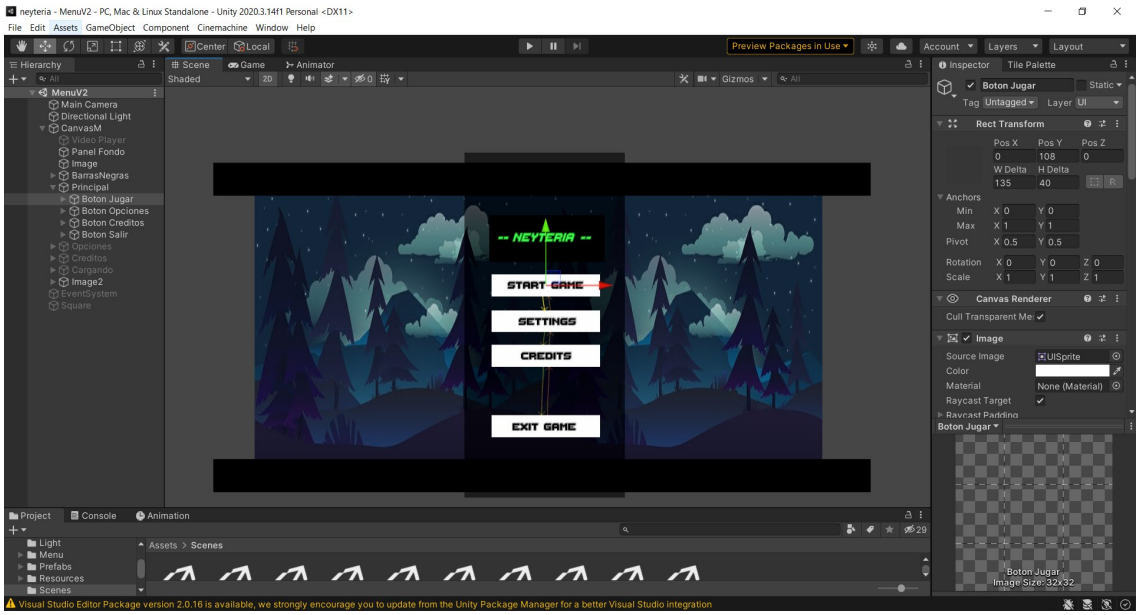


Figura 56. Escena MenuV2 para prueba de funcionamiento de menús

Por último, como prueba final, se ha realizado la ejecución del videojuego para detectar defectos y errores que pudiesen ser causados o encontrados solo en ambientes reales, revisando además, que se cumple el correcto funcionamiento de los requisitos funcionales y no funcionales definidos en la sección de Definición de requisitos.

7.2 Encuestas a usuarios

Una manera de obtener información y feedback sobre el juego es preguntando a usuarios, que prueben el videojuego y contesten a una encuesta y/o unas preguntas.

Es buena idea enseñar el videojuego a amigos y/o familiares, pero siempre que tengan cierta competencia en la materia.

Se realizaron una serie de preguntas a amigos, tras haber completado más de la mitad del primer nivel del juego. La encuesta fue realizada por un total de ocho jugadores. Sus respuestas serán analizadas a continuación:

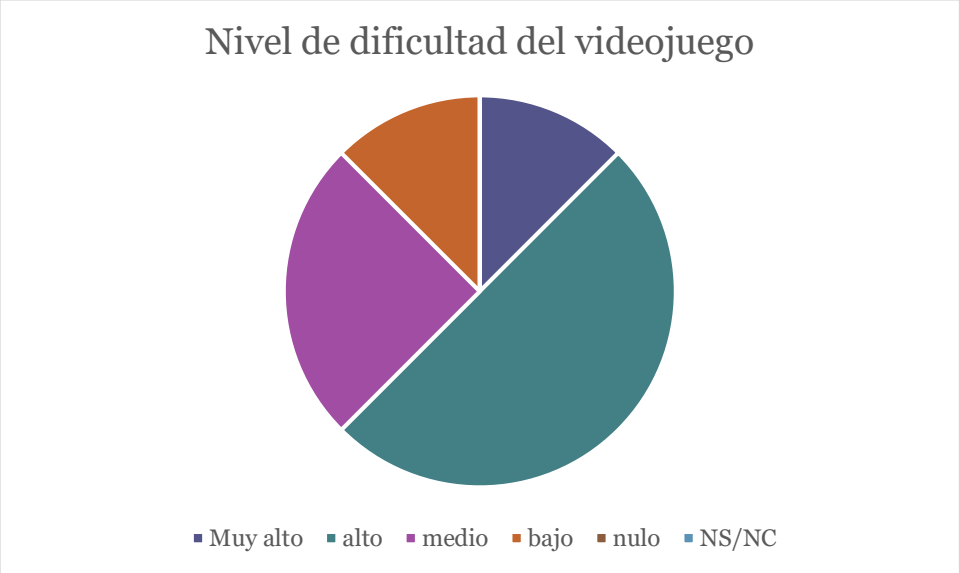


Figura 57. Resultado de la encuesta sobre la dificultad

El videojuego cuenta con un nivel de dificultad elevado, y eso ha podido verse en las encuestas. Se puede observar, también, que depende de la maestría del usuario puede parecerle más sencillo o más difícil, pero la media conseguida es la esperada.

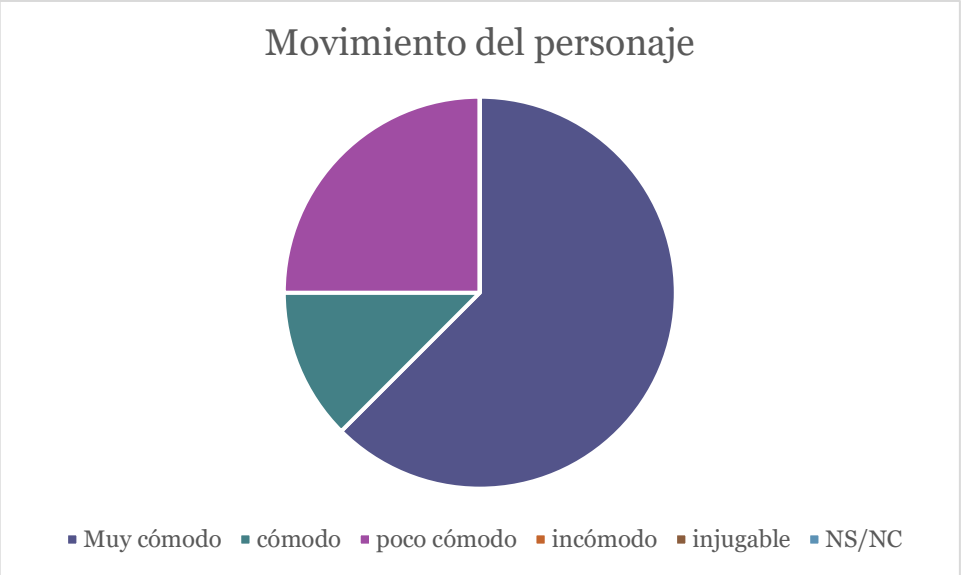


Figura 58. Resultado de la encuesta sobre el movimiento el personaje

El movimiento del personaje está realizado usando técnicas para mejorar el control y la fluidez del personaje. Estas mejoras muestran que efectivamente se ha conseguido la comodidad buscada para este juego.

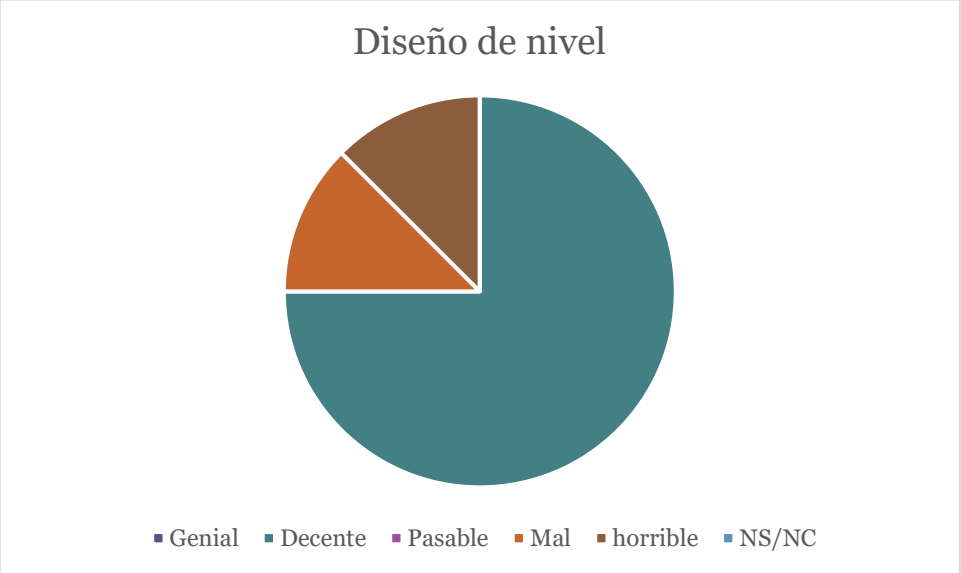


Figura 59. Resultado de la encuesta sobre la calidad de diseño de nivel

El diseño de niveles parece haber sido implementado correctamente, aunque para jugadores experimentados puede parecer demasiado simple o homogéneo.

Figura 60. Resultado de la encuesta sobre el entretenimiento del juego

El entretenimiento del juego parece tener el aprobado, pero dado a la dificultad que representa el juego, solo los jugadores con buen dominio de los controles parecen disfrutar plenamente del juego. Este era un resultado esperable, dado que este juego está dirigido a este tipo de jugadores.

Además, hemos subido el proyecto a una plataforma de juegos indie, itch.io, donde es posible que alguien se lo descargue y deje una reseña constructiva.

CAPÍTULO 8

Conclusiones

En este trabajo hemos realizado la creación de un videojuego, funcional, entretenido y original. Hemos conseguido completar todos los objetivos formulados inicialmente, por lo cual deberíamos estar muy satisfechos con el trabajo hecho.

Sin embargo, la realización de los niveles ha sido bastante dura y seguramente se pueda conseguir una forma de realizarlos de forma más sencilla y eficiente.

En cambio, la realización de los objetivos relacionados a los enemigos ha sido bastante fluida y divertida. El uso de la herencia y de los prefabs han propiciado una base para poder hacer modificaciones útiles e interesantes, de manera que se ha ahorrado trabajo y se ha podido conseguir aún más variedad para el proyecto.

El trabajo en equipo ha sido clave para el desarrollo del proyecto. Gracias a la confianza y cooperación que hemos tenido los alumnos, el proyecto ha sido muy ameno y entretenido. El número de reuniones y de entregables han propiciado una constancia en el proyecto, de manera que se ha podido realizar de forma agradable.

8.1 Relación del trabajo con los estudios cursados

El uso de distintas tecnologías, de distintos propósitos, y de lenguajes variados, nos han propiciado una ventaja a la hora de realizar el proyecto, permitiendo que pudiésemos ser capaces de crear proyectos de software de cierta complejidad.

Sin ninguna duda, cumplir estos objetivos ha sido gracias a asignaturas cursadas a lo largo de la carrera. De todos modos, especificaremos las siguientes asignaturas:

- Interfaces Persona Computador, cuyo tema 4 ha sido de gran ayuda a la hora de la especificación de requisitos.
- Ingeniería del Software, donde nos enseñaron los ciclos de vida del software y pruebas del mismo.
- Gestión de Proyectos, ya que nos dieron una guía y dirección en el área de proyectos y toma de decisiones en la organización.

- La Optativa de Desarrollo de Videojuegos 3D ha sido muy útil a la hora de realizar este TFG, ya que nos enseñaron Unity y C#.
- La optativa de Desarrollo de Videojuegos 2D, que nos ha dejado abierto el camino para la continuación del proyecto y por lo tanto la realización de este TFG.

Por último, nos han sido especialmente útiles las siguientes competencias transversales:

- Innovación, creatividad y emprendimiento, gracias al desempeño e ímpetu de crear un proyecto original con múltiples herramientas de vanguardia.
- Análisis y resolución de problemas, donde ha sido necesario crear soluciones efectivas a los objetivos propuestos de dicho proyecto.
- Diseño y proyecto, para la correcta ejecución y proyección del trabajo.
- Trabajo en equipo y liderazgo, por la coordinación y capacidad de cooperación en grupos heterogéneos.

CAPÍTULO 9

Trabajos futuros

A lo largo del trabajo hemos ido cubriendo objetivos, pero dejando de lado otros también importantes por el camino.

Este proyecto tiene un gran potencial para ser ampliado y para añadirle nuevas mejoras.

Al ser un videojuego plataformas, acción y aventuras es relativamente sencillo pensar en nuevas ideas para el futuro. Desde nuevas mecánicas, nuevas habilidades, objetos, armas, módulos… E incluso nuevos modos de juego.

Probablemente un modo de juego cooperativo ofrecería a este juego un toque más maduro y familiar, dando así un empujón dentro del mundo indie, ya que el modo cooperativo no está muy explotado aún en el mercado.

Las temáticas del juego han estado bastante bien, pero podría haber sido mejor con más tipos distintos de Tiles y más variedad de sprites para hacer decoración en los niveles.

Los elementos de nivel y de puzles, siempre pueden ser más rebuscados y entretenidos. Una selección adecuada de estos podría darle un toque más original a este tipo de juegos.

Se podrían implementar mecánicas interesantes a los enemigos, como bloqueos y curaciones, o incluso añadir probabilidades para que salgan golpes críticos aleatoriamente.

Sería muy beneficioso contactar con un artista para que diseñe assets específicos, de esa manera no gastaríamos tanto tiempo en la búsqueda de estos elementos y, además, el arte mejoraría de forma general la calidad del juego, permitiendo tener más sprites y tiles acordes a la temática del juego.

CAPÍTULO 10

Referencias

[1] Frasca, G. (2009). Juego, videojuego y creación de sentido. Una introducción. Comunicación: revista Internacional de Comunicación Audiovisual, Publicidad y Estudios Culturales, 1 (7), 37-44. http://hdl.handle.net/11441/58039

[2] José Luis Eguia Gómez et al. (2013, marzo). Videojuegos: conceptos, historia y su potencial como herramientas para la educación. Revista de investigación. Editada por Área de Innovación y Desarrollo, S.L. https://www.3ciencias.com/wp-content/uploads/2013/04/videojuegos.pdf

[3] Colaboradores de Wikipedia. (2022, 1 abril). Videojuego independiente. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Videojuego_independiente

[4] Diana Díaz Montón. *La traducción amateur de videojuegos al español* (2011). TRANS. Revista de Traductología. (N.º 15). http://www.trans.uma.es/pdf/Trans_15/69-82.pdf

[5] de Zayas, R. (2021, 5 febrero). 5 + 1 razones para traducir tu videojuego profesionalmente. Tatutrad. https://tatutrad.net/5-1-razones-para-traducir-tu-videojuego-con-profesionales/

[6] Trigás Gallego, M. (2012). Metodologia scrum. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

[7] Mike Croucher. (2020). Academic Benefits of Using git and GitHub - Walking Randomly. https://walkingrandomly.com/?p=6653

[8] Colaboradores de Wikipedia (2022) - Guía de los fundamentos para la dirección de proyectos PMBOK. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Gu%C3%ADa_de_los_fundamentos_para_la_direcci%C3%B3n_de_proyectos

[9] Belli, S., & Raventós, C. L. (2008). Breve historia de los videojuegos. *Athenea Digital. Revista de pensamiento e investigación social*, (14), 159-179. https://www.redalyc.org/pdf/537/53701409.pdf

[10] Plarium (2022). Juegos Metroidvania: todo lo que debes saber. https://plarium.com/es/blog/metroidvania-games/

[11] AltexSoft (November, 2019): Non-functional Requirements: Examples, Types, How to Approach. https://www.altexsoft.com/blog/non-functional-requirements/

[12] Adam (october, 2021). Functional Requirements in Software Engineering: Top Tips for Writing a Clear Document. https://studiosoftware.com/blog/functional-requirements-in-software-engineering-how-to-improve-project-performance/

[13] Luis Matamala Belinchón (2021) ¿Qué son los Stakeholders y cómo se gestionan en los proyectos? IMF Smart Education. https://blogs.imf-formacion.com/blog/mba/que-son-los-stakeholders/

[14] Christian Leis (2022) Qué es y Cómo hacer Brainstorming. https://aulacm.com/guia-hacer-brainstorming-generar-ideas-creativas/

[15] Milena González (2022) Análisis DAFO: Guía con Ejemplos https://aulacm.com/analisis-dafo-ejemplo-plantilla/

[16] Unity (2021) - Manual: Prefabs - Unity Technologies https://docs.unity3d.com/es/530/Manual/Prefabs.html

[17] Unity (2021) - Manual: Prefabs Variants - Unity Technologies https://docs.unity3d.com/Manual/PrefabVariants.html

[18] The best place for answers about Unity - Unity Answers https://answers.unity.com/index.html

[19] Yúbal Fernández (2019). Qué es Github y qué es lo que le ofrece a los desarrolladores. Xataka. https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

[20] O G Glazunova et al 2021 J. Phys.: Conf. Ser. 1840 012030 - The effectiveness of GitHub cloud services for implementing a programming training project: students' point of view. https://iopscience.iop.org/article/10.1088/1742-6596/1840/1/012030/pdf

[21] David Herron (2018) - What are the differences between Apache and GitHub? – Quora. https://www.quora.com/What-are-the-differences-between-Apache-and-GitHub-Is-it-possible-that-Apache-will-also-get-acquired-by-Microsoft-like-GitHub

Anexo 1

Licencias

A continuación, se mostrará una serie de referencias a creadores de sprites, tiles y sonidos usados en el proyecto.

Créditos a chierit por sprites para bosses
https://itch.io/profile/chierit

Créditos a Clembod por sprites para bosses
https://itch.io/profile/clembod

Créditos a LuizMelo por sprites para bosses
https://itch.io/profile/luizmelo

Créditos a kronovi por sprites para enemigos
https://itch.io/profile/darkpixel-kronovi

Créditos a LuizMelo por sprites para enemigos
https://itch.io/profile/luizmelo

Créditos a CreativeKind por sprites para enemigos
https://itch.io/profile/creativekind

Créditos a Szadi art. por sprites para enemigos
https://itch.io/profile/szadiart

Créditos a AstroBob por sprites para enemigos
https://itch.io/profile/astrobob

Créditos a sanctumpixel por sprites para enemigos
https://itch.io/profile/sanctumpixel

Créditos a penusbmic por sprites para enemigos
https://itch.io/profile/penusbmic

Créditos a Szadi art. por tiles para decoración
https://itch.io/profile/szadiart

Créditos a CreativeKind por tiles para decoración
https://itch.io/profile/creativekind

Créditos a Anokolisa por tiles para diseño de niveles
https://itch.io/profile/anokolisa

Créditos a Pipoya por sprites para elementos del nivel
https://itch.io/profile/pipoya

Créditos a CreativeKind por sprites para elementos del nivel
https://itch.io/profile/creativekind

Créditos a Tiny Worlds por la fuente de texto
https://itch.io/profile/tinyworlds

Créditos a adwitr por sprites para el Hud
https://itch.io/profile/adwitr

Créditos a wubs por el generador de sonidos
https://itch.io/profile/wubs

A continuación, se mostrará una serie de referencias de las aplicaciones usadas en el proyecto.

Licencia de Unity y UnityHub:
Se dispone una licencia "Unity Personal" para la realización del proyecto Neyteria, la cual es gratis siempre que se cumplan los términos de la licencia.
https://unity3d.com/es/unity/activation/personal

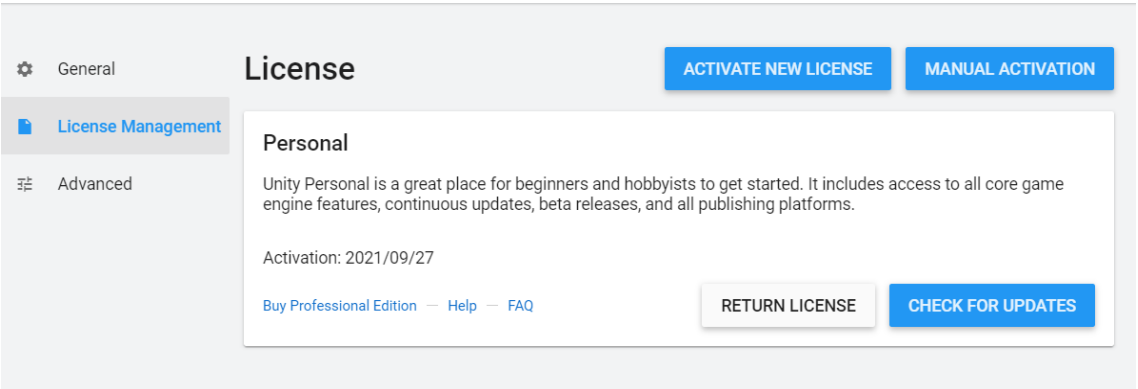


Figura 61. Licencia de Unity usada para el proyecto Neyteria

Licencia de HacknPlan:
La aplicación de HacknPlan es gratis siempre que se cumplan los Términos y condiciones. https://hacknplan.com/terms-and-conditions/

Licencia de Visual Studio Community:
La aplicación de Visual Studio Community es gratis haciendo uso de la versión personal, y cumpliendo los términos de licencia https://visualstudio.microsoft.com/es/free-developer-offers/

Licencia de GitHub:
La aplicación de GitHub es gratis haciendo uso de la versión personal, y cumpliendo los términos de licencia https://docs.github.com/en/site-policy/github-terms/github-terms-of-service#b-account-terms

Licencia de GitHub:
La aplicación de GitHub es gratis y cumpliendo los términos de licencia, Freeware, software propietario

Licencia de Microsoft Teams:
Se dispone de la licencia de alumno obtenida gracias a nuestra institución académica, la UPV, y respetando los términos que indica la licencia. https://www.microsoft.com/es-es/microsoft-365/academic/compare-office-365-education-plans?activetab=tab:primaryr1

Anexo 2

Objetivos de Desarrollo Sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**			**X**	
ODS 4. **Educación de calidad.**			**X**	
ODS 5. **Igualdad de género.**			**X**	
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**			**X**	
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**			**X**	
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Tabla – Grado de relación del proyecto con los ODS

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

De los anteriores objetivos de desarrollo sostenibles mencionados, el proyecto relacionado está relacionado con:

- **Salud y bienestar:** la idea presentada promueve la salud y bienestar, dado a que los videojuegos han demostrado mantener y ayudar al estado físico y mental de los jugadores. Les ayuda a mantener el cerebro en funcionamiento y a usarlo de una manera ingeniosa y ágil, previniendo el envejecimiento prematuro de este.

- **Educación de calidad:** el proyecto neyteria es un videojuego, el cuál dispone de ideas y valores que van a ser implícita o explícitamente enseñados al usuario. Por lo tanto, el videojuego ha sido desarrollado de una manera que eduque correcta al jugador y le propicie un crecimiento personal adecuado y bien dirigido.

- **Igualdad de Género:** el sector de los videojuegos está siendo recientemente explorado y explotado por un público femenino, lo cual hace que sea muy importante un correcto diseño y desarrollo de entretenimiento justo e igual ante los distintos géneros. También hacemos una mejor inclusión del género femenino mediante la creación de personajes femeninos en el videojuego.

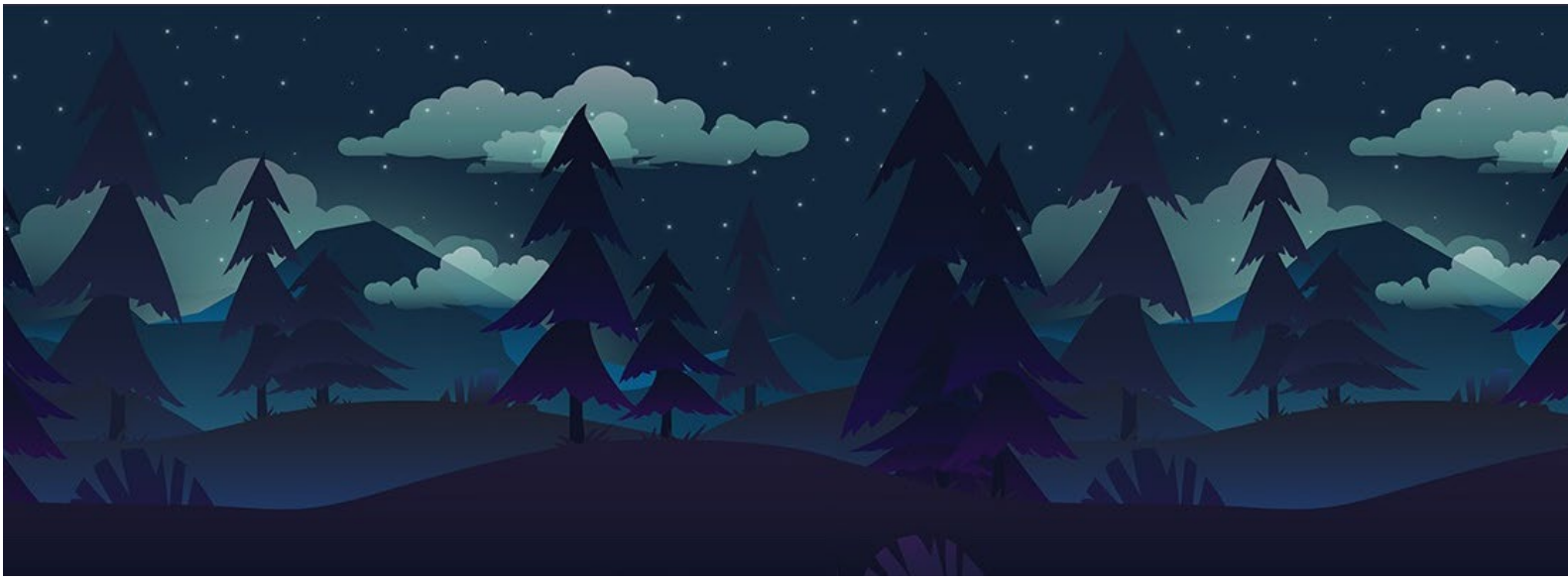
- **Trabajo decente y crecimiento económico:** este proyecto busca promover el trabajo decente y crecimiento económico debido al que el área en el que se encuentra el proyecto es muy conexo y digital. La globalización promueve a que los proyectos digitales u on-line se comuniquen de forma extensa y variada a lo largo de la red. Mediante una organizada elaboración y ejecución, hemos conseguido el objetivo que teníamos para conseguir un trabajo decente. Además, el uso variado de assets públicos mediante licencias libres, ha ayudado al proyecto y su uso en este va a permitir a estos creadores ser más conocidos en el mercado. Como trabajo futuro del proyecto, se puede proponer un precio al proyecto, dando así una remuneración y movimiento económico al mercado.

- **Producción y consumo responsables:** para la realización de este proyecto hemos sido capaces de minimizar el consumo eléctrico y de otros recursos, mediante un uso responsable y efectivo de los materiales y aplicaciones a nuestro alcance. También cabe considerar, que el diseño del juego fue pensado para que se ejecute de forma óptima y efectiva, causando así un menor consumo de recursos que, lo brinda al proyecto un consumo más responsable.

Anexo 3

GDD

Game Design Document



Neyteria

Strange World

Preparado Por:

Gianfranco Pousa

David Garrido

Última actualización:

02/07/22

TABLA DE CONTENIDOS

Análisis del Juego

Mata a tus enemigos y sobrevive para llegar al final en un entorno hostil usando tus capacidades de lucha y herramientas a tu alcance.

Deberás pasar a través de distintos niveles saltando, golpeando, corriendo, esquivando y usando tus habilidades y artefactos, para enfrentar y matar al boss de cada nivel.

Prepárate una buena equipación mediante el loot de los enemigos y obtén grandes recompensas en los puzzles ubicados entre cada nivel.

Objetivo del Juego

Neyteria, un juego de plataformas y acción para PC donde deberás usar tu capacidad, reflejos e inteligencia para poder superar exitosamente todos los niveles, de manera que avances en la historia y llegues al verdadero final.

Géneros

Plataformas, acción, aventura.

Plataformas

PC.

Público Objetivo

Jugadores apasionados de los retos difíciles de acción y plataformas, que estén

acostumbrados a juegos de PC.

Análisis Competitivo

Referentes

Hollow knight, Celeste, Gris, Hyper Light Drifter. Juegos de plataformas donde haya enemigos, en general.

Competencia

Juegos como los nombrados como referentes y juegos de los géneros plataformas, acción y/o aventura.

Características Diferenciadoras

La idea principal es combinar los aspectos básicos de un plataformas en distintos niveles junto con características de acción y metiendo puzzles para ir cambiando el ritmo del juego.

Historia y personajes

Historia

CAPÍTULO I: El alba de un nuevo mundo.

En algún momento desconocido de la historia, el protagonista de esta se despierta, confuso y aturdido, en una habitación nada cuidada y prácticamente vacía, que está medio derrumbada con fragmentos de las paredes mezclados con la roca que rodea la habitación. Apenas hay una luz en el techo, parpadeando y a punto de colapsar. Cuando se vuelve más consciente, se da cuenta de que su brazo derecho está hecho de circuitos recubiertos de un material elástico pero rígido a la vez, que permite su manipulación a gusto sin problemas, y de tres cavidades circulares que parecen tener conexión con el interior del brazo. A pesar de no recordar absolutamente nada de su vida, le suena que nunca ha tenido un brazo robótico, sino uno normal como el de cualquier persona.

Entre la oscuridad puede distinguir restos robóticos y algunos huesos agrietados. Junto a estos, observa distintas armas para defenderse y considera que alguna de ellas le podría ser útil ante este entorno incierto. Desorientado y sin más opciones, decide avanzar y adentrarse en las cuevas subterráneas donde ha aparecido. Poco después se encuentra con criaturas abismales que tienen intención de atacarle. En ese momento, por instinto, saca el arma que había recogido al principio y se dispone a acabar con las criaturas.

Continúa avanzando por las Grutas del Destino hasta que se encuentra con una base muy cuidada, con un dispositivo muy grande con el que puede comunicarse a través del brazo robótico.

CAPÍTULO II: Memorias cálidas.

El protagonista se desenvuelve ágilmente en el campo de batalla mientras va haciendo suyo el control de los poderes del brazo biónico. Poco a poco, el protagonista y su brazo se van volviendo uno, y en el proceso va ganando capacidad de combate, habilidades y agilidad que le proporciona el mecanismo biónico.

Pero no menos importante, el protagonista va obteniendo también los recuerdos y memorias que yacen en el mecanismo futurista que dispone como brazo. Va teniendo visiones y obteniendo recuerdos, que aunque no son suyos, dan un inicio a la maduración de una personalidad y una identidad propia, como cualquier humano tendría.

De esta manera, se dispone a escapar de aquel lugar y vivir una nueva vida desde cero. Una vida tranquila, pero emocionante, donde se dedicará a buscar otros humanos, y de esta manera, poder conseguir tener una familia como en las memorias o en las visiones se describían. De esta manera sigue ascendiendo por la cueva hasta que se encuentra otra puerta deslizante, la cual abre con su brazo biónico y deja ver tras de sí una zona de la cueva, más vegetada y luminosa, indicando más vitalidad y una probabilidad más grande de que puedan haber humanos vivos en el mundo.

CAPÍTULO III: Dudas de la superficie.

El brazo biónico ya es parte del protagonista, la fusión es ahora total, y así lo son sus memorias y pensamientos. Ya no hay dificultad de comunicarse en combate, ni duda en la elección de habilidades. El protagonista ahora es, plenamente, un cyborg.

Esta unión se ha ido produciendo poco a poco, a lo largo de meses, por lo que el protagonista aún no se ha dado cuenta de este evento. Sin embargo, las visiones cada vez son más fuertes y ya no era capaz de distinguir qué es real y qué son recuerdos. Por fin había salido a la superficie y notaba los rayos de los soles en su rostro y su cuerpo. La ilusión que sentía por formar una familia debería estar ya en su punto máximo, pero no era así.

En su mente ya no yacía bondad e ilusión. Ahora había una nube dentro de su cabeza, que no le permitía sentir esas sensaciones. En cambio, solo sentía odio, rabia y sed de venganza. Sentimientos primitivos que una vez tuvo, o tal vez fueran del brazo biónico. Pero eso ya no importaba, el protagonista ya no era el mismo que una vez fue, un humano asustado y asolado en el interior de la cueva; ahora era un ser poderoso y ansioso por buscar más poder y cumplir su venganza a quienes lo atraparon.

CAPÍTULO IV: La verdad de Oriquath. Neyteria

La comodidad y tranquilidad que ofrecía la superficie ya no lo eran para el protagonista. No había nadie allí. Se sentía como un extraño en ese mundo, otra vez. Empezaba a notar que no debería estar ahí, o no debería haber existido nunca. Poco a poco iba cayendo más profundo en la locura. Ahora estaba vagando por el planeta, sin rumbo, solo propulsado por el poco odio que le quedaba, mientras iba, incontrolablemente, a quedarse vacío por dentro.

Pero un día, algo pasó. En sus viajes recorriendo la superficie, encontró un templo. Objetivamente, no tenía nada de especial, eran un montón de rocas y ladrillos formando estructuras, como había en todo el planeta. Pero al mirarlo de cerca, el protagonista empezó a tener sensaciones. Su corazón helado empezó a descongelarse, y unas lágrimas empezaron a brotar de sus ojos. "¿Por qué?", se preguntaba extrañado tras esa reacción.

Dudoso, decidió entrar y echar un vistazo. Por dentro, veía cristaleras, imágenes, estatuas. Todo le daba nostalgia. Siguió mirando hasta que el brazo empezó a reaccionar. El brazo estaba brillando y le estaba dirigiendo hacia el interior del templo. Las visiones no paraban, como si de mil vidas se trataran. Hasta que de repente pararon.

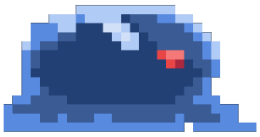
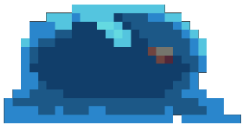
En el silencio y la oscuridad, apareció un ser de luz que iluminaba completamente el sitio. "Adelante, no tengas miedo", le dijo el ser de luz. El protagonista dudoso, se acercó y se arrodilló ante él. "¿Quién eres, y qué es este lugar que me hace sentir así?", preguntó firme mientras lo miraba. "Soy un espíritu y he venido a pedirte que salves el mundo" "Es el deber del elegido" respondió el ser de luz mientras le tocaba la cabeza.

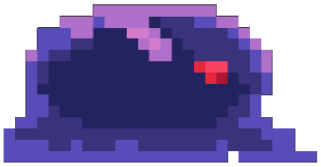
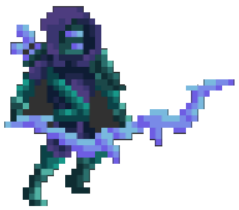
Instantes después, el ser desapareció y el templo volvió a su oscuridad inicial. El protagonista, tras esta sagrada revelación se puso en marcha y bajó por un pasadizo del templo, en busca de su enemigo final.

CAPÍTULO V: El ocaso del viejo mundo.

Cada vez, el protagonista está más profundo en el planeta. La oscuridad es casi absoluta y seguramente nunca más vuelva a ver la luz del día. Cada vez hace más calor y el terreno es de un color más rojizo.

Personajes

Personaje	Descripción	Características	Información Miscelánea
Personaje principal 	Personaje jugable. Es el protagonista de la historia. El personaje principal es un humano medio cyborg, modificado con tecnología avanzada, con un brazo robótico que le permite, no solo tener una fuerza sobrehumana, sino también altos reflejos a la hora de combatir.	El protagonista podrá moverse por el mapa y usar habilidades y herramientas de supervivencia y combate cuando las necesite. Con astucia deberá atravesar todos los niveles y escapar del territorio hostil.	Ha acabado en un mundo desconocido y deberá desarrollar útiles habilidades de supervivencia para salvar su vida.
Slime (Primer enemigo) 	Primer enemigo que se encuentra el jugador. Es un slime que hace daño al entrar en contacto con el jugador.	Dispone con una inteligencia mínima para moverse de manera repetitiva hacia los lados, como si estuviera en guardia.	Se dice que viven en las cuevas y se alimentan del moho que yace entre las rocas.
SlimeNvl2 	Slime del Nivel 2. Tiene más daño, vida y defensa que el original.	Dispone con una inteligencia mínima para moverse de manera repetitiva hacia los lados, como si estuviera en guardia.	Se dice que viven en las cuevas y se alimentan del moho que yace entre las rocas.

SlimeNvl4 	Slime del Nivel 2. Tiene más daño, vida y defensa que el original.	Dispone con una inteligencia mínima para moverse de manera repetitiva hacia los lados, como si estuviera en guardia.	Se dice que viven en las cuevas y se alimentan del moho que yace entre las rocas.
SlimeBerserker 	Slime en su versión Berserker. Gana tamaño, daño, vida y defensa.	Dispone con una inteligencia mínima para moverse de manera repetitiva hacia los lados, como si estuviera en guardia.	Se dice que viven en las cuevas y se alimentan del moho que yace entre las rocas.
Archer 	Uno de los primeros enemigos a distancia que se encuentra el jugador. Es un arquero arcano que dispara al jugador.	Su inteligencia le permite hacer guardia y vigilar. Cuando detecta al jugador lo persigue y le dispara. Si lo pierde de vista, vuelve a hacer guardia.	Los arqueros arcanos siguen vagando por el universo, saciando su sed de sangre.
Archer 	Archer del Nivel 2. Tiene más daño, vida y defensa que el original.	Su inteligencia le permite hacer guardia y vigilar. Cuando detecta al jugador lo persigue y le dispara. Si lo pierde de vista, vuelve a hacer guardia.	Los arqueros arcanos siguen vagando por el universo, saciando su sed de sangre.
ArcherHero	Último enemigo a distancia que se encuentra el jugador. Es un arquero maldito que dispara al jugador.	Su inteligencia le permite hacer guardia y vigilar. Cuando detecta al jugador lo persigue y le dispara. Si lo pierde de vista, vuelve a hacer guardia.	Los arqueros malditos siguen cazando presas por la superficie o cuevas poco profundas por todo el planeta. Se dice que mantienen una dieta equilibrada.

			
FireWorm 	Enemigo con disparos a distancia. Se caracteriza por un gran daño en cada impacto.	Su inteligencia le permite hacer guardia y vigilar. Cuando detecta al jugador lo persigue y le dispara. Si lo pierde de vista, vuelve a hacer guardia.	Traído de los orígenes del planeta, es una de las formas de vida más antiguas y agresiva que se tenga registros.
FireWormLvl4 	NightBorne del Nivel 4. Tiene más daño, vida y defensa que el original.	Su inteligencia le permite hacer guardia y vigilar. Cuando detecta al jugador lo persigue y le dispara. Si lo pierde de vista, vuelve a hacer guardia.	Traído de los orígenes del planeta, es una de las formas de vida más antiguas y agresiva que se tenga registros.
NightBorne 	Enemigo de combate cercano, con alta velocidad.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Seres artificiales con un afán de consumir todo lo que esté a su alcance. Se alimentan de la energía vital de los seres vivos.
NightBorneNvl2 	NightBorne del Nivel 2. Tiene más daño, vida y defensa que el original.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Seres artificiales con un afán de consumir todo lo que esté a su alcance. Se alimentan de la energía vital de los seres vivos.

NightBorneNvl3 	NightBorne del Nivel 3. Tiene más daño, vida y defensa que el original.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Seres artificiales con un afán de consumir todo lo que esté a su alcance. Se alimentan de la energía vital de los seres vivos.
NightBorneNvl4 	NightBorne del Nivel 4. Tiene más daño, vida y defensa que el original.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Seres artificiales con un afán de consumir todo lo que esté a su alcance. Se alimentan de la energía vital de los seres vivos.
NightBorneBerserker 	NightBorne en su versión Berserker. Gana tamaño, daño, vida y defensa.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Seres artificiales con un afán de consumir todo lo que esté a su alcance. Se alimentan de la energía vital de los seres vivos.
HAO4 	Enemigo de combate cercano. No es muy rápido, pero suele ir en grupos grandes y tiene bastante rango.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Almas perdidas que se niegan a descansar debido a su furia en sus vidas pasadas.
HAOLvl3 	HAO del Nivel 3. Tiene más daño, vida y defensa que el original.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Almas perdidas que se niegan a descansar debido a su furia en sus vidas pasadas.

HAO4Berserker 	HAO4 en su versión Berserker. Gana tamaño, daño, vida y defensa.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Almas perdidas que se niegan a descansar debido a su furia en sus vidas pasadas.
ImpMaze 	Enemigo de combate cercano. No dispone de mucha vida, pero suele ir en grupos grandes y tiene posibilidades de revivir..	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia. Cuando su vida se reduce a cero, tiene una probabilidad de volver a la vida (con una penalización) y seguir luchando.	Demonio encontrado en las profundidades del planeta. Este recibe su energía del calor sagrado de Arold.
ImpAxe 	Enemigo de combate cercano.Dispone de mucha vida, y puede ir solo o en grupos grandes. Tiene posibilidades de revivir al ser abatido.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia. Cuando su vida se reduce a cero, tiene una probabilidad de volver a la vida (con una penalización) y seguir luchando.	Demonio encontrado en las profundidades del planeta. Este recibe su energía del calor sagrado de Arold. Suelen ser los líderes del grupo.
ImpAxeBerserker	ImpAxe en su versión Berserker. Gana tamaño, daño, vida y defensa.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia. Cuando su vida se reduce a	Demonio encontrado en las profundidades del planeta. Este recibe su energía del calor sagrado de Arold. Suelen ser los líderes del grupo.

		cero, tiene una probabilidad de volver a la vida (con una penalización) y seguir luchando.	
Bat1 	Enemigo volador de combate lejano. No es muy rápido, pero vuela y respeta las distancias.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia. Tiene un sistema para no acercarse demasiado al usuario, por lo que no se quedará ni muy lejos ni muy cerca.	Seres corrompidos que habitan suelen habitar las profundas cuevas profundas. Son bastante territoriales e inteligentes.
Bat2 	Enemigo volador de combate lejano. No es muy rápido, pero vuela y respeta las distancias. Esta versión dispone de mayor tamaño y proyectiles con seguimiento.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia. Tiene un sistema para no acercarse demasiado al usuario, por lo que no se quedará ni muy lejos ni muy cerca.	Seres corrompidos que habitan suelen habitar las profundas cuevas profundas. Son bastante territoriales e inteligentes. Suele ser el jefe de la manada.
TinyFly	Enemigo volador de combate cercano. Posee una alta velocidad, volando reduce las distancias con el jugador.	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Este animal es uno de los más longevos del planeta y se alimenta de otros organismos a su alcance.

			
TinyFlyLvl4 	TinyFly del Nivel 4. Tiene más velocidad, daño, vida y defensa que el original	Su funcionamiento se basa en hacer guardia y vigilar. Cuando detecta al jugador lo persigue e intenta golpearle. Si lo pierde de vista, vuelve a hacer guardia.	Este animal es uno de los más longevos del planeta y se alimenta de otros organismos a su alcance.
Bringer Of Death 	Enemigo Jefe del nivel 1. Combate cercano y habilidades a distancia. Invoca a otros enemigos del nivel 1.	El funcionamiento inicial es de hacer guardia esperando al jugador. Dispone de un poderoso ataque cercano y de rápidos ataques a distancia que impactan por encima del jugador. Dispone de 3 fases. La primera se trata del funcionamiento anterior. En la segunda (su vida baja de la mitad) invoca a 4 enemigos para ayudarlo en su pelea. En la tercera fase (su vida baja del 20%) invoca a 6 enemigos en el combate.	Ser maldito que vive en las profundidades del planeta. Con su legendaria arma, la segadora de Sagoh, y su vestimenta de Dal'avar, concede la muerte a todos ellos que busquen su perdición.
Elemental Water	Enemigo Jefe del nivel 2. Combate cercano y habilidades a distancia. Invoca a otros enemigos del nivel 2.	El funcionamiento inicial es de hacer guardia esperando al jugador. Dispone de un poderoso ataque cercano y de rápidos ataques a distancia que impactan por encima del jugador. Además dispone de 2	Diosa lorelei que vive escondida en las profundidades de la cueva. Porta una legendaria arma, el cetro de Talorian, posee el don para controlar

		habilidades que se desbloquean al partir de la fase dos. La primera de ellas es un golpe a media distancia; y la segunda de ellas es un golpe cercano. Ambas son en área. Dispone de 3 fases. La primera dispone de un funcionamiento básico, con una forma determinada de andar, el ataque cercano y los ataques a distancia. En la segunda (su vida baja de la mitad) invoca a 4 enemigos para ayudarlo en su pelea, y desbloquea las dos nuevas habilidades. También se moverá de una forma más agitada. En la tercera fase (su vida baja del 20%) invoca a 6 enemigos en el combate.	las aguas y se cubre con su vestimenta de Princesa Lorelei. Con su enorme piedad, concede una paz tranquila a todos aquellos que no son capaces de cesar su agonía y quedan encantados con su melodía.
ElectricDefendant 	Enemigo Jefe de mitad del nivel 3. Combate cercano y habilidades a distancia. Invoca a otros enemigos del nivel 3.	Mismo funcionamiento que ElementalWater, pero con habilidades distintas.	Con sus milenios de antiguedad, este ser mantiene seguro y protegido el árbol madre de yygnail. Mediante el uso de su manto legendario, el protector de Afhén, y su sagrada espada, el brazo eléctrico de Railr, mantendrá alejado a todo aquel que busque dañar el bosque sagrado.

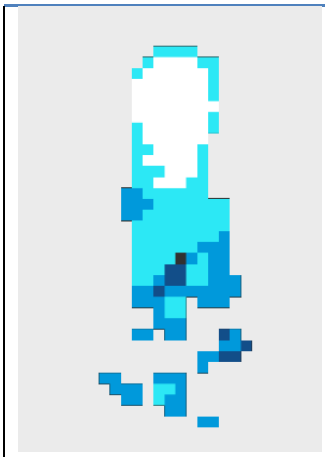
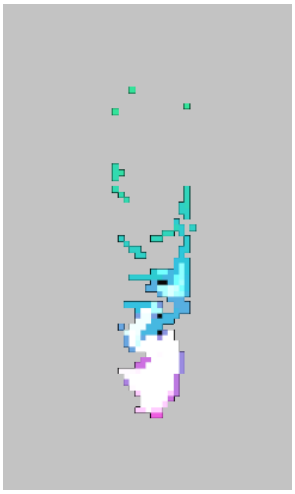
EvilWizard 	Enemigo Jefe final del nivel 3. Combate cercano y habilidades a distancia. Invoca a otros enemigos del nivel 3.	Mismo funcionamiento que ElementalWater, pero con habilidades distintas.	La traición y soledad que sufrió a lo largo de su vida, convirtió a este pobre clérigo en un ser oscuro y malévolo que se desvió de la fe de Afhén. Mediante el uso del hábito de Rohim y el cetro de mal'ar, este ser se encargará de llevar desdicha y desasosiego a todo aquel que ose entrar en las profundidades del bosque de la noche.
DemonOfAbyss 	Enemigo Jefe final del nivel 4. Combate cercano y habilidades a distancia. Invoca a otros enemigos del nivel 4.	Además del funcionamiento desarrollado en ElementalWater, este enemigo cuenta con una habilidad de evasión de daño. Cuando su vida llega a cero, invoca al enemigo SlimeBoss en la posición de muerte.	Súdbito de la orden del maligno, también apodado como Malzahar, que defiende la zona más profunda del reino del mal. Porta la espada maldita de sasog, y la armadura demoníaca de kat'lon, para llevar a un infierno terrenal a todo aquel que se adentre en este terreno maldito.
SlimeBoss	Enemigo Jefe final Principal del nivel 4. Combate cercano y habilidades a distancia. Invoca a otros enemigos de alto calibre.	Mismo funcionamiento que ElementalWater, pero con habilidades distintas. Cuenta con habilidades más potentes e invocaciones de	Monstruo espiritual maldito, que se imbuye en los cuerpos de los hombres mediante la maldición de Malzahar. Este espíritu legendario

		enemigos berserker.	y mitológico lleva a su final a todo ser y terreno que se interponga en su camino cuando es invocado en el mundo terrenal.

Proyectiles

Proyectil	Descripción	Características	Información Miscelánea
Bullet 	Proyectil que es lanzado por el personaje principal Player mediante el cañón láser.	El proyectil colisionará con enemigos y elementos del entorno.	Una de las balas básicas de energía de neutrones. Simple, pero potente.
Arrow 	Proyectil que es lanzado por el enemigo Archer.	El proyectil colisionará con el Player y elementos del entorno.	Flechas de metal inyectadas con energía de neutrones. Rápida y dolorosa potente.
Arrow 	Proyectil que es lanzado por el enemigo ArcherHero.	El proyectil colisionará con el Player y elementos del entorno.	Flechas de metal inyectadas con energía de neutrones. Rápida y dolorosa potente.

FireWormProjectile 	Proyectil que es lanzado por el enemigo FireWorm.	El proyectil colisionará con el Player y elementos del entorno.	Trozos de magma y ácido, convertidos en arma arrojadiza.
BatProjectile 	Proyectil que es lanzado por los enemigo Bat1 y Bat2.	El proyectil colisionará con el Player y elementos del entorno. Puede tener seguimiento o no del Player.	Trozo de carne y hueso, desprendido del cuerpo del huésped y los desechos de sus presas.
BringerOfDeathProjectile 	Proyectil que es invocado por el enemigo BringerOfDeath.	El proyectil solo colisionará con el Player. Este proyectil contiene animación. Solo en el momento idóneo (clímax) se producirá el daño al Player.	Energía oscura, obtenida de las almas perdidas que perecen en las profundidades.
ElementalWaterProjectile	Proyectil que es invocado por el enemigo ElementalWater.	El proyectil solo colisionará con el Player. Este proyectil contiene animación. Solo en el momento idóneo (clímax) se producirá el daño al Player.	Porciones de almas y energía de agua y luna que forman una especie de fuego, que arde y quema todo aquello que causa dolor a los hombres.

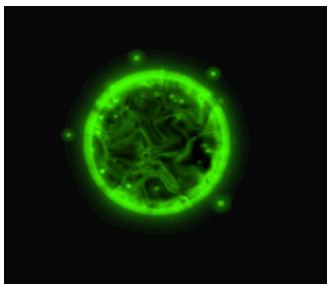
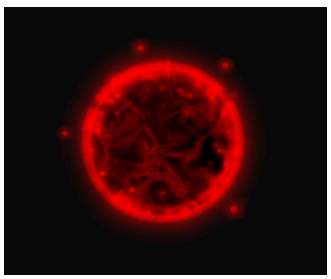
			
ElectricDefendantProjectile 	Proyectil que es invocado por el enemigo ElectricDefendant.	El proyectil solo colisionará con el Player. Este proyectil contiene animación. Solo en el momento idóneo (clímax) se producirá el daño al Player.	Fuerza eléctrica de los cielos canalizada en un punto cálido, donde se purifican las dudas y los males.
EvilWizardProjectile 	Proyectil que es invocado por el enemigo EvilWizard.	El proyectil solo colisionará con el Player. Este proyectil contiene animación. Solo en el momento idóneo (clímax) se producirá el daño al Player.	Energía maldita y sagrada que interactúan en un punto del espacio, luchando para acabar la una con la otra.

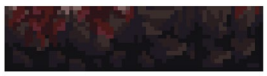
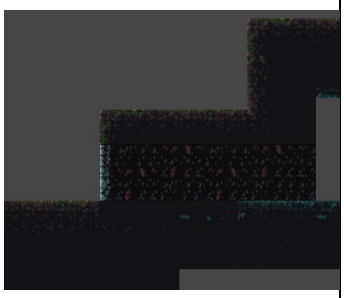
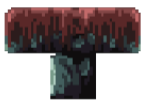
DemonOfAbyssProjectile 	Proyectil que es invocado por los enemigo DemonOfAbyss y SlimeBoss.	El proyectil solo colisionará con el Player. Este proyectil contiene animación. Solo en el momento idóneo (clímax) se producirá el daño al Player.	Brasas candentes e ígneas invocadas en un punto, que abrasan e incineran cualquier trozo tipo de materia que encuentre a su alcance.

Decoración e Interactuables

Entidad	Descripción	Características	Información Miscelánea
Puerta SciFi 	Objeto decorativo	Reacciona al movimiento del jugador cuando es atravesado.	Puertas de alta tecnología que una vez fueron usadas por una civilización adelantada.
Obelisco 	Objeto decorativo	Ilumina el mapa y ofrece una visión temática del nivel al jugador.	Elemento sagrado y de culto, que fue construido como símbolo de veneración de una civilización anterior.

ToggleButton 	Objeto interactuable. El jugador puede usarlo mediante el uso de la tecla Barra Espaciadora.	Permite la generación u ocultación de terreno y plataformas del nivel.	Mecanismos de la era antigua que fueron usados para la mejora de vida de la civilización.
ToggleButtonBlue 	Objeto interactuable. El jugador puede usarlo mediante el uso de la tecla Barra Espaciadora.	Permite la activación y desactivación de elementos de puzles del nivel.	Mecanismos de la era antigua que fueron usados para la mejora de vida de la civilización.
Suministrador 	Objeto interactuable. El jugador puede usarlo mediante el uso de la tecla Barra Espaciadora.	Ofrece al jugador una serie y cantidad de materiales útiles para su uso en el sistema de inventario y tienda del juego.	Estructura antigua que mezcla conocimientos de tecnologías modernas y ancestrales.
Agua 	Objeto interactuable. El jugador reaccionará al elemento mediante el contacto con él.	El jugador será desplazado en la dirección en la que se mueva dicha corriente. Esta permitirá o no el movimiento del personaje mientras esté dentro de ella.	Corriente de agua encontrada en ciertas alturas de las cuevas y el planeta.
Lava 	Objeto interactuable. El jugador reaccionará al elemento mediante el contacto con él.	El jugador recibirá gran daño de forma frecuente mientras siga en contacto con este elemento.	Río de lava procedente de las profundidades del abismo y del centro del planeta.

Ventilador Ascendente 	Objeto interactuable. El jugador reaccionará al elemento mediante el contacto con él.	El jugador será desplazado en sentido ascendente mientras siga dentro de la corriente de aire.	Objeto mecánico diseñado por algún tipo de civilización no muy antigua.
Ventilador Ascendente Activable 	Objeto interactuable. El jugador reaccionará al elemento mediante el contacto con él.	El jugador será desplazado en sentido ascendente mientras siga dentro de la corriente de aire. Este ventilador estará desactivado, y podrá ser activado o desactivado mediante otros elementos del nivel.	Objeto mecánico diseñado por algún tipo de civilización no muy antigua.
Wrap Portal Verde 	Objeto interactuable. El jugador puede usarlo mediante el uso de la tecla Barra Espaciadora.	Al pulsar la tecla correspondiente, el jugador será teletransportado de un lado del portal hasta el otro.	Energía del cosmos condensada para permitir la comunicación entre dos puntos dimensionales separados.
Wrap Portal Rojo 	Objeto interactuable. El jugador puede usarlo mediante el uso de la tecla Barra Espaciadora.	Al pulsar la tecla correspondiente, el jugador será teletransportado de un lado del portal hasta el otro.	Energía del cosmos condensada para permitir la comunicación entre dos puntos dimensionales separados.
Elevator 	Objeto interactuable. El jugador reaccionará al elemento mediante el contacto con él.	Plataforma móvil de estilo decorativo. Podrá ser usada por el jugador, y deberá ser activada para su funcionamiento, el cual le permitirá moverse entre dos puntos.	Elemento tecnológico creado para facilitar la vida de alguna civilización antigua,

Plataforma Móvil 1 	Objeto interactuable. El jugador reaccionará al elemento mediante el contacto con él.	Plataforma que se mueve sistemáticamente entre dos puntos. Podrá ser usado por el protagonista. Podrá estar activado o desactivado, y podrá ser interactuado por otros elementos del nivel.	Elemento tecnológico creado para facilitar la vida de alguna civilización antigua,
Plataforma Móvil 2 	Objeto interactuable. El jugador reaccionará al elemento mediante el contacto con él.	Plataforma que se mueve sistemáticamente entre dos puntos. Podrá ser usado por el protagonista. Podrá estar activado o desactivado, y podrá ser interactuado por otros elementos del nivel.	Elemento tecnológico creado para facilitar la vida de alguna civilización antigua,
Plataforma Destruible 	Objeto interactuable. El jugador reaccionará al elemento el uso de ataques contra él.	Plataforma que tiene la capacidad de desaparecer si sufre un número suficiente de ataques por parte del jugador.	Rocas y trozos de tierra colocados a propósito, con el fin de bloquear el acceso a los invasores.
Plataforma no rígida 	Objeto interactuable. El jugador reaccionará al elemento mediante el contacto con él.	Plataforma que se va disolviendo hasta desaparecer cuando contacta con el jugador. Reaparece después de un tiempo.	Terreno arcaico con propiedades mágicas, que permiten reacciones exóticas en la materia.

Jugabilidad

Visión general del Juego

El juego se basa en los géneros plataformas y acción, aunque también bebe de aventura y puzzles, dado a que deberás explorar los distintos niveles y deberás realizar distintos rompecabezas que te otorgarán ciertas recompensas al resolverlos.

El juego será desarrollado para PC, será offline y de un solo jugador. Al jugar en solitario, se creará una experiencia más inmersiva en el juego, aumentando tu concentración y concentración durante el juego. Tendrá dos modos de juego, normal y difícil.

Experiencia del jugador

Vivirás la emoción e intriga de la supervivencia en un lugar recóndito y desconocido, donde podrás ir mejorando a tu personaje, superando así los distintos niveles y obteniendo nuevas herramientas y nuevas mejoras.

Deléitate de los hermosos paisajes de cada nivel. Corre, salta, asesina a tus enemigos y fortalécete mientras avanzas en la historia principal.

Pautas de jugabilidad

Siendo un juego donde deberás desenvolverte y matar monstruos, cierto nivel de dificultad y violencia es de esperarse. No obstante, ayudándonos del estilo de diseño y a la fantasía pretendemos alejarnos un poco del realismo y la brutalidad de los golpes, haciéndolo apto para un mayor rango de edades.

El juego se irá volviendo más difícil conforme se vaya avanzando en los niveles y la historia, lo que le irá aportando emoción y evitará la monotonía en la realización de cada nivel.

Se usará un lenguaje maduro con ciertos toques de ironía, aunque no será muy complejo. Se podrán encontrar referencias sutiles repartidas por los niveles, niveles que serán finitos.

Objetivos del juego y Recompensas

El jugador irá obteniendo nuevas herramientas y módulos conforme vaya recorriendo los niveles. El loot será aleatorio, pero habrá momentos donde recibas un objeto de forma asegurada, esto es, al derrotar a un boss y al realizar correctamente un puzzle entre niveles.

Como se explicará más tarde, las zonas o niveles son afectados por el ciclo de día y noche, y algunos fenómenos meteorológicos, los cuales ofrecerán de forma directa una ventaja o desventaja al jugador.

También podemos encontrar una desventaja que no parece muy evidente. Esta es tener mala suerte y que te poque poco loot, o que no resuelvas bien los puzzles. Esto supondrá, a la larga, tener menos opciones, y seguramente, menos capacidades para poder realizar los niveles correctamente.

La dificultad de cada nivel se irá incrementando, de forma que los enemigos tendrán más nivel y no dejarán tanto loot. Obviamente, los objetos dejados por estos monstruos más fuertes, serán de mejor calidad, por lo que conforme se avance hacia el final será mucho más importante la recolección de estos. Esto da por hecho que el jugador se acostumbrará al juego y se volverá más experto en sus mecánicas.

Mecánicas del juego

Atributos del personaje principal

El personaje es un humano medio cyborg, modificado con tecnología avanzada, con un brazo robótico que le permite, no solo tener una fuerza sobrehumana, sino también altos reflejos a la hora de combatir. Pero lo más importante es que el brazo tiene tres ranuras en las que, más adelante del juego, podrá insertar distintos tipos de *Módulos del ocaso* (ver más adelante). Puede equiparse también con armas de distintos tipos (espada corta, maza, lanza y cañón láser) y trajes que le proporcionarán habilidades pasivas. Tanto las armas como los trajes tendrán un peso que irá variando y afectará a la capacidad de movilidad del personaje al llevarlos equipados.

El personaje podrá realizar los movimientos básicos de un plataformas: saltar, moverse hacia los lados, hacia abajo y correr. Además podrá realizar acciones como atacar con el arma que esté portando y hacer cambios de arma, podrá realizar evasión para esquivar golpes enemigos y podrá usar módulos que tenga a su disposición. El personaje dispondrá de una vida y resistencia limitadas, además de atributos y estadísticas del personaje.

Las estadísticas del personaje principal son:

- **Nombre**: [*string*] Nombre que recibe el personaje protagonista.
- **Ataque**: [*float*] El ataque total que posee el personaje.
- **Defensa**: [*float*] La defensa total que posee el personaje.
- **Resistencia**: [*float*] La resistencia total que posee el personaje.
- **Salud**: [*float*] Los puntos de salud totales que posee el personaje.
- **Peso**: [*float*] El peso total que posee el personaje.
- **Arma**: [*arma*] El arma que tiene equipada actualmente el personaje.
- **Traje**: [*traje*] El traje que tiene equipado actualmente el personaje.
- **Módulos**: [*módulos*] Módulos del Ocaso que tiene conectados actualmente el personaje.

Además, el personaje tiene otros atributos ocultos que no son mostrados al usuario:

- **Vel. Ataque**: [*float*] La velocidad con la que ataca el personaje.
- **Vel. Movimiento**: [*float*] La velocidad con la que se mueve el personaje.
- **Evasión**: [*float*] El tiempo en segundos de invulnerabilidad que tiene el personaje al evadir.

- **Gravedad**: [*float*] La fuerza de gravedad con la que es atraído hacia abajo el personaje.
- **Salto**: [*float*] La altura con la que el personaje puede saltar.

Cálculo del daño recibido

El daño que va a recibir el personaje de los enemigos se calcula a partir de la siguiente fórmula:

$$dañoRecibido = 5 * \frac{atqEnemigo}{\sqrt{defPersonaje}} * (1 - dañoReducido)$$

siendo ***atqEnemigo*** el ataque del enemigo que provoca el daño al jugador (dependiente del nivel del enemigo), ***defPersonaje*** la defensa del personaje y ***dañoReducido*** la reducción del daño del personaje.

Atributos de los enemigos

Los enemigos son entidades que se encuentran repartidas por todas las zonas del mapa. Los hay de distintos tipos y de distintos niveles. Cuanto mayor sea el nivel del enemigo, mejores serán sus estadísticas.

Según el tipo de enemigo podrán moverse, saltar, lanzar proyectiles, e incluso evadir un ataque al protagonista.

Las características principales de los enemigos son:

- **ID**: [*int*] número de identificación del módulo dentro del juego.
- **Nivel**: [*int*] El nivel del enemigo.
- **Ataque**: [*float*] El ataque base que posee el enemigo.
- **Defensa**: [*float*] La defensa base que posee el enemigo.
- **Salud**: [*int*] Los puntos de salud base que posee el enemigo.
- **Nombre**: [*string*] nombre del enemigo.
- **Descripción**: [*string*] descripción del enemigo.
- **Vel. Ataque**: [*float*] La velocidad con la que ataca el enemigo.
- **Vel. Movimiento**: [*float*] La velocidad con la que se mueve el enemigo.

Todas las características físicas de los enemigos escalan con su nivel según la fórmula recursiva

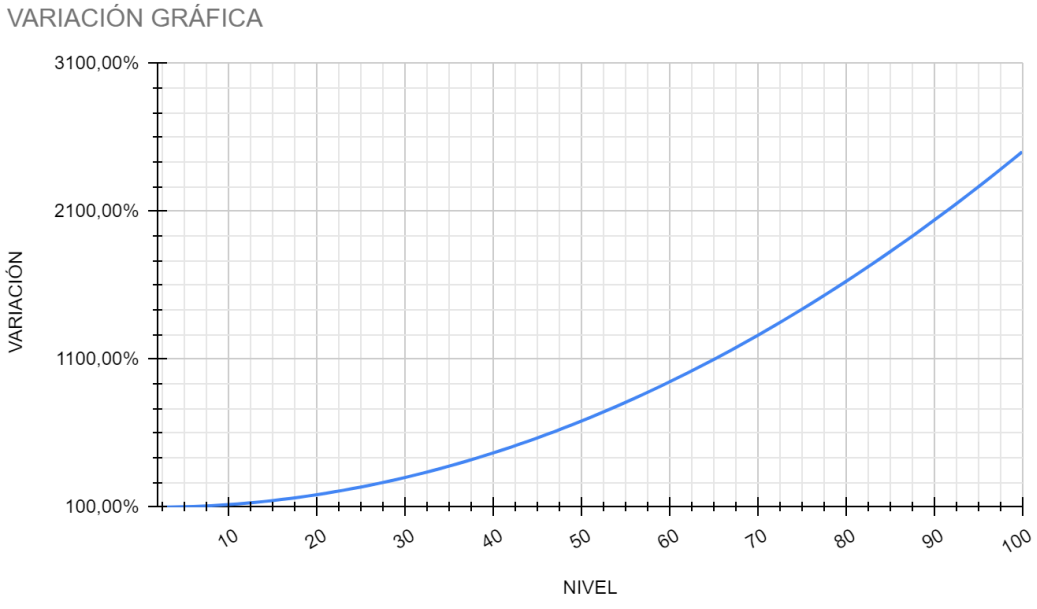
$$var(n) = var(n-1) \cdot \frac{(n-1) \cdot a}{2}, \qquad \forall n \in [2, nivelMax]$$

que, al convertir en forma iterativa, se queda como

$$var(n) = 100\% + \sum_{i=2}^{n} \frac{(i-1) \cdot a}{2} = 100\% + \frac{a}{2} \cdot \sum_{i=2}^{n} (i-1)$$
$$= 100\% + \frac{n(n-1) \cdot a}{4}$$

siendo *n* el nivel del enemigo y *a* la constante de aumento en el que se basa la fórmula. El porcentaje de variación que se obtiene se aplica sobre las características bases del enemigo. El nivel máximo que puede alcanzar un enemigo por sí solo (en la fórmula como *nivelMax*) es 100.

Para $a = 0{,}99\%$, se obtiene la siguiente gráfica:



Data: Enemigos

Items y Recursos

ID	ICONO	NOMBRE	RAREZA	DESCRIPCIÓN	USABLE
0		NONE	1	NONE	No
1		Degiterio	1	Recurso utilizado para canjear en el Nexo Central.	No
2		Occaterio	4	Material raro que se puede emplear para canjear en el Nexo Central, entre otras cosas.	No
3		Suero vital	2	Suero que restaura unos pocos puntos de salud.	Sí
4		Suero energético	2	Suero que reduce un poco el consumo de la resistencia.	Sí
5		Suero fortalecedor	2	Suero que aumenta ligeramente el ataque.	Sí
6		Suero protector	2	Suero que aumenta ligeramente la defensa.	Sí
7		Mineral fragmentado	2	Pedazo de mineral desbordado de las paredes de las cuevas de Oriquaht.	No
8		Nexotek	5	Cápsula que permite la unión con el Gran Nexo Central. Consúmelo para teletransportarte al último Nexo que te conectaste.	Sí
9		Mineral compacto	3	Trozo de mineral consistente formado por minerales más pequeños. Se puede también encontrar en ciertas zonas de las Cuevas del Alba.	No

10		Piedras de lava	3	Residuos de piedras pequeñas formadas por la lava de Oriquath. No se puede manipular directamente con las manos debido a su alta temperatura, pero sí al utilizar algún tipo de protección metálica que resista el calor.	No
11		Roca de magma	4	Roca bañada en un poderoso e incandescente magma creada por la compresión de pequeñas piedras de lava. Quema todo lo que se le acerque.	No
12		Sollozos del crepúsculo	4	Ciertas piedras de las que se desconoce el origen y la formación. Si las aprietas con fuerza, con suerte se pueden oír los sollozos de aquellas que serán víctimas de la Catástrofe del Principio y Fin.	No
13		Temor del crepúsculo	5	Gran roca que amacena un poderoso y siniestro poder. Al sujetarla con las manos, se puede sentir el miedo y los gritos de dolor de esos que perecerán el la Catástrofe del Principio y Fin.	No

Data: Items

Modos de juego

1. **Modo Normal**

Modo de juego por defecto. La vista es en tercera persona. Este modo de juego es básicamente la dificultad para la que está pensada la historia y los objetivos, para que sea un juego entretenido e interesante. Está guiada hacia un público más general que quiera disfrutar de un juego con cierta dificultad y a la vez no sea frustrante por su dificultad, que reduciría así la apreciación del desarrollo de la historia.

2. **Modo Difícil**

Modo de juego opcional. La vista es en tercera persona. Este modo de juego está realizado para jugadores avanzados que quieran proponerse retos. Realmente no cambia mucho con el modo original, pero el simple hecho de un decremento de los atributos del personaje y aumento en los de los enemigos, junto a cambios en las probabilidades, supone un gran escalón de dificultad. Aquí, el jugador experimentará más intensidad, más tensión y más frustración, lo que dará una perspectiva más hostil e interesante a los jugadores veteranos.

Equipamiento - Sistema de puntuación del Personaje

El personaje principal no tiene sistema de niveles, por lo que cualquier aumento en su daño, vida y demás atributos dependen exclusivamente de su equipación.

ARMAS

Las armas son equipamiento que podrá utilizar el personaje para defenderse y atacar a los enemigos (excluyendo el brazo). Cada arma tiene las siguientes características:

- **ID**: [*int*] número de identificación del arma dentro del juego. No distingue entre los tipos de armas.
- **Tipo**: [*struct*] el tipo de arma. Puede ser *Espada Corta*, *Maza, Lanza* o *Cañón Láser*.
- **Ataque**: [*float*] define el daño que hace ese arma por sí sola.
- **Peso**: [*float*] define la masa de ese arma. Esta variable afecta al movimiento y desplazamiento del personaje, así como a la stamina que usará este al golpear con el arma.
- **Nombre**: [*string*] nombre del arma.
- **Descripción**: [*string*] descripción del arma.

ID	TIPO	ATAQUE	PESO	NOMBRE	DESCRIPCIÓN

0	Espada	0	0	Unnamed	Unnamed
1	Espada	3,0	4,0	Espada Corta	Una espada de corto alcance forjada para aquellos que poseen el coraje de enfrentarse a este mundo.
2	Maza	5,0	7,0	Maza	Una maza robusta pensada para demoler a las bestias que habitan las cavernas. Pesada pero poderosa.
3	Laser	1,0	2,0	Cañón Láser	Un cañón que canaliza energía para disparar pequeñas balas láser a gran velocidad.
4	Espada	3,5	4,3	Sable picapiedra	
5	Maza	6,8	8,2	Porra demolerocas	
6	Laser	1,9	3,3	Pistola rompemuros	
7	Espada	3,2	3,7		
8	Espada	5	6,1		
9	Laser	2,5	4		
10	Lanza	4,0	5,0	Lanza ligera	Una lanza de medio alcance utilizada por los guerreros más salvajes y feroces de este mundo.
11	Lanza	5,1	6	Lanza pesada	

Data: Armas

TRAJES

Los trajes son equipamiento que cubrirá al personaje con el propósito de reducir el daño recibido por el entorno y de otorgar habilidades pasivas que pueden, o bien activarse dado una condición o condiciones, o bien permanecer siempre activas. Cada traje tiene las siguientes características:

- **ID**: [*int*] número de identificación del traje dentro del juego.
- **Defensa**: [*float*] define la reducción del daño del traje por sí solo que recibirá el personaje.
- **Peso**: [*float*] define la masa de ese traje. Esta variable afecta al movimiento y desplazamiento del personaje.
- **Habilidad**: [*script*] define la habilidad pasiva del traje.

- **Nombre**: [*string*] nombre del traje.
- **Descripción**: [*string*] descripción del traje.

ID	DEFENSA	PESO	HABILIDAD	NOMBRE	DESCRIPCIÓN
250	1,0	2,0	Ninguna	Túnica protectora	Vestimenta que utilizó alguna civilización antigua para protegerse contra otras civilizaciones.
251	1,3	2,5	Aumenta los puntos de salud en 20	Traje de vitalidad	Un traje tejido con plantas místicas que cura las heridas de los guerreros más débiles.
252	1,6	3,5	Aumenta el ataque un 12%	Manto de la furia	El vestido que concentra la ira necesaria para luchar en combate sin miedo a la debilidad.
253	1,4	3,0	Aumenta la velocidad de ataque en un 15%	Ropa de los Rauvnir	Ropa que solía pertenecer a la raza de los Rauvnir, quienes tenían una brutal agilidad en combate con las armas.
254	1,4	3,0	Aumenta la velocidad de movimiento en un 15%	Túnica del mensajero	Los mensajeros usaban estas prendas para desplazarse rápidamente por las tierras de Oriquath, pero algunos cazadores y guerreros las usaban para huir de sus presas o perseguirlas.
255	1,7	3,5	Aumenta la velocidad de ataque en un 7% y la velocidad de movimiento en un 8% cuando la salud se encuentre por encima del 75%.	Túnica ligera	
256	2,2	4,0	Al derrotar un enemigo, el personaje recupera un tanto por ciento de salud equivalente al nivel del enemigo derrotado	Vestido del cosechador	
257	2,5	7,0	Cuando el personaje se encuentra en el	Conjunto volador	

			aire, hay un 30% de probabilidad de no consumir resistencia al intentar consumirla.		
258	2,1	4,5	Cuando la salud del personaje se encuentra por debajo del 20%, el ataque aumenta un 50%	Armadura de Ftreqis	
259	2,0	5,0	Al activar un módulo del ocaso, los tiempos de espera de todos los módulos equipados se reducen en un 15%, pero la velocidad de movimiento se reduce en un 5%. Este efecto se puede acumular hasta 3 veces como máximo y cada efecto desaparecerá cuando se pueda volver a usar cada Módulo del Ocaso.	Sábana del tiempo	
260	2,5	5,0	Aumenta la probabilidad de que haya tormenta	Bendición de las nubes	
261	1,8	4,0	Al llevar equipado una espada ligera, el daño inflingido aumentará en 10/15/25% según se tenga equipado 1/2/3 Módulos del Ocaso respectivamente.	Manto de la locura	Vestido legendario creado mediante artes oscuras que saca la demencia más profunda del alma.

Data: Trajes

MÓDULOS DEL OCASO

Los *Módulos del Ocaso* son unos módulos esféricos que se pueden insertar en el brazo robótico del personaje. Estos módulos proporcionan habilidades distintas a las de los trajes, mucho más poderosas y útiles. Algunos de los módulos del ocaso son tan poderosos que necesitan de varias esferas para poder funcionar. Las habilidades no son pasivas, es decir, las puede accionar el jugador cuando desee. Cada módulo del ocaso tiene las siguientes características:

- **ID**: [*int*] número de identificación del módulo dentro del juego.
- **Ranuras**: [*int*] indica las ranuras necesarias para que el módulo funcione. Estas ocuparan las ranuras del brazo, que tiene 3 como máximo.
- **Duración**: [*float*] tiempo en segundos que dura el efecto de la habilidad del módulo.
- **TdE**: [*float*] tiempo de espera en segundos para poder volver a usar la habilidad del módulo una vez finalizado su efecto.
- **Habilidad**: [*script*] define la habilidad del módulo.
- **Nombre**: [*string*] nombre del módulo.
- **Descripción**: [*string*] descripción del módulo.

ID	RANURAS	DURACIÓN	TdEspera	HABILIDAD	NOMBRE	DESCRIPCIÓN
1	1	15,0	30,0	Durante el periodo del efecto, la capacidad de salto aumenta en un 50/75/100%.	Módulo de supersalto	Un sencillo módulo desarrollado y programado para los primeros prototipos de Neyteria. Al conectarse con estos, se puede llegar más alto al pegar un salto. Útil para llegar a zonas que de normal no se alcanzan con un simple salto.
2	2	5,0	60,0	Durante el periodo del efecto, los puntos de salud no decrecen.	Módulo de invencibilidad	Desde la Guerra del Ocaso se empezaron a desarrollar estos poderosos módulos, fuertes pero con limitaciones. En la guerra, muchas vidas se perdieron instantáneamente, con un simple toque de armas. Fue por eso que, después de dicha batalla, se desarrollo este módulo capaz de resistir cualquier tipo de

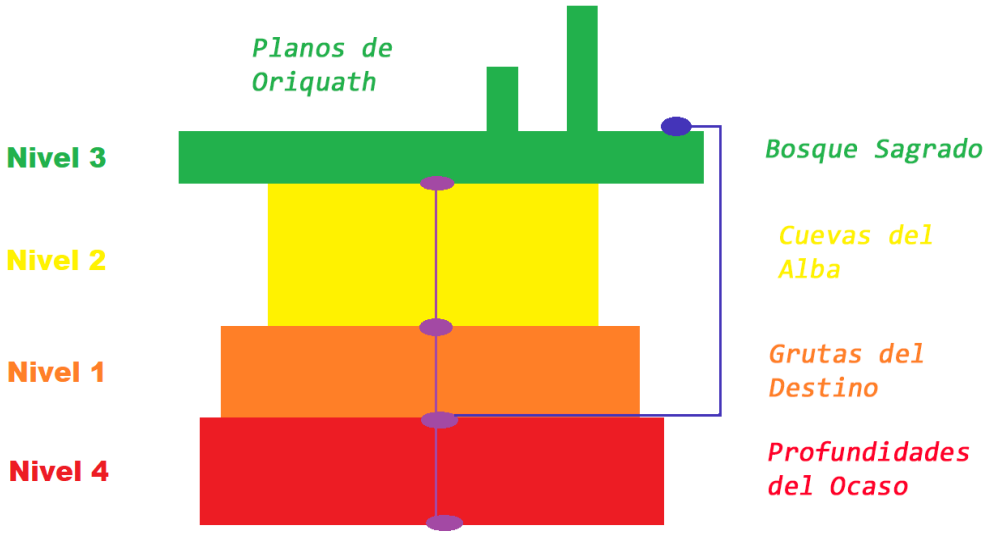
						ataque durante un corto periodo de tiempo.
3	2	7,0	60,0	Durante el periodo del efecto, la resistencia no decrece.	Módulo de vigor	Aquellos ágiles valientes que pudieron esquivar los ataques de sus rivales y contraatacarlos a la vez, perecieron al poco tiempo de lucha por falta de resistencia. Pues para ello se desarrolló este módulo, que permite extender sorprendentemente el aguante de su portador.
4	1	10,0	25,0	Durante el periodo de efecto, el daño recibido se verá reducido en un 20/40/60%	Módulo de Rumarh	Se cuenta que este Módulo del Ocaso se diseñó con la idea de asemejarse a uno de los guerreros gigantes más resistentes. Se llamaba Rumarh y era admirado por su capacidad de resistir los golpes con mayor facilidad.

Data: Módulos

Diseño de nivel. Mecánicas

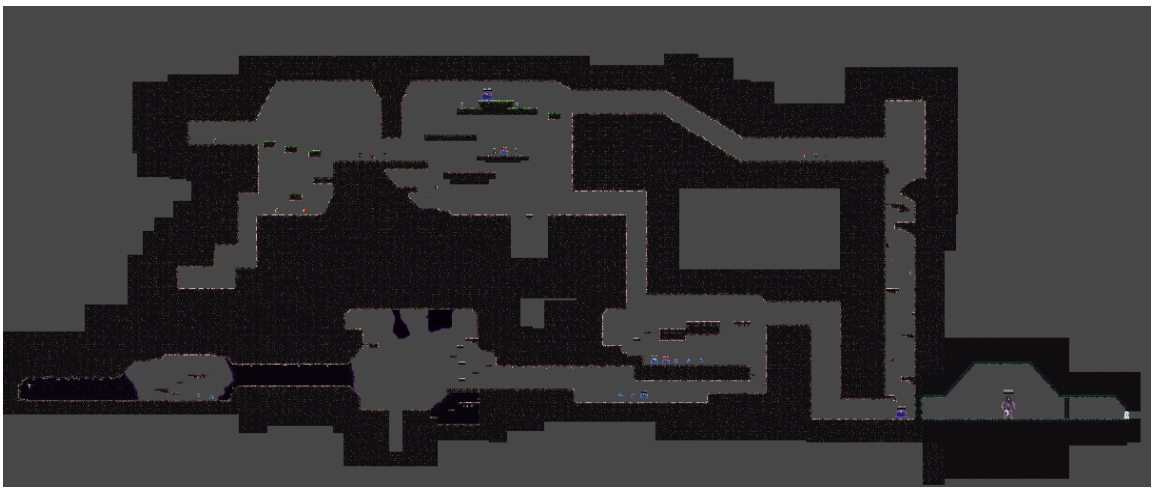
El mapa del juego está distribuido por zonas o niveles, que a su vez están agrupadas en distintas regiones. Todas las zonas están conectadas en un mismo espacio, es decir, que realmente no hay pantallas de carga de cara al usuario, sino que se van cargando las zonas adyacentes a la que se encuentra el personaje (totalmente sujeto a cambios en caso de que no se pudiese implementar). El mapa tendrá un total de 4 niveles, con temáticas, paisajes y ambientes diferentes, cada uno con un boss final de zona. Se

dispondrá, además, de un notable incremento de la dificultad conforme vayamos superando cada nivel. (Falta la explicación de cada nivel, sobre todo artísticamente).

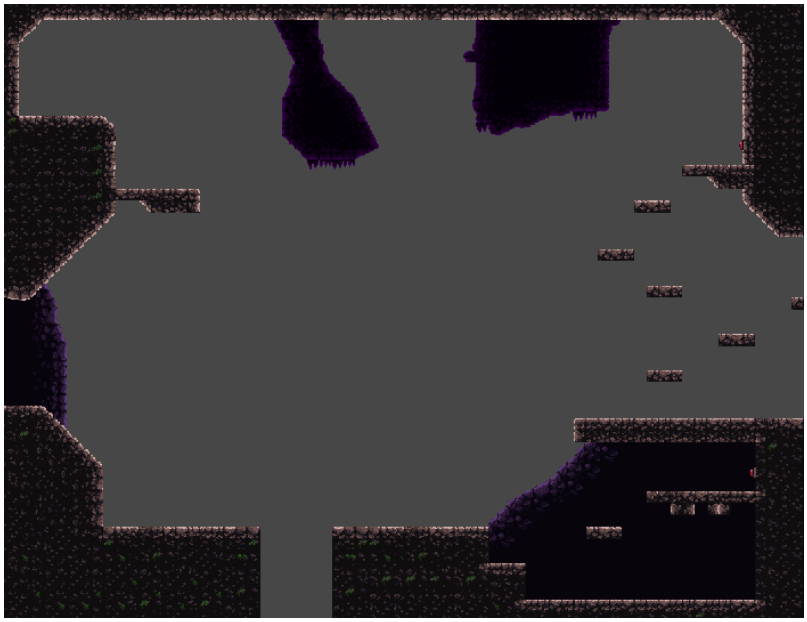


1. **Nivel 1 - Grutas del Destino**

Típica estética de cuevas a las que estamos acostumbrados. Un poco de oscuridad, tonos un poco oscuros y enemigos de acuerdo a la temática.



Nivel 1 - imagen 1 (general)



Nivel 1- imagen 2 (ejemplo de mapa desde cerca)

2. Nivel 2 - Cuevas del Alba

Estética de cuevas, pero con más iluminación y cierta vegetación. Tonos un poco más claros y enemigos de acuerdo a la temática.

Nivel 2 - imagen 1 (general)

Nivel 2 - imagen 2 (ejemplo de mapa desde cerca)

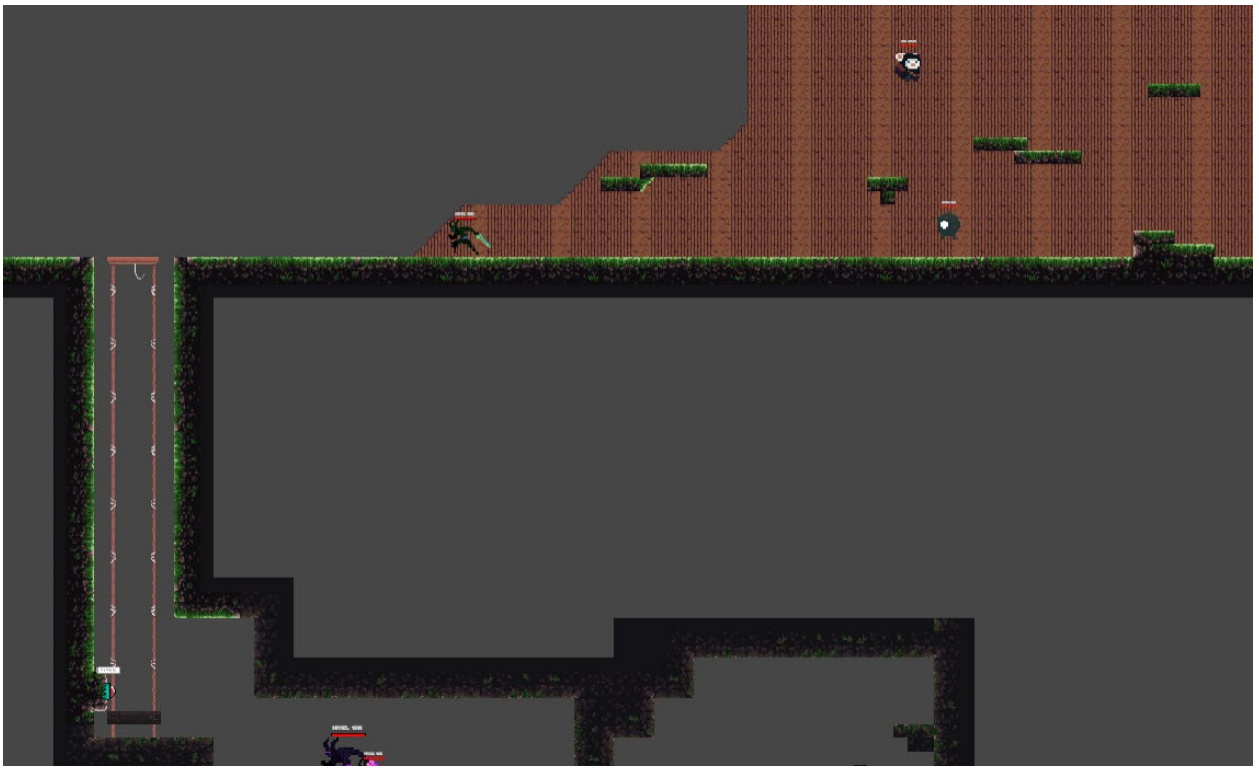
3. Nivel 3.1 - Planos de Oriquath

Estética de superficie con vegetación y edificios/infraestructuras como ruinas. Buena iluminación y fondo de mapa muy vistoso viendo los soles y alguna estrella. Enemigos de acuerdo a la temática.

Nivel 3.2 - Bosque Sagrado

Estética de superficie con más vegetación y pocas ruinas. Buena iluminación y fondo de mapa muy vistoso con buena vista del cielo y los árboles. Enemigos de acuerdo a la temática.

Nivel 3 - imagen 1 (general)

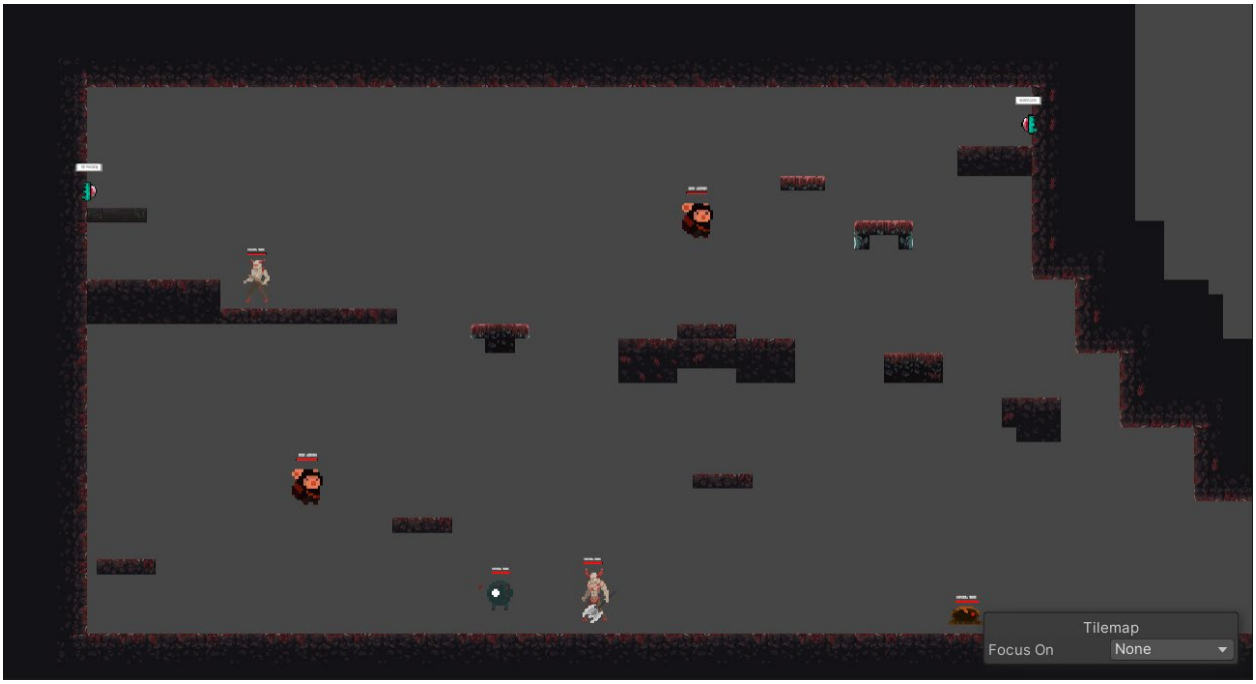


Nivel 3 - imagen 2 (ejemplo de mapa desde cerca)

4. Nivel 4 - Profundidades del Ocaso

Estética de cuevas, con más oscuridad y temática muy árida. Tonos más rojizos, vegetación nula, y decoración de lava abundante. Enemigos de acuerdo a la temática.

Nivel 4 - imagen 1 (general)

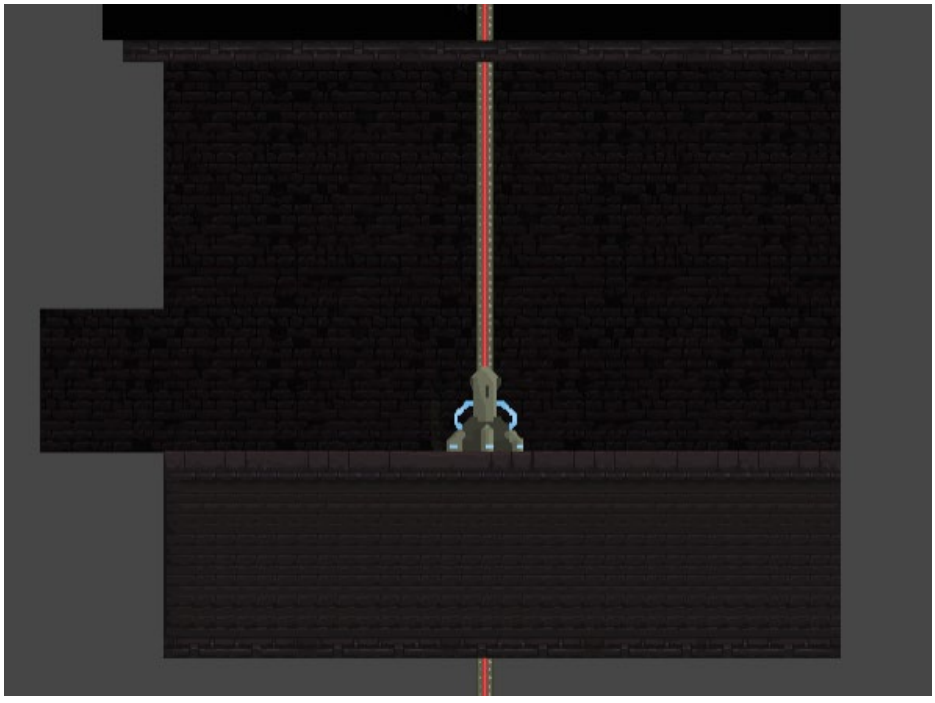


Nivel 4 - imagen 2 (ejemplo de mapa desde cerca)

5. RestZone - Zona de descanso entre niveles

Estética de edificio, con poca oscuridad y temática suave. Tonos marrones, vegetación nula, y decoración de infraestructura.

ZoneLoader - imagen 1



ZoneLoader - imagen 2

Mapa final con todos los niveles

Neyteria- imagen del mapa completo

CLIMA

Durante todo el tiempo, en cualquier zona del mapa, incluídas las subterráneas e interiores, el ciclo de día y noche y ciertos fenómenos meteorológicos pueden afectar considerablemente al entorno. En estas situaciones, por si se diera el caso de que el jugador no se encontrase al aire libre, se le notificará por pantalla que el clima ha cambiado. De igual manera, se le indicará cuando ha vuelto a la normalidad.

1. **Día y Noche**

Cuando cae la noche, los enemigos se fortalecen, incrementando su nivel en 5 unidades. En esta situación, el nivel de los enemigos se mostrará de color naranja y todos sus estados se verán escalados por el nivel. Además, el intervalo de invulnerabilidad de la evasión del personaje se verá reducido en un 25%.

Una vez termina la noche y amanece, el nivel vuelve al original.

2. **Tromenta**

El clima es muy inestable en este planeta, pues cuando llueve siempre hay tormenta. Es el mejor momento para enfrentarse a los enemigos, ya que la lluvia tiene ciertas propiedades que los debilitan, de modo que el nivel de estos se decrementa en 5 unidades. En esta situación, el nivel de los enemigos se mostrará de color azul celeste y todos sus estados se verán escalados por el nivel. También puede provocar que el personaje se resbale al desplazarse por la superficie. Además, se reduce el tiempo de espera de los Módulos del ocaso en un 25%.

Una vez termina la tormenta y el cielo se despeja, el nivel vuelve al original. La probabilidad de que haya tormenta es del 10% cada día (únicamente durante el día y en la superficie).

3. **Eclipse**

Es difícil que unos de los soles se cruce con una de las lunas, pero cuando esto sucede, el estado de la atmósfera se altera negativamente y las partículas que normalmente debilitan a los enemigos, invierten su efecto incluso a mayor escala. En esta situación, el nivel de los enemigos se incrementa en 6/7/8/9/10 unidades y se mostrará de color rojo, por lo que todos sus estados se verán escalados por el nivel. También afecta a la velocidad de movimiento del personaje, que se verá reducida en un 25%. Además, el tiempo de espera de los Módulos del ocaso se incrementará en un 25%.

Una vez finaliza el eclipse, el nivel vuelve al original. La probabilidad de que se produzca un eclipse es del 7/6/5/4/3% cada día, respectivamente (únicamente durante el día).

DROPS

NIVEL 1 - GRUTAS DEL DESTINO

El juego comienza en el nivel 1, que corresponde a la zona de las Grutas del Destino. El personaje se despierta en un pasillo oscuro repleto de partes robóticas averiadas.

Al salir del pasillo se llega a una pequeña zona de introducción con dos plataformas, una más alta que la otra, donde se le deja al jugador experimentar con el salto. En la plataforma más alta encontramos el primer Suministrador [1].

ENTIDAD	ITEM	CANTIDAD	RATE
Suministrador [1]	Degiterio	10	100%
Suministrador [2]	Degiterio	10	100%
	Occaterio	1	5%
	Mineral fragmentado	2	25%
Suministrador [3]	Degiterio	10	100%
	Occaterio	2	5%
	Suero vital	2	100%
Suministrador [4]	Degiterio	10	100%
	Occaterio	2 (A)	10%
	Mineral fragmentado	5 (A)	25%
Suministrador [5]	Degiterio	15	100%
	Occaterio	2 (A)	10%
	Módulo de supersalto	1	100%

Suministrador [6]	Degiterio	20 (A)	80%
	Occaterio	4 (A)	25%
	Mineral fragmentado	5 (A)	25%
	Suero energético	5 (A)	50%
Suministrador [7]	Degiterio	15	100%
	Occaterio	2	100%
	Suero vital	2 (A)	50%
	Suero energético	2 (A)	50%
Suministrador [8]	Sable picapiedra	1	100%
	Porra demolerocas	1	100%
	Pistola rompemuros	1	100%
	Degiterio	30	100%
	Occaterio	5	100%
	Mineral fragmentado	5	100%

Sonidos del Juego

Los sonidos del juego se pueden categorizar en dos tipos, los relacionados con el ambiente y los que tienen que ver con la interacción con el juego.

En la parte ambiental podemos encontrar sonidos y efectos del nivel como aire, fauna y música de fondo de ambiente del nivel. Cada nivel tendrá su propia música, además de tener modificaciones distintas cuando sea día o noche y según la meteorología, representando una variación la intensidad o dificultad del nivel. También podemos encontrar música ambiental en el menú principal, un poco más activa para que llame al jugador a empezar a jugar.

En la parte de interacción podemos encontrar los ruidos de queja del personaje principal, el sonido de muerte, sonido de acción de las herramientas, las habilidades y los módulos; los sonidos de daño y muerte de cada enemigo y también los sonidos de interacción en el menú principal (al moverse y al pulsar por el menú principal y sus submenús).

Esquema de los controles

Como el juego está pensado para PC, su esquema de controles será para un teclado, aunque si se detecta un joystick se podrá utilizar como controlador también.

Tecla	Botón (Joystick)	Acción que ejecuta
W		El personaje realiza un salto.
A		Mueve al personaje hacia la izquierda.
D		Mueve al personaje hacia la derecha.
Shift		El personaje realiza una evasión.
Q		El personaje cambia de arma.
Click izquierdo		El personaje realiza un ataque con el arma equipada.

Click derecho		El personaje realiza un ataque especial con el arma equipada.
Espacio		El personaje interactúa con el entorno.
1		El personaje activa el módulo 1
2		El personaje activa el módulo 2.
3		El personaje activa el módulo 3.
E		El personaje utiliza el objeto rápido
←		El personaje cambia de objeto rápido (siguiente).
→		El personaje cambia de objeto rápido (anterior).
Ctrl + Click Izquierdo		El personaje fija la vista en un enemigo.
Rueda del ratón		El personaje cambia la vista fija entre enemigos.
Escape		Se pausa el juego y se muestran todos los menús.
F		Se muestra el inventario
R		Se muestran las estadísticas.
Z		Se muestra el mapa.

Estética del juego e interfaz de usuario

El juego va a tener una estética principalmente pixel art, con gamas de colores variables según el nivel correspondiente. Dada nuestra inexperiencia en el sector artístico intentaremos no cometer errores artísticos, pero no tenemos duda de que se nos escaparán cosas (sobre todo en la iluminación).

El HUD y la interfaz del juego se explicarán a continuación:

HUD:

El HUD se colocará en la parte superior izquierda de la pantalla, dejando el resto del espacio disponible para que el jugador pueda disfrutar mejor el juego. Aunque, para menor visibilidad la barra de vida del jefe (aparte de estar encima de la entidad como los demás monstruos) estará en la parte inferior central de la pantalla, con su nombre. Cuando una entidad reciba daño o curación saldrá un texto indicando el daño producido o curado. También habrá un texto abajo a la izquierda semitransparente que indicará cuál es la zona/el nivel actual.

- La vida se representará mediante una barra verde.
- La stamina mediante una barra amarilla debajo de la de vida. Esta barra se volverá azul unos instantes cuando se use una habilidad.
- Inmediatamente abajo se encontrarán tres círculos morados que representarán las ranuras para módulos del brazo, que indicará qué ranuras están disponibles y/o cuánto tiempo de espera hay hasta el siguiente uso del módulo correspondiente.
- A su derecha se encuentra un cuadrado naranja que indicará el ítem activable que está en ese momento preparado para su uso.

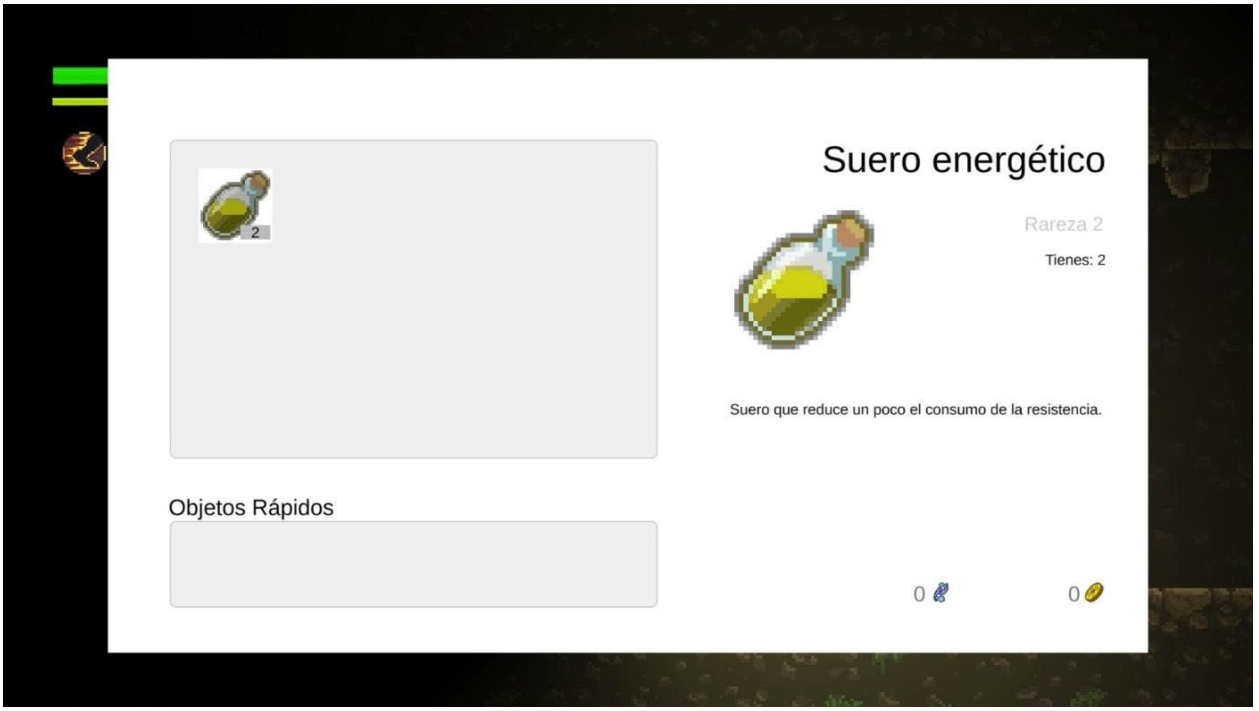
Hud del personaje detallado

Visión de la cámara principal en el juego.

HUD y Visión de la cámara principal en el juego (Boceto).

Interfaz inventario:

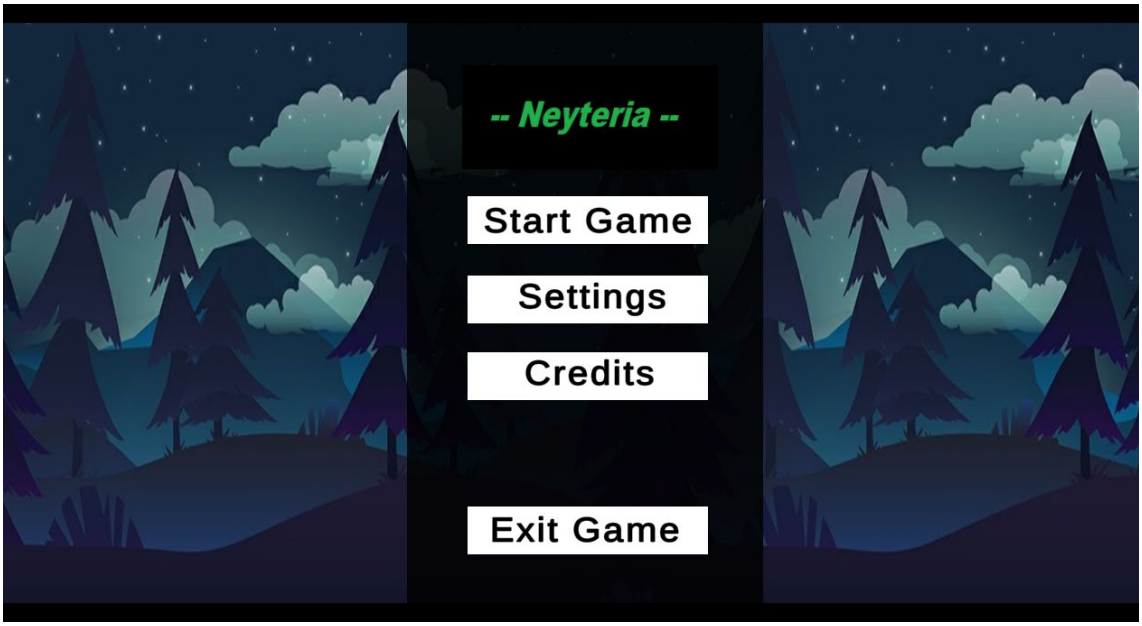
El inventario aparecerá en el centro de la pantalla, dejando el resto del espacio disponible ofreciendo una sensación más inmersiva en el juego.



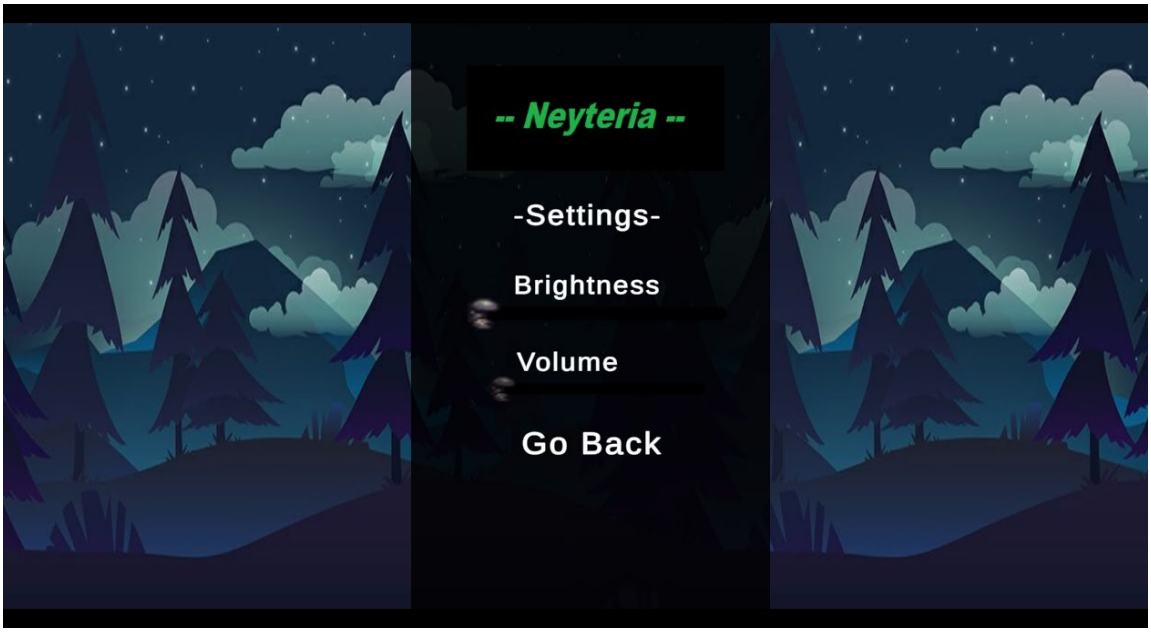
Vista del inventario en el juego.

Navegación del menú principal:

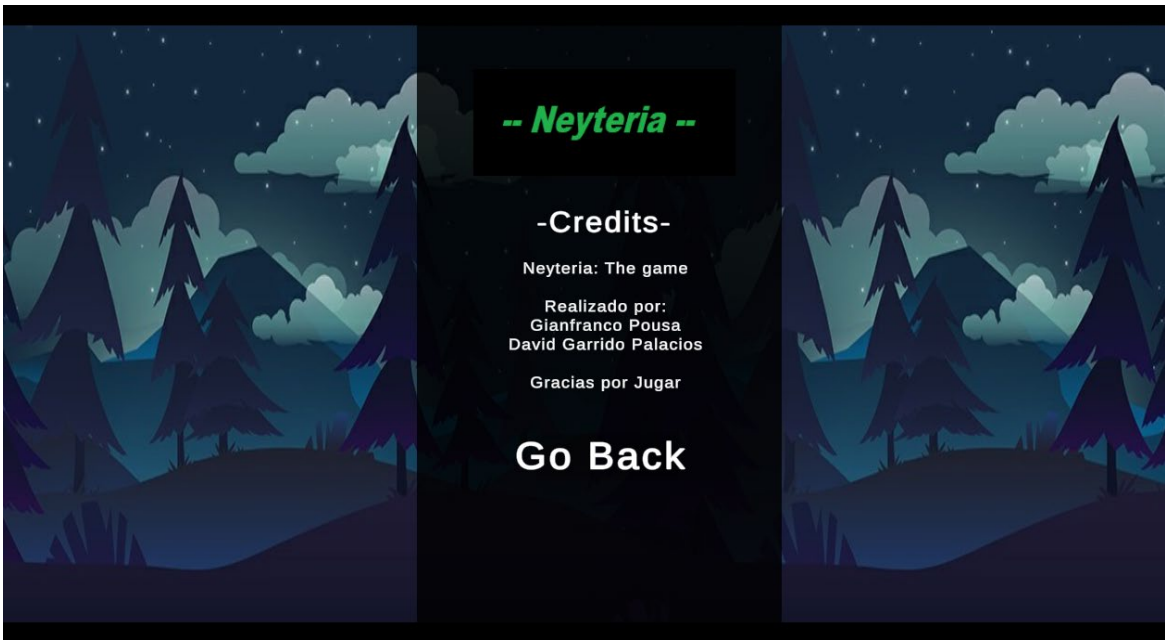
La Página Principal del juego dará acceso a la página de Start Game, Settings o Crédits. También se podrá salir del juego desde esta. Desde las demás páginas se podrá volver a la principal.



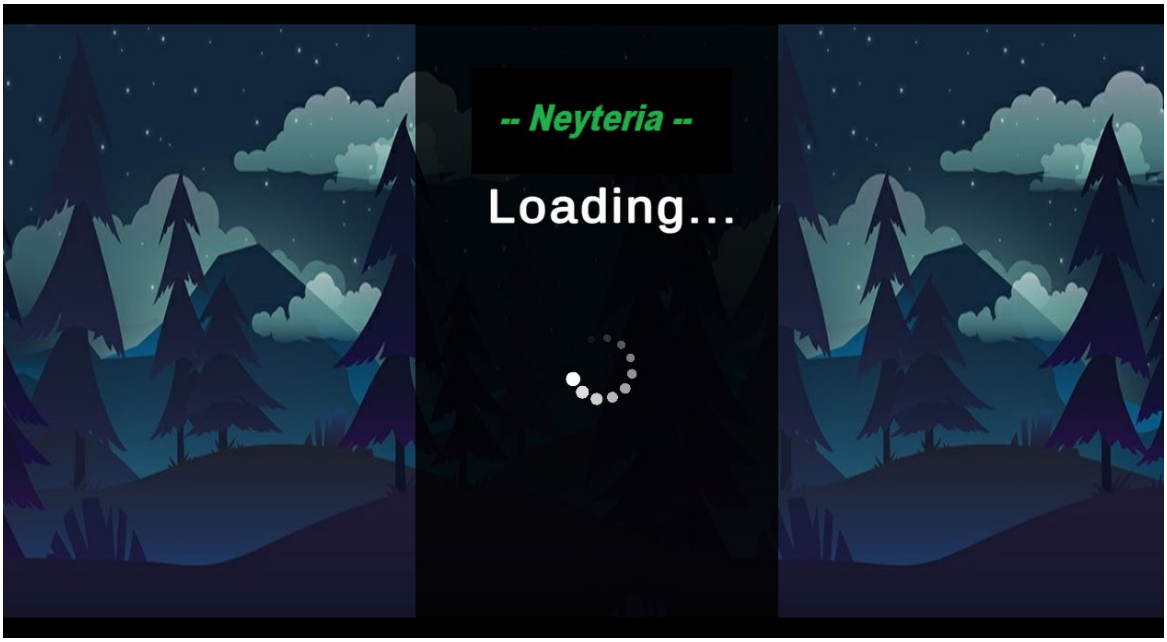
Página principal del juego.



Página de las configuraciones del juego “Settings”.



Página de los créditos del juego “Credits”.



Página de carga del juego “Start Game”.

Pantalla de muerte:

La pantalla de muerte terminará el juego y dará acceso a la Principal.

Pantalla de muerte del juego “Game Over”.

Comandos de la consola admin

Para abrir la consola en modo administrador, se debe pulsar las teclas “T + Enter” y para salir de la consola, basta con pulsar “Enter” sobre la línea de comandos. Al escribir un comando que contenga como mínimo los dos primeros términos, si se pulsa la tecla “Tabulador” aparecerá una ayuda en rojo mostrando los parámetros del comando que faltan por introducir.

COMANDO	**PARÁMETROS**	**DESCRIPCIÓN**
comm health <**cant**>	cant: - “max”: establece la salud a máximo. - [float]: otorga dichos puntos de salud.	Comando para modificar la cantidad de salud del personaje principal.
comm selfkill		Comando para matar al personaje principal
comm player <**atributo**> <**valor**>	flag: - “health”. - “max_hp”. - “stamina”. - “max_stamina”. - “attack” - “defense”. - “weight”. - “mov_speed”. - “jump_cap”. - “dash_range”. - “gasto_dash”. - “stamina_vel_rec”. - “dmg_reduc”. - “inmune”. - “stamina_decrease”. - “no_cd”. - “dash_on_air”. - “one_hit_kill”.	Comando para modificar los atributos principales del personaje principal.
comm item <**opt**> <**id**> [**cant**]	opt:	Comando para otorgar o quitar objetos al

	- “give”. - “remove”. id: id del objeto cant: cantidad del objeto. Si no se especifica: - “give” -> otorga x1 - “remove” -> quita todos	personaje principal.

Glosario

Asset: recursos que utiliza un videojuego y que forman parte de él en el momento de su creación. Los assets de un videojuego son los sprites, las animaciones, los paquetes de sonido...

Asset store: biblioteca de assets comerciales y gratuitos creados por Unity Technologies y miembros de la comunidad.

Boss: Enemigo jefe final de nivel.

Canvas: El Canvas es un componente que controla cómo se representará un grupo de elementos de la interfaz de usuario. Todos los elementos de la interfaz de usuario deben ser elementos hijos de un canvas. Es posible tener más de un canvas en una escena.

Component: Unity es un sistema basado en componentes. Para dar funcionalidad a un determinado GameObject, le adjuntas diferentes componentes. Incluso sus propios scripts son componentes.

Colliders: Los colliders son componentes integrados en Unity que brindan detección de colisiones utilizando sus diversos 'cuadros delimitadores',

Dash: Habilidad del personaje que le permite recorrer una breve distancia de manera rápida y sin recibir daño.

Gameobject: Los GameObjects son los objetos fundamentales en Unity que representan personajes, accesorios y escenarios. No logran mucho por sí mismos, pero actúan como contenedores de componentes, que implementan la funcionalidad real. Por ejemplo, un objeto Light se crea adjuntando un componente Light a un GameObject.

Gameobject empty: GameObject con el único componente Transform (obligatorio en todos los Gameobjects).

GDD: El GDD (Game Design Document) se trata de un documento síntesis de todo lo que va a ser el juego. Será el documento guía que marcará el rumbo del proceso de la creación del videojuego y este siempre se tiene que ir editando según se avanza en el desarrollo del juego. Un GDD está creado y editado por el equipo de desarrollo y es muy utilizado para organizar esfuerzos dentro de un equipo de desarrollo de videojuegos. El documento es creado por el equipo de y es utilizado como guía durante el proceso de desarrollo del juego. El desarrollador tiene que respetar el GDD durante el proceso de desarrollo.

Grid: El componente Grid almacena datos dimensionales del diseño de la cuadrícula y proporciona funciones auxiliares para recuperar información sobre la cuadrícula, como la conversión entre la ubicación de la celda y la ubicación del espacio local de los elementos dentro de la cuadrícula.

Pitch doc: El pitch doc es un documento que resume toda la información sobre el juego que quieres crear, por lo tanto, debe contener toda la información importante para convencer a nuevos colaboradores de sumarse al proyecto, y de una forma breve y concisa.

Ports: un port es una adaptación de un programa a otra plataforma.

Prefab: Los prefabs son objetos reutilizables, y creados con una serie de características dentro de la vista proyecto, que serán instanciados en el videojuego cada vez que se estime oportuno y tantas veces como sea necesario. El prefab actúa como una plantilla a partir de la cual se pueden crear nuevas instancias del objeto en la escena. Cualquier edición hecha a un prefab asset será inmediatamente reflejado en todas las instancias producidas por él.

Prefab variant: A partir de estos prefabs, se pueden crear los prefabs variants, los cuales permiten usar un prefab como base, y al partir de él, realizar cambios sin influir en el prefab original. La modificación de elementos del prefab original, producirá la actualización en los prefabs variants hechos al partir de él, siempre que esos elementos no hayan sido modificados por el prefab variant.

Scripts: un script es un pequeño programa que podemos asociar a un GameObject como si se tratara de un componente más, proporcionándole una funcionalidad concreta que definimos nosotros. estos scripts podemos escribirlos en alguno de los lenguajes que Unity soporta de forma nativa: C#, UnityScript y Boo.

Sprites: Los sprites son un conjunto de imágenes que representan a un personaje o a un objeto (o una parte de ellos) de manera gráfica y que se utiliza para poder crear cualquier efecto de movimiento o para cambiar su estado o posición en la escena.

Tileset: Un tileset es un conjunto de texturas (o tiles) reunidas en una misma imagen, una composición. Estas texturas forman las piezas gráficas que componen el escenario de un videojuego 2D: suelos, paredes, escaleras, techos, fondos, estructuras, etc.

Triggers: Un trigger es un disparador de eventos que no registra una colisión con un Rigidbody entrante. En su lugar, envía mensajes OnTriggerEnter, OnTriggerExit y OnTriggerStay cuando un cuerpo rígido entra o sale del volumen de trigger.