UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de un videojuego de mazmorras centrado en la gestión de recursos y cartas usando el motor de juegos Unity.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Puigcerver Ibáñez, Enric

Tutor/a: Carrascosa Casamayor, Carlos

Cotutor/a: Mollá Vayá, Ramón Pascual

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Hacer un videojuego de mazmorras centrado en la gestión de recursos y cartas usando el motor de juegos Unity

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Puigcerver Ibáñez, Enric

Tutores: Carlos Carrascosa Casamayor y Ramón Pascual Mollá Vayá

Curso Académico 2022 - 2023

Agradecimientos

En primer lugar, me gustaría expresar mi profundo agradecimiento a todas las personas que me han animado y apoyado durante la realización de este proyecto.

Especialmente, deseo destacar el valioso esfuerzo realizado por Carlos Carrascosa Casamayor y Ramón Pascual Molla Vaya, mis tutores en este proyecto, quienes me han brindado una enorme asistencia, así como a Pau Vidal, mi compañero y amigo, por su valiosa colaboración en este trabajo.

Por otro lado, quiero transmitir mi más sincero reconocimiento a aquellos profesores que se preocupan por sus alumnos, que los inspiran y resuelven cualquier desafío que se les presente.

También deseo agradecer a todos los compañeros y amigos que he tenido el placer de conocer y que me han acompañado durante estos cuatro años de carrera.

Finalmente, quiero manifestar mi más profundo agradecimiento a mi familia, por su apoyo y ayuda incondicional.

Resumen

Este proyecto consiste en la programación y creación de un videojuego *roguelike* y de cartas con una temática de Dragones y Mazmorras. Además de la programación, la creación de un videojuego implica tener que montar el apartado visual y que funcione en tiempo real. Aparte de la programación y aspecto visual del videojuego en sí, también se han programado herramientas para poder hacer pruebas del juego y herramientas para facilitar la creación de contenido futuro.

Palabras clave: Videojuego de mazmorras, Gestión de recursos y cartas, Unity, Dragones y Mazmorras

Resum

Aquest projecte consisteix en la programació i creació d'un videojoc de masmorres i de cartes amb una temàtica de Dracs i Masmorres. A més de la programació, la creació d'un videojoc implica muntar l'apartat visual i que funcione en temps real. A part de la programació i l'aspecte visual del videojoc en sí, també s'han programat eines per a poder fer proves del joc i eines per a facilitar la creació de contingut futur.

Paraules clau: Videojoc de masmorres, gestió de recursos i cartes, Unity, Dracs i Masmorres

Abstract

This project involves the programming and creation of a roguelike and card video game with a Dungeons and Dragons theme. In addition to programming, creating a video game involves setting up the visual aspect and making it work in real time. Apart from the programming and visual aspects of the game itself, tools have also been programmed to enable game testing and to facilitate the creation of future content.

Keywords: Roguelike, cards and resources management, Unity, Dungeons and Dragons

Tabla de contenidos

Lista de figuras

1. Introducción

El trabajo de fin de grado “Hacer un videojuego de mazmorras centrado en la gestión de recursos y cartas usando el motor de juegos Unity” es un proyecto perteneciente a los estudios de Grado en Ingeniería Informática de la Escola Tècnica Superior d’Enginyeria Informàtica (Universitat Politècnica de València), concretamente en la especialidad Computación.

La producción y el desarrollo de videojuegos han experimentado una transformación significativa a lo largo de los años, pasando de ser un recurso técnico de entretenimiento a convertirse en una de las industrias del ocio más influyentes. Tanto es así, que el Consejo de la Unión Europea incluyó en 2022 [1], a petición de España, la industria del videojuego entre los sectores creativos prioritarios.

Además, el impacto económico de esta industria es innegable. Grandes empresas invierten fuertemente en el desarrollo y la promoción de videojuegos y también, los estudios independientes han encontrado un gran éxito en este sector. Según recoge la Asociación Española de Videojuegos (AEVI) en su Anuario del 2022 [1], el consumo de videojuegos en España creció un 12,09%, facturando 2.012 millones de euros.

Los avances tecnológicos han permitido la creación de experiencias interactivas cada vez más emocionantes, que han atraído a millones de personas de todo el mundo, generando una enorme cantidad de seguidores. En España, según el Anuario 2022 de AEVI [1], la comunidad de usuarios es de 18,2 millones de videojugadores, de los cuales, el 53% son hombres y el 47% son mujeres.

Asimismo, los videojuegos han demostrado tener un poderoso impacto cultural y social. Han creado comunidades virtuales en línea donde los jugadores pueden conectarse, competir y colaborar entre sí. [2]

En resumen, la industria del videojuego ha evolucionado enormemente, convirtiéndose en una fuerza dominante en el ámbito del entretenimiento. Su influencia se extiende a nivel económico, cultural y social, y continúa creciendo a medida que se exploran nuevas tecnologías y se expanden los límites de la creatividad. [3]

Los videojuegos de mazmorras son un género popular dentro de la industria del videojuego. Se caracterizan por ambientarse en mazmorras llenas de peligros y desafíos, donde los jugadores deben explorar, combatir enemigos y resolver acertijos para avanzar en la historia. [4]

Los videojuegos de mazmorras han experimentado un éxito sin precedentes en los últimos años, convirtiéndose en uno de los favoritos de los estudios independientes. [4]

Además, ha habido una tendencia en la industria de los videojuegos a fusionar los juegos de cartas con los de mazmorras, dando lugar a una nueva categoría de juegos que combina lo mejor de ambos géneros. [5]

1.1. Motivación

Comenzaré diciendo que soy un apasionado de los videojuegos y desarrollar un videojuego lo más perfecto y divertido posible ha sido un reto que he vivido con gran entusiasmo.

La idea de realizar un videojuego en el TFG surgió mientras cursaba, en el primer cuatrimestre de cuarto, la asignatura *Desarrollo de videojuegos 3D* con el profesor Ramón Pascual Molla Vaya y poco a poco se fue materializando.

Me ilusionaba enfrentarme a los desafíos y contratiempos diarios que surgen en el proceso de creación de un videojuego (problemas técnicos, errores en el código, fallos en el diseño, falta de recursos, …) y buscar y encontrar una solución plausible ha sido muy gratificante, dado que siempre me ha gustado la resolución de problemas.

Las prácticas las he hecho en la empresa *Apolo Kids*, una empresa dedicada a desarrollar herramientas digitales para que los niños puedan aprender mientras se divierten jugando. Estas prácticas han sido una experiencia muy positiva y me han reforzado la buena elección de este proyecto.

1.2. Objetivos

El objetivo del proyecto es desarrollar un videojuego de mazmorras centrado en la gestión de recursos y cartas. Para la consecución de este objetivo se han tomado como referencia los juegos ***The Binding of Isaac*** y ***Slay the Spire***, así como la utilización del motor de desarrollo Unity.

1.3. Metodología

Se ha optado por utilizar una metodología ágil en el proceso de trabajo con el fin de lograr una entrega más eficiente de este proyecto.

Entre las metodologías ágiles más populares se encuentran Scrum, Kanban, Extreme Programming (XP), Lean Software Development (LSD), Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD) y Crystal. [6]

Dado que un videojuego consta de diversos módulos separados, se ha elegido FDD [7], también llamada metodología de desarrollo basado en funcionalidades. Esta metodología se caracteriza por su enfoque centrado en el cliente y se destaca por sus iteraciones cortas y lanzamientos frecuentes.

El uso de la metodología FDD ha permitido tener una mayor flexibilidad y adaptabilidad a medida que se avanzaba en el proyecto. Además, ha ayudado a realizar entregas periódicas y obtener retroalimentación temprana de los tutores, facilitando las mejoras y ajustes necesarios en el desarrollo del videojuego.

Es importante destacar que esta metodología está diseñada para el trabajo en equipo, y dado que este proyecto es individual, no se han seguido todas las características del método de manera estricta.

La metodología FDD se compone de 5 pasos. Los tres primeros se hacen al inicio del proyecto.

1) Desarrollar el modelo general
2) Elaborar la lista de funcionalidades
3) Planificar funcionalidades
4) Diseñar funcionalidades
5) Desarrollar funcionalidades

Los dos últimos pasos se repiten de forma iterativa para desarrollar y completar todas las funcionalidades del proyecto. Tener en cuenta que el quinto paso no solo comprende desarrollar la nueva funcionalidad desde cero, sino también integrar al resto del proyecto sin que haya ningún error tras un periodo de testeo.

Las reuniones con los tutores del TFG se llevaron a cabo en intervalos variables de 1 a 3 semanas. Durante estas reuniones, se presentaron los resultados finales de la funcionalidad programada para la iteración y se recopiló el feedback de los tutores para realizar modificaciones en caso necesario. Luego se iniciaba la siguiente iteración hasta completar el proyecto.

1.4. Estructura

En este apartado se detalla la estructura de este trabajo. En el primer capítulo de introducción se exponen los objetivos y la metodología utilizada. En el segundo capítulo se detalla en qué situación se encuentra el tema elegido, los videojuegos de cartas, los autobattlers, los de mazmorras, los de mazmorras centrados en gestión de recursos y se hace un estudio de los tres principales videojuegos de referencia. En el tercer capítulo se realiza un análisis del problema donde se detallan los requisitos funcionales y no funcionales y se analiza el marco legal. En el cuarto capítulo muestra el diseño del proyecto y la tecnología utilizada. El quinto capítulo expone el desarrollo de la solución propuesta. En el sexto capítulo se detalla la etapa de la implantación de la solución. En el séptimo capítulo se presentan con que herramienta se han hecho las pruebas realizadas para la verificación del trabajo. El octavo capítulo se expone una conclusión donde se explica si se ha alcanzado el objetivo planteado y se relaciona este trabajo con los estudios cursados. En el noveno capítulo se enumeran posibles trabajos futuros. Finalmente, el décimo capítulo muestra todas las referencias bibliográficas.

1.5. Colaboraciones

Este proyecto inicialmente tenía previsto utilizar el TFG de un compañero, Pau Vidal Ramón. Sin embargo, debido a motivos personales, Pau no ha podido completar su parte del proyecto para esta entrega, por lo que esa parte estará ausente en el proyecto.

2. Estado del arte

En este apartado se van a explicar y analizar los diferentes géneros de videojuegos a los que pertenece el proyecto. Además de explicar más profundamente los videojuegos en los que se inspira este proyecto.

2.1. Videojuegos de cartas

Los juegos de cartas podrían haber surgido en los siglos IX, XI o XII. Existen diferentes versiones que sugieren que los juegos de cartas podrían haberse originado en China, Persia o India. [8]

Es muy probable que los naipes llegaran a Europa desde Oriente en el siglo XIV, introducidos por los árabes a través de los reinos cristianos de España. También se ha mencionado la posibilidad de que los cruzados hayan contribuido a su difusión. [8]

Inicialmente, estaba destinado principalmente a la alta sociedad, ya que las cartas se fabricaban a mano y su decoración podía ser extremadamente detallada y habilidosa.

A finales del siglo XV, con la llegada de las imprentas y la creciente popularidad de los juegos de cartas, el diseño de las barajas alcanzó un estándar más uniforme. En Europa, se utilizaban cuatro tipos de barajas: alemana, suiza, francesa y latina, que han prevalecido hasta la actualidad. [8]

Además de los cuatro mazos tradicionales, también existen muchos otros tipos de barajas de cartas hoy en día.

En definitiva, a lo largo de los siglos, los juegos de cartas se han establecido como un pasatiempo perdurable que, gracias a los avances tecnológicos, también se ha trasladado al ámbito de los videojuegos. [9]

Los videojuegos de cartas son una categoría popular en la industria del *gaming*. Estos juegos combinan elementos estratégicos y de recolección de cartas en un entorno virtual. Los jugadores construyen mazos de cartas, cada una con habilidades y atributos únicos, y luego emplean esas cartas para competir contra otros jugadores o enfrentar desafíos en el juego.

Algunos ejemplos de videojuegos de cartas son [10]:

- ***Hearthstone***[1]: Lanzado en 2014 y creado por la empresa *Blizzard Entertainment*, es uno de los juegos de cartas más populares. Los jugadores construyen mazos basados en personajes y elementos del universo fantástico de *Warcraft*, universo de ficción de los juegos de la serie *Warcraft* y *World of Warcraft*, donde aparecen diferentes planos de existencia, nuevos planetas y nuevas razas.

Figura 2.1: *Hearthstone.* Fuente: HardwareZone Singapore, 2014.

- ***Magic: The Gathering Arena***[2]***:*** Es la versión digital del famoso juego de cartas físico *Magic: The Gathering*. Fue lanzado en 2018. Los jugadores construyen mazos poderosos y estratégicos para enfrentarse en partidas multijugador o para participar en eventos especiales.

Figura 2.2: *Magic: The Gathering Arena.* Fuente: Marca, 2021.

- ***Gwent: The Witcher Card Game***[3]: Basado en el popular juego de rol *The Witcher*, *Gwent* es un juego de cartas táctico de 2017 en el que los jugadores utilizan mazos personalizados para enfrentarse en batallas estratégicas.

Figura 2.3: *Gwent.* Fuente: Steam

[1] Hearthstone - Página oficial. https://hearthstone.blizzard.com/es-es
[2] Magic – Página Oficial. https://magic.wizards.com/es/mtgarena
[3] Gwent – Página Oficial. https://www.playgwent.com/en

- ***Legends of Runeterra*[4]:** Desarrollado por *Riot Games* en 2020, este juego de cartas se basa en el universo de *League of Legends*. Los jugadores construyen mazos utilizando campeones y cartas temáticas, y compiten en partidas estratégicas uno contra uno.

Figura 2.4: *Legends of Runeterra.* Fuente: Pocket Tactics, 2020.

Estos son solo algunos ejemplos, pero existen muchos otros videojuegos de cartas disponibles en diferentes plataformas, cada uno con su propio estilo y mecánicas de juego.

2.1.1. Juegos *Auto Battlers*

El género de videojuegos de *Auto Battlers* es un subgénero de los juegos de cartas y juegos de estrategia. [11]

En los juegos de este tipo, los jugadores posicionan sus personajes o cartas en un tablero de batalla cuadriculado durante una fase de preparación. Una vez finalizada esta fase, los personajes combaten automáticamente contra los personajes del equipo contrario sin recibir instrucciones adicionales de ningún usuario.

Fue en enero de 2019 cuando el juego ***Dota Auto Chess*** [5]popularizó este género.

Figura 2.5: *Dota Auto Chess.* Fuente: Polygon, 2019.

[4] Legends Of Runaterra - Página Oficial. https://playruneterra.com/es-es/
[5] Dota Auto Chess – Steam. https://steamcommunity.com/sharedfiles/filedetails/?id=1613886175

Entre los juegos más populares de juegos del género *Auto Battler* se destacan los siguientes:

- ***Teamfight Tactics***[6]***:*** Desarrollado y publicado por *Riot Games* en 2019, es un modo de juego de *League of Legends* inspirado en el estilo de los *Auto Battlers*. El jugador se enfrenta a siete oponentes en una competición para construir un equipo que luchará en su nombre.

Figura 2.6: *Teamfight Tactics.* Fuente: Vandal, 2022.

- ***Hearthstone Battlegrounds***[7]: En 2019, la compañía *Blizzard* adaptó su juego *Hearthstone* a este género al crear un modo de juego llamado *Hearthstone Battlegrounds*.

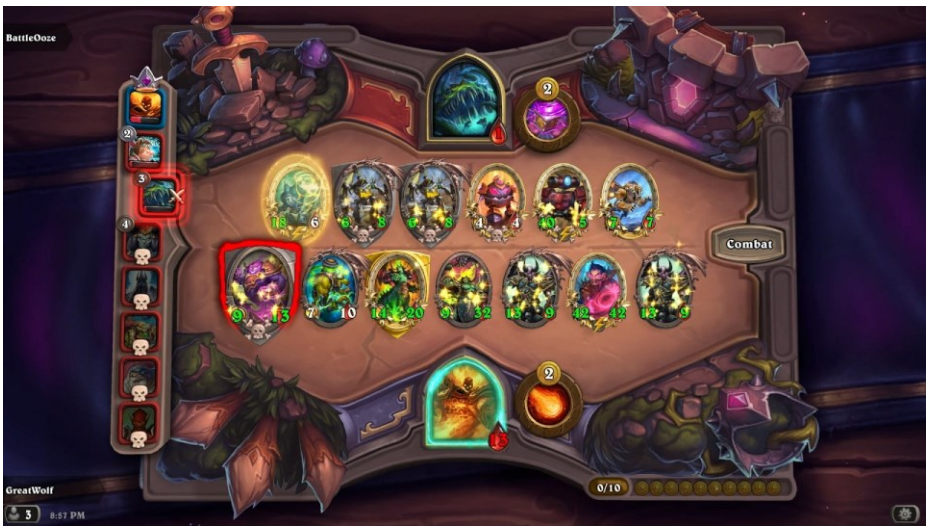

Figura 2.7: *Hearthstone Battlegrounds.* Fuente: Newsweek, 2019.

[6] TFT – Página Oficial. https://teamfighttactics.leagueoflegends.com/es-es/
[7] Hearthstone – Página Oficial. https://hearthstone.blizzard.com/en-us/battlegrounds/

2.2. Videojuegos de mazmorras

Los videojuegos de mazmorras, o también llamados videojuegos *roguelike*, son un subgénero de los juegos de rol basados en *Rogue*, un videojuego con gráficos ASCII desarrollado por Michael Toy, Glenn Wichman y Ken Arnold en el año 1980. [12]

En *Rogue*[8], el jugador se adentraba en una mazmorra creada por un poderoso mago *Rodney*, y en su camino, se enfrentaba a numerosos monstruos y se encontraba valiosos tesoros, todo con el objetivo de recuperar el *Amuleto de Yendor*. Este juego volvía a configurar sus elementos al inicio de cada partida. Así, aunque con más experiencia, en cada nueva partida el jugador se encontraba retos diferentes generados por procedimientos.

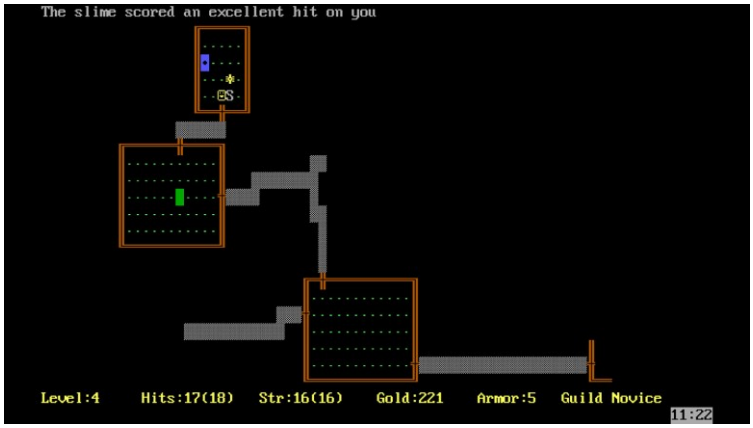

Figura 2.8: *Rogue.* Fuente: Steam.

Debido a la proliferación de numerosas variaciones en el género de videojuegos de mazmorras, un grupo de desarrolladores (entre ellos Ido Yehieli, Radomir Dopieralski i Jeff Lait) debatieron y establecieron, en la Conferencia Internacional de Desarrollo de videojuego de mazmorras 2008, celebrada en Berlín, una serie de pautas que definirían los requisitos para denominar a un juego como *roguelike* [13]. Estas directrices, conocidas como la "*Interpretación de Berlín*", se dividían en dos listas [14]:

1. Factores de alto valor
   - Generación aleatoria de mazmorras
   - Muerte permanente
   - Videojuego basado en turnos
   - Estructuración del mapa por casillas, así como el movimiento y combate.
   - Alta dificultad y complejidad
   - Gestión de recursos para sobrevivir
   - Hack and Slash
   - Elementos de exploración
2. Factores de bajo valor
   - Un solo jugador
   - Los monstruos son similares al jugador
   - Reto táctico
   - Caracteres ASCII

[8] Rogue – Steam. https://store.steampowered.com/app/1443430/Rogue/

- Se exploran mazmorras
- Describe el estado del jugador y del juego con números

Cuantas más reglas cumpliera un juego, más se podría considerar como un *roguelike*. John Harris, de Game Set Watch, ilustró esto al utilizar estos criterios para asignar puntuaciones numéricas a algunos juegos considerados videojuegos de mazmorras. *Linley's Dungeon Crawl* y *NetHack* alcanzaron la puntuación más alta, obteniendo 57,5 puntos de los 60 posibles. Por otro lado, juegos como *Toe Jam & Earl* y *Diablo*, solo obtuvieron aproximadamente la mitad de los puntos [14].

En su artículo "*Screw the Berlin Interpretation!*" de 2013 [16], Darren Gray critica la Interpretación de Berlín, acusándola de ser inexacta, obsoleta y poco representativa de un género vibrante y abierto. En particular, ridiculiza características como el uso de ASCII y los dungeons como irrelevantes para un género que tradicionalmente prioriza la mecánica de juego por encima de la estética o la ambientación.

En la actualidad, los videojuegos de mazmorras han evolucionado y no cumplen casi ninguna de esas reglas de la Interpretación de Berlín. Sin embargo, existen algunas características principales que suelen estar presentes en este tipo de juegos:

- Los mapas o niveles de mazmorra se generan aleatoriamente, siendo, generalmente, cada *run* o recorrido diferente.
- En la mayoría de los casos, suelen ser de un solo jugador.
- La muerte del personaje del jugador es permanente.

Así, cada partida es única y generada de manera aleatoria, lo que significa que los jugadores se enfrentan a nuevos desafíos y sorpresas en cada intento. A medida que avanzan, adquieren experiencia y conocimiento sobre el juego, lo que les ayuda a tomar decisiones más informadas y estratégicas. Sin embargo, nunca saben exactamente qué obstáculos encontrarán en cada nueva aventura, lo que aumenta la emoción y el factor de sorpresa. Esta incertidumbre y la necesidad de adaptarse y superar los desafíos en tiempo real hacen que los *roguelikes* sean experiencias emocionantes y adictivas para muchos jugadores. [17]

Dentro de *roguelike* existe un subgénero llamado *roguelite*. La principal diferencia está en la progresión del jugador. En los *roguelike*, cuando mueres, pierdes todo tu progreso por completo y debes comenzar desde cero en la siguiente partida. En cambio, en los *roguelite*, conservas ciertos elementos entre partidas, como armas, habilidades, objetos u otros elementos que te ayudan en los intentos siguientes. [18]

Entre los juegos más destacados de mazmorras encontramos los siguientes:

- ***The Binding of Isaac***:[9] Es un juego independiente creado por Edmund McMillen en 2011. Este título estableció mecánicas, dinámicas y estéticas que se han mantenido desde entonces, convirtiéndolo en un referente en la industria. Incluso sus propios creadores han lanzado expansiones sucesivas a lo largo de los años, lo que demuestra la relevancia y vigencia del juego.

[9] The Binding of Isaac – Steam.
https://store.steampowered.com/app/113200/The_Binding_of_Isaac/

Este juego, inspirado en la historia bíblica del sacrificio de Isaac, es el viaje de un bebé para intentar escapar de su madre y de un sótano con niveles generados aleatoriamente repleto de demonios y monstruos.

Figura 2.9: *The Binding of Isaac.* Fuente: Captura de pantalla del juego.

- ***Hades***[10]: En este videojuego, que salió al mercado en 2020, *Zagreus*, el hijo de *Hades*, se embarca en un intento de escapar del Inframundo y alcanzar el Monte Olimpo. Aunque *Zagreus* morirá repetidamente, el jugador puede utilizar el tesoro acumulado para mejorar sus atributos o desbloquear nuevas armas y habilidades.
  Cuando el jugador fracasa al escapar, regresa a un lugar común donde el juego aprovecha para narrar historias de la mitología griega clásica.
  Hades recibió numerosos reconocimientos. Fue nombrado *Juego del Año* en varias publicaciones, incluyendo la lista de los 10 mejores juegos de 2020 de *Time*. Además, obtuvo varios premios, entre ellos el galardón al Mejor Juego Independiente en los *Golden Joystick Awards 2020* y en *The Game Awards 2020*.

Figura 2.10: *Hades.* Fuente: Steam.

[10] Hades – Steam. https://store.steampowered.com/app/1145360/Hades/

- ***Enter the Gungeon***[11]: Este juego de mazmorras y tiroteos fue lanzado en 2016. Sigue la historia de un grupo de marginados arrepentidos que buscan la absolución personal a través de disparos, saqueos y acrobacias, todo con el objetivo de alcanzar el tesoro supremo de la legendaria *Armazmorra*: el arma capaz de cambiar el pasado. El juego presenta diversas habitaciones a las que se puede acceder mediante ascensores y algunos secretos, cada una de ellas poblada por una serie de enemigos aleatorios y diversas amenazas.

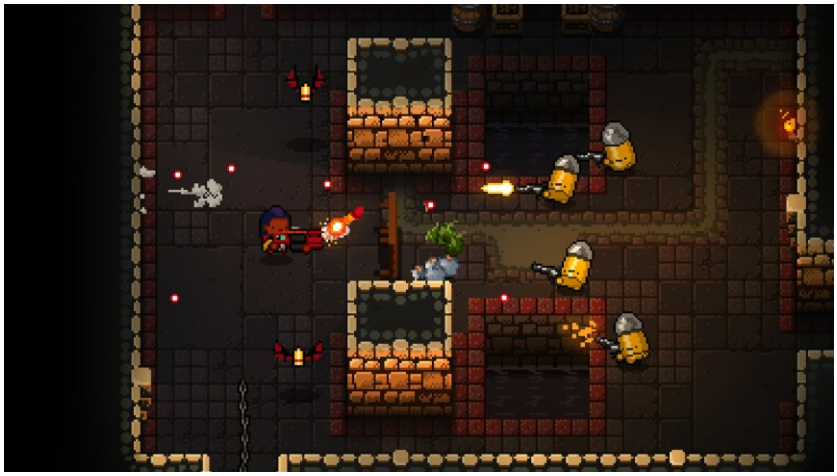

Figura 2.11: *Enter the gungeon.* Fuente: Steam.

- ***Moonlighter***[12]. Este videojuego, desarrollado por un estudio español en 2018, se enmarca dentro del género *roguelite*. *Moonlighter* consiste en realizar incursiones en mazmorras para obtener objetos que luego se venden en la tienda a otros héroes. De esta manera, las mecánicas de acción y combate se entrelazan con elementos de gestión y comercio.

Figura 2.12: *Moonlighter.* Fuente: Steam.

[11] Enter the Gungeon – Steam. https://store.steampowered.com/app/311690/Enter_the_Gungeon/
[12] Moonlighter – Steam. https://store.steampowered.com/app/606150/Moonlighter/

- ***Rogue Legacy 1***[13]: Es un juego publicado en 2013. Tiene como objetivo explorar una mazmorra generada de forma aleatoria, derrotar a cuatro jefes en cada uno de los cuatro entornos únicos de la mazmorra y por último, enfrentarse al jefe final.

Figura 2.13: *Rogue Legacy 1.* Fuente: Steam.

- ***Rogue Legacy 2***[14]: Fue lanzado en 2022. En cada partida el jugador se sumerge en un castillo familiar generado aleatoriamente. Cada vez que el héroe muere, el jugador debe elegir a uno de sus descendientes para comenzar de nuevo.

Figura 2.14: *Rogue Legacy 2.* Fuente: Polygon, 2022.

[13] Rogue Legacy 1 – Steam. https://store.steampowered.com/app/241600/Rogue_Legacy/
[14] Rogue Legacy 2 – Steam. https://store.steampowered.com/app/1253920/Rogue_Legacy_2/

- ***Darkest Dungeon***[15]: Fue lanzado en 2020. Es un juego de rol gótico en mazmorras y por turnos, donde el jugador recluta, entrena y lidera a un equipo de héroes imperfectos a través de bosques enmarañados, laberintos olvidados y criptas en ruinas.

Figura 2.15: *Darkest Dungeon.* Fuente: Steam.

2.3. Videojuegos de mazmorras centrados en la gestión de recursos y cartas

En los últimos años, ha habido una tendencia en la industria de los videojuegos a fusionar los juegos de cartas con los *roguelike*, dando lugar a una nueva categoría de juegos que combina lo mejor de ambos géneros.

Estos juegos fusionados presentan mecánicas de construcción de mazos de cartas en un entorno *roguelike*. Los jugadores exploran mazmorras generadas de forma procedimental, enfrentándose a enemigos y desafíos mientras construyen y mejoran su mazo de cartas.

En este tipo de juegos, los jugadores van obteniendo nuevas cartas durante su progresión en la mazmorra, cada una representando habilidades, hechizos, objetos u otras acciones que pueden utilizar en su aventura. A medida que avanzan, los jugadores pueden mejorar su mazo, añadir nuevas cartas y optimizar su estrategia para enfrentarse a los desafíos que vayan surgiendo.

Este enfoque ofrece una experiencia de juego única y estratégica. Los jugadores deben tomar decisiones tácticas sobre qué cartas incluir en su mazo, cómo combinarlas y cuándo utilizarlas, ya que cada elección puede tener un impacto significativo en su éxito o fracaso en la mazmorra.

Estos *roguelike* de construcción de mazos ofrecen una gran variedad de combinaciones y enfoques, lo que aumenta su rejugabilidad y permite a los jugadores experimentar diferentes estrategias en cada partida. Además, suelen contar con sistemas de progresión, desbloqueo de

[15] Darkest Dungeon – Steam.
https://store.steampowered.com/app/262060/Darkest_Dungeon/

nuevas cartas y elementos que permiten a los jugadores aumentar su poder y diversificar aún más su estilo de juego.

En resumen, la fusión de juegos de cartas y *roguelike* ha dado lugar a una nueva categoría de juegos que combina la estrategia y la construcción de mazos con la aleatoriedad y el desafío de los *roguelikes*. Estos juegos ofrecen una experiencia única y emocionante para los jugadores que disfrutan de la planificación táctica, y la diversidad en sus partidas. [5]

Entre los juegos que combinan el género de los juegos de cartas con las características de un roguelike destacamos los siguientes:

- ***Slay the Spire***[16]: Es uno de los juegos más populares de este género. Este juego, lanzado en 2019, está ambientado en una torre de fantasía.

Figura 2.16: *Slay the Spire.* Fuente: Arkansas Online, 2020.

- ***Monster Train***[17]: Este juego de 2020 tiene lugar en un tren camino al infierno.

Figura 2.17: *Monster Train.* Fuente: Steam.

[16] Slay the Spire – Steam. https://store.steampowered.com/app/646570/Slay_the_Spire/
[17] Monster Train – Steam. https://store.steampowered.com/app/1102190/Monster_Train/

- ***Griftlands***[18]: Es un videojuego, creado en 2021, de acción, aventuras, donde el jugador lucha contra enemigos en un mundo de ciencia ficción en ruinas.

Figura 2.18: Griftlands. Fuente: Epic Games.

- ***Loop Hero***[19]: Es un juego desarrollado en 2021. Su héroe debe explorar un mundo en bucle generado procedimentalmente. El objetivo principal es derrotar al Señor de la Oscuridad y restaurar el mundo a su antigua gloria.

Figura 2.19: *Loop Hero*. Fuente: Steam.

- ***Indies' Lies***[20]: Se trata de un juego lanzado en 2023 en el que el jugador explora un mundo de fantasía medieval.

Figura 2.20: Indies'Lies. Fuente: Steam.

[18] Griftlands – Steam. https://store.steampowered.com/app/601840/Griftlands/
[19] Loop Hero – Steam.: https://store.steampowered.com/app/1282730/Loop_Hero/
[20] Indies' Lies – Steam.: https://store.steampowered.com/app/1638390/Indies_Lies/

- ***Stacks: Space***[21]: Es un juego de temática espacial.

Figura 2.21: *Stacks Space.* Fuente: Google Play Store.

- ***Hellcard***[22]: Fue lanzado en 2023. Es un juego de cartas *roguelike* cooperativo. Puede jugarse tanto en modo individual como en modo multijugador. El juego tiene lugar en las mazmorras de papel del *Libro de los Demonios*.

Figura 2.22: *Hellcard.* Fuente: Steam.

[21] Stacks:Space – Steam. https://store.steampowered.com/app/2302160/StacksSpace/=US
[22] Hellcard – Steam. https://store.steampowered.com/app/1201540/HELLCARD/

2.4. Videojuegos referentes para este proyecto

Para este proyecto se van a tomar tres de estos juegos como referencia. Estos juegos son ***The Binding of Isaac***, ***Slay the Spire*** y ***Teamfight Tactics.***

2.4.1. The Binding of Isaac

Información general

- Desarrollador y diseñador: Edmund McMillen
- Distribuidor: Edmund McMillen y Florian Himsl
- Compositor: Danny Baranowsky
- Plataformas: Mac OS X, Microsoft Windows, Xbox One, PS4, PSVita, Wii U, Nintendo Switch, Nintendo 3DS
- Fecha de lanzamiento: 28 de septiembre de 2011
- Género: Videojuego de mazmorras
- Número de jugadores: 1

Características de juego

Este juego consiste en el viaje de un niño llamado Isaac, que intenta escapar de su madre y de un sótano con niveles generados aleatoriamente lleno de demonios y monstruos. Está inspirado en la historia bíblica del sacrificio de Isaac.

Las mazmorras son habitaciones rectangulares interconectadas mediante puertas situadas en el centro de las paredes. Al avanzar a una nueva habitación con monstruos, las puertas se cierran y es necesario eliminar a todos los monstruos para que se abran. La mazmorra siempre acaba en la habitación del jefe. Al derrotar al jefe, se podrá acceder a la siguiente mazmorra (piso) a través de una trampilla, y una vez se haya avanzado, no será posible regresar a la mazmorra anterior.

Figura 2.23:*The Binding of Isaac.* Fuente: Captura de pantalla del videojuego.

Isaac principalmente utiliza sus lágrimas como forma de ataque. A lo largo del juego, el jugador encontrará objetos recolectables que serán de gran ayuda. Estos objetos suelen obtenerse como recompensas al completar habitaciones, dentro de cofres, en la tienda o

generados aleatoriamente en el nivel. Entre estos objetos se encuentran las monedas, las bombas, las llaves y los corazones. Las monedas pueden utilizarse para comprar en tiendas o en ciertas máquinas. Las bombas se emplean para causar daño a los monstruos. Las llaves permiten abrir puertas cerradas o cofres. Y los corazones permiten recuperar la salud del jugador.

Características que se adaptarán al proyecto

El sistema de generación del terreno y la forma de las habitaciones del presente proyecto se basará en el videojuego ***The Binding Of Isaac***.

2.4.2. Slay the Spire

Información general

- Desarrollador: Mega Crit Games
- Editor: Mega Crit Games
- Distribuidor: HumbleBundle
- Plataformas: PC, PlayStation 4, Nintendo Switch, Xbox One, dispositivos iOS y Android.
- Fecha de lanzamiento: 23/01/2019 (PC), 21/05/2019 (PS4), 06/06/2019 (NSW), 14/08/2019 (XBO), 13/06/2020 (IPH, IPD), 03/02/2021 (AND)
- Género: *Roguelike*, Juego de construcción de mazos
- Número de jugadores: 1

Figura 2.24: *Slay the Spire.* Fuente: Nintendo.

Características de juego

El objetivo del juego es llegar hasta lo más alto de una torre de tres niveles generados de manera aleatoria y derrotar al jefe final. A lo largo de la ascensión por los diferentes pisos, el jugador se encontrará con enemigos, jefes y eventos especiales que también aparecen de forma aleatoria.

En este juego, el jugador va obteniendo nuevas cartas durante su progresión en la mazmorra, cada una de ellas representando habilidades u otras acciones que se pueden utilizar en la partida.

Reliquias

Las reliquias son objetos que otorgan habilidades durante toda la partida, y su efecto puede estar relacionado con el tipo de cartas que se utilicen.

Cartas

Cada carta en el mazo del jugador tiene un costo que variará según la utilidad de la carta. En cada turno, el jugador tendrá tres puntos para gastar en cartas.

Las cartas pueden ser de habilidad, poder o ataque, y dependiendo de su tipo, algunas reliquias beneficiarán más o menos.

Las cartas de ataque son una herramienta fundamental para infligir daño a los oponentes en el juego. Están diseñadas para causar daño directo a los enemigos, y algunas de ellas incluso cuentan con efectos especiales que pueden debilitar a los enemigos o eliminarlos por completo.

Las cartas de habilidad proporcionan una mayor defensa al personaje del jugador, permitiéndole bloquear los ataques enemigos. Sin embargo, estas cartas no solo se limitan a la defensa, sino que también pueden tener un efecto ofensivo al brindar un beneficio adicional, como un aumento de fuerza. Estas cartas solo pueden ser utilizadas una vez durante un combate.

Las cartas de poder tienen la capacidad de mejorar al personaje o proporcionar algún tipo de habilidades especiales.

El jugador tiene la posibilidad de aumentar el número de cartas al ganar un combate. También, es posible obtener cartas a través de eventos especiales o comprándolas en la tienda del comerciante.

Sin embargo, todas aquellas cartas que no brindan un beneficio directo al jugador se incorporan al mazo debido a los efectos causados por los enemigos o al uso de cartas específicas. Además, ciertas reliquias pueden ser la causa de agregar este tipo de cartas al mazo.

Oro

Este recurso se utiliza para comprar en la tienda del comerciante.

Al comenzar el juego, el jugador recibe 99 monedas de oro, aunque puede aumentar la cantidad al vencer en combates. Además, existe la posibilidad de obtener oro a través de ciertas cartas, al completar eventos especiales o gracias a las bonificaciones de algunas reliquias.

Pociones

Son elementos de un solo uso que dan bonificaciones.

Energía

La energía es un recurso vital para que el jugador pueda jugar cartas. El costo de energía puede variar de 0 a 3 puntos. Cada personaje tiene la capacidad de almacenar hasta un máximo de tres puntos de energía. Además, es posible aumentar esta limitación mediante el uso de reliquias o cartas especiales.

Personajes

Al inicio de la partida, el jugador puede escoger entre cuatro personajes, el blindado, la silenciosa, el defectuoso y la vigilante. Cada uno de ellos tiene su propio HP (puntos de vida), reliquia inicial y mazo de cartas.

Habitaciones

En el mapa se pueden encontrar los siguientes tipos de habitaciones:

- Monstruo: Enemigo contra el que se combate. Si el jugador lo vence podrá elegir una nueva carta, obtener oro o una poción.
- Monstruo élite: Es un monstruo más fuerte y le da reliquias al jugador cuando es derrotado.
- Cofre: En esta sala se encuentra un cofre que, al abrirlo, se obtiene una valiosa reliquia.
- Tienda: Lugar donde el jugador puede adquirir cartas, reliquias o remover una carta de su mazo a cambio de una determinada cantidad de oro.
- Interrogante: Puede ser, un monstruo, una tienda, un cofre o un evento.
- Los eventos: Son salas especiales donde el jugador puede obtener bonificaciones como reliquias, oro o aumentar su vida, aunque también existe la posibilidad de enfrentar maldiciones como perder vida o perder oro.
- Campamento: Es un lugar donde el jugador puede tomar un descanso y recuperar vida o forjar una carta para mejorarla.
- Jefe: Hay un jefe en cada uno de los tres pisos, siendo el jefe del tercer piso el más poderoso.
- Jefe final: Una vez que el jugador completa el juego con los tres personajes, se desbloquea el jefe final.

Características que se adaptarán al proyecto

El sistema de gestión de cartas, reliquias y oro de este proyecto se basará en el de ***Slay The Spire***.

2.4.3. Teamfight Tactics

Información general

- Desarrollador: Riot Games
- Distribuidor: Riot Games
- Productor: Dax Andrus
- Plataformas: Microsoft Windows, macOS. Android, iOS
- Fecha de lanzamiento: 26 de junio de 2019
- Género: *Auto Battler*
- Número de jugadores: 8

Figura 2.25: *Teamfight Tactics*. Fuente: Uptodown.

Características de juego

El jugador tendrá que reunir un ejército de sus campeones favoritos para que realicen los combates automáticos.

Cada partida de este juego está dividida en dos fases. Durante la primera, el jugador podrá elegir qué unidades quiere desplegar y cómo las quiere sobre el campo de batalla. En la segunda fase, se podrá ver cómo se desarrolla el enfrentamiento contra sus enemigos. Si se gana se obtendrán más monedas que si se pierde, pero siempre se recibirá una compensación, con la que se podrá seguir adquiriendo unidades y subiendo de nivel.

Características que se adaptarán al proyecto

El sistema de combate de este proyecto será parecido al del videojuego ***Teamfight Tactics***.

2.5. Propuesta

La idea de este proyecto es combinar de forma original los tres juegos mencionados, de forma que el videojuego se quede con un sistema de combate parecido al del ***Teamfight Tactics***, una gestión de recursos y cartas parecida a la del ***Slay the Spire*** y una generación de terreno y habitaciones parecida a la del ***Binding of Isaac***.

3. Análisis

En este apartado se van a explicar los requisitos funcionales y no funcionales del proyecto. Además, se analizará los problemas del marco legal que puede tener este proyecto.

3.1. Requisitos Funcionales

Los requisitos funcionales definen funciones del sistema de *software* o sus componentes. Estas funciones se describen como un conjunto de entradas, comportamientos y salidas, determinan funcionalidades que el *software* tiene que satisfacer.

Los requisitos han sido obtenidos a través de la observación del comportamiento de otros videojuegos similares y de la propia experiencia del autor como jugador de este tipo de juegos.

Los requisitos funcionales del proyecto son:

- El usuario debe poder crear y eliminar diferentes perfiles con los que jugar.
- Cada perfil deberá guardar su propia partida y su propio progreso.
- Los perfiles deberán ser persistentes, es decir, aunque se cierre el juego o se creen otros perfiles para jugar, estos no se perderán.
- No debe de haber ningún límite de perfiles creados.
- El usuario podrá poner nombre a los perfiles.
- El usuario podrá cambiar de perfil desde el menú una vez ha seleccionado uno.
- Debe de haber logros.
- Desde el menú principal el usuario podrá acceder a los logros.
- Se deberá poder distinguir qué logros se han conseguido y cuáles no.
- Si el usuario empieza una partida, podrá poder salir al menú principal en cualquier momento y volver a la misma partida más adelante.
- Dentro de la partida, el usuario podrá ver en cualquier momento que cartas de esbirro tiene en su mazo.
- Dentro de la partida, el usuario podrá ver en cualquier momento que cartas de artefacto posee.
- Dentro de la partida, el usuario podrá ver en cualquier momento la cantidad de oro que posee.
- Dentro de la partida, el usuario podrá ver en cualquier momento la cantidad de vida que tiene, así como su vida máxima.
- Dentro de la partida, el usuario tendrá acceso a un minimapa que indique en qué habitación se encuentra y el resto de habitaciones.
- Las habitaciones de cada nivel o piso deberán ser generadas de manera procedural y aleatoria.
- Cada nivel debe de tener por lo menos estas tres habitaciones especiales, “Tienda”, “Tesoro” y “Jefe”. En caso de que el generador no permita que estén estas tres habitaciones, el nivel se re-generará hasta que lo permita.
- Las tres habitaciones especiales mencionadas en el apartado anterior deberán estar situadas en “Callejones sin salida”, es decir, habitaciones que tengan solo una habitación adyacente.

- Dentro de la partida, el usuario podrá mover al personaje sin problema sin atravesar paredes y objetos que no debería poder atravesar.
- En caso de muerte dentro de la partida, el jugador no podrá seguir con la misma, deberá empezar una nueva.
- El juego deberá distinguir entre dos tipos de cartas, Esbirros y Artefactos.

3.2. Requisitos No Funcionales

Los requisitos no funcionales son los que no dependen de la funcionalidad del *software*, en este proyecto los requisitos no funcionales son:

- El juego solo será jugable desde ordenador, no desde cualquier otra plataforma.
- El idioma del videojuego será el castellano.
- El juego deberá ser ejecutado en la resolución 1920x1080.
- El tiempo de juego no deberá depender del procesador del ordenador que lo ejecuta, de esta forma se evita que el personaje se mueva el doble de rápido en un ordenador el doble de potente.

Hay características que son interesantes, como por ejemplo que la resolución del juego no sea fija o poder adaptar el videojuego a diferentes idiomas. Pero estas características se salen de los objetivos de este TFG, por lo que se dejan como posibles ampliaciones en caso de seguir con el proyecto en un futuro.

3.3. Marco Legal

La principal preocupación respecto al marco legal ha sido el respetar la propiedad intelectual, ya que al ser informático y no diseñador, la mayoría del arte ha tenido que ser descargado de Internet.

Para solucionar esto se ha tenido muy en cuenta de donde se ha conseguido el arte del juego, que ha sido principalmente de:

- **Tienda de Unity**: Se han destacado muchos *assets* gratuitos y también se han comprado algunos.
- **Hechos con Paint.NET**: Otros *assets* se han hecho a mano desde cero, a pesar de no artista.
- **Hechos con Inteligencia artificial*:*** Otras imágenes han sido hechas con inteligencia artificial, herramienta que genera imágenes sin licencia que pueden ser usadas sin problema.

4. Diseño

En este apartado se van a explicar las mecánicas del videojuego sin entrar en demasiados detalles en cada apartado, con el fin de que el lector de esta memoria sea capaz de tener una idea general del funcionamiento del videojuego para tener un contexto general cuando se explique cada apartado detalladamente.

4.1. Arquitectura del sistema

4.1.1. Sistema de perfiles.

Este proyecto está pensado para ser usado por un solo usuario en un solo ordenador, sin embargo, es importante que el propio usuario pueda tener diferentes archivos de guardado con diferentes progresos, ya sea porque diferentes personas quieran tener su propio archivo de guardado o porque una misma quiera volver a empezar sin perder su progreso original.

Por este motivo se ha decidido implementar un sistema de perfiles.

El número de perfiles que puede haber no debe estar limitado.

Figura 4.1: *Crear perfil*

Figura 4.2: *Nombre del perfil*

Cada perfil debe guardar su propio progreso individualmente y evidentemente, los perfiles deben tener persistencia. Es decir, cuando sales y entras del juego, estos perfiles estarán guardados, no se perderán.

Al igual que se podrán crear perfiles, también se podrán eliminar, perdiendo así su progreso.

4.1.2. Menú Principal

En el Menú principal se encontrarán cuatro funcionalidades principales con un botón cada una: Empezar partida, Continuar Partida, los Logros y Cambiar de Perfil.

Figura 4.3: *Menú Principal 1*

Al pulsar el botón rojo de empezar partida, simplemente se creará una partida nueva desde cero. Más adelante se entrará en más detalles.

El botón de continuar partida puede estar activado o desactivado como se observa en la Figura 4.4. Esto se debe a que este juego presenta una de las características más importantes del género de videojuegos *Roguelike*, la muerte permanente del personaje.

Figura 4.4: *Menú principal 2*

Cuando el personaje muera, no se podrá seguir con la partida. El botón de "Continuar Partida" saldrá en gris y estará desactivado. Lo mismo ocurrirá en caso de ganar la partida.

Al pulsar el botón de “Logros”, se podrán observar tanto los logros que se han conseguido como los que aún faltan por terminar.

Figura 4.5: *Logros*

Finalmente, en caso de apretar en el botón de “Cambiar Perfil”, se volverá a la pantalla del sistema de perfiles.

4.1.3. Movimiento del Personaje e Interfaz

Al entrar en una nueva partida, lo primero que se verá en el centro de la pantalla será al personaje, que se moverá con las flechas o con las teclas “A”,”W”,”S” y “D”.

Figura 4.6: *Nueva partida*

El personaje no podrá atravesar paredes ni los diferentes elementos del terreno.

Respecto a la interfaz, se verán varios elementos:

- **La vida:** Situada en la esquina superior izquierda de la pantalla, representada por un corazón y dotada de dos valores: Vida actual del personaje y la Vida máxima del personaje. La vida máxima del personaje no podrá nunca bajar de 5 y tampoco podrá superar 99.

Figura 4.7: *Vida*

- **El Oro:** Estará situado también en la esquina superior izquierda de la pantalla, pero debajo del icono de la vida. Solo tendrá un valor que representa la cantidad de oro actual que posee el personaje. El oro servirá para intercambiarlo por objetos en la habitación de la tienda.

Figura 4.8: *Oro*

- **Minimapa:** En la esquina inferior izquierda estará el minimapa. En él, se podrá ver la disposición de las diferentes habitaciones que han sido generadas aleatoriamente.
  También se podrá ver la localización de las diferentes habitaciones especiales.
  Con el botón del Tabulador, se hará el minimapa más grande o lo devolverá al tamaño original, posibilitando ver más habitaciones. En las imágenes se puede observar bien este efecto.
  La habitación en la que se encuentre el jugador estará resaltada de color verde.

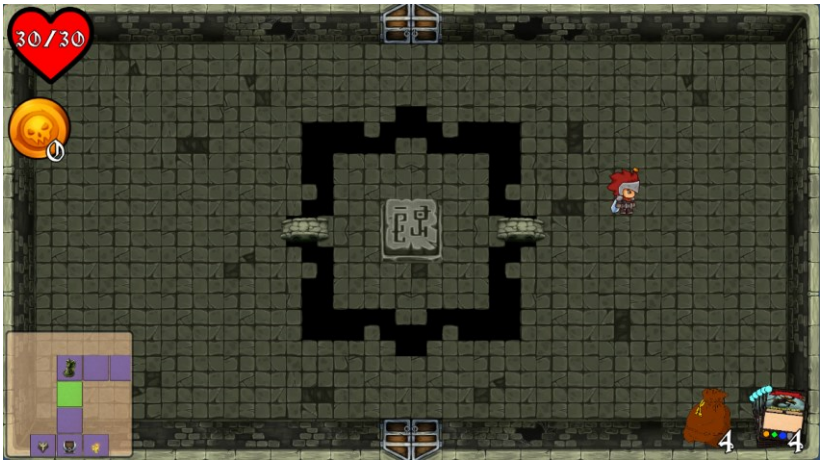

Figura 4.9: *Minimapa 1*

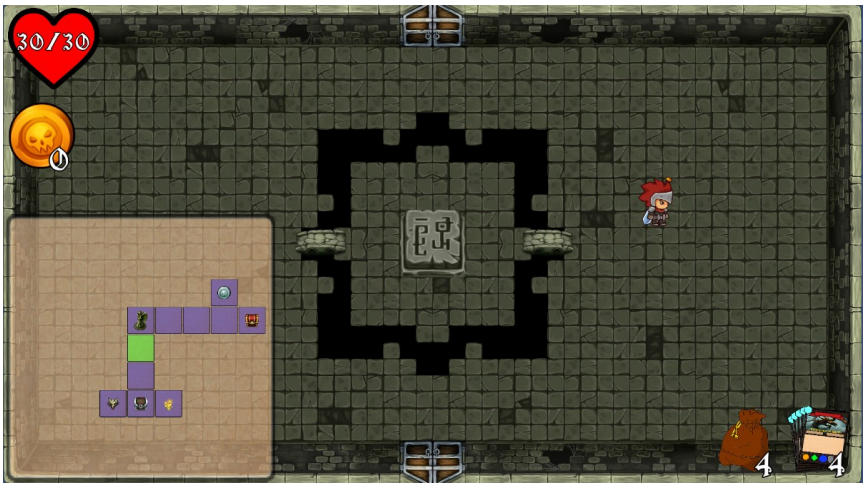

Figura 4.10: *Minimapa 2*

Las habitaciones especiales se marcarán con diferentes iconos.

Figura 4.11: *Habitación en la que empiezas en cada piso*

Figura 4.12: *Habitación con un cofre normal*

Figura 4.13: *Habitación de la vela mágica que sirve para curarse*

Figura 4.14: *Habitación de la Tienda*

Figura 4.15: *Habitación del Tesoro*

Figura 4.16: *Habitación del Jefe*

- **Bolsa de artefactos:** Estará, junto al mazo de esbirros, en la esquina inferior derecha.
  En la interfaz está representado por una bolsa y en caso de apretar sobre ella, se abrirá un menú donde se observarán todos los artefactos que dispone el personaje.
  El número que se ve, representa al número de cartas de artefacto que posee el personaje.

Figura 4.17: *Bolsa de artefactos*

Como se observa en la imagen 4.18, en caso de tener más cartas de artefacto de las que caben en este menú, se podrán ver todas arrastrando hacia abajo con el ratón.

Figura 4.18: *Cartas de artefactos del personaje*

- **Mazo de esbirros:** Se encontrará en la esquina inferior izquierda de la pantalla, a la derecha de la bolsa de artefactos.
  El número que se verá, representará el número de cartas de esbirro que posee el personaje.

Figura 4.19: *Mazo de esbirros*

Al igual que la bolsa de artefactos, funcionará como botón y al apretarlo, se observarán todas las cartas de esbirro que posee el jugador.

En caso de tener más cartas de esbirro de las que caben en el menú, se podrán ver el resto de cartas arrastrando el ratón, al igual que si fuese una página web.

Figura 4.20: *Cartas de esbirro del jugador*

4.1.4. Cartas

En este proyecto se encontrarán dos tipos principales de cartas: cartas de artefactos y cartas de esbirros.

Las cartas de artefacto otorgarán efectos pasivos, la mayoría de ellos durante el combate. En la imagen 4.21 se pueden observar algunos ejemplos.

Figura 4.21: *Cartas de artefacto*

Las cartas de esbirro se utilizarán en combate. Estas cartas tendrán una serie de estadísticas que vendrán dadas por su Raza (Elfo, Enano, Humano…) y su Clase (Druida, Mago, Brujo…).

Estas estadísticas serán: El maná, la fuerza, la destreza, la magia, la armadura y la vida.

Los Esbirros serán los que se usarán en el sistema de combate. Aunque no esté implementado aún, ya que ese módulo corresponde al trabajo final de grado de mi compañero Pau y él no lo tendrá acabado para esta entrega, sí que se explicará para que el lector pueda tener una idea general del funcionamiento del videojuego una vez esté terminado.

En la imagen 4.22 se pueden ver ejemplos de este tipo de cartas.

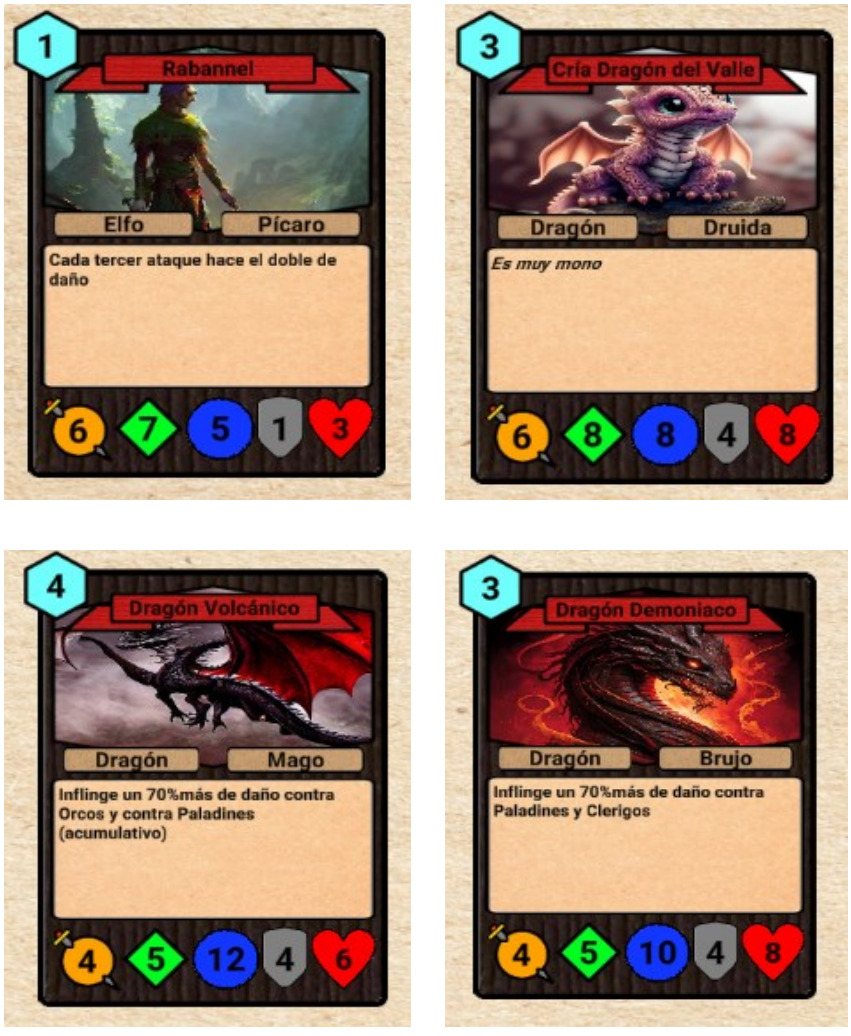

Figura 4.22: *Cartas de esbirro*

4.1.5. Recompensas

Cuando el jugador abra un cofre o derrote enemigos, recibirá recompensas.

Estas recompensas se otorgarán mediante objetos *pickup* que el jugador podrá recoger cuando pase por encima.

Habrá varios tipos de *pickup* que puede encontrar el jugador.

Pickup de Esbirro

El *pickup* de esbirro tendrá forma de carta. Esta carta rotará sobre sí misma, mostrando un dorso rojo cuando se vea por detrás. En la imagen 4.23 se puede ver como se verá por delante y por detrás.

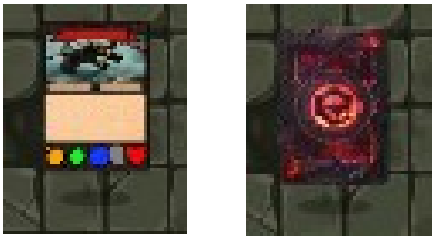

Figura 4.23: *Pickup de Esbirro*

Además de rotar sobre sí misma, también subirá y bajará lentamente, dando la sensación de que está flotando.

Cuando el jugador pase por encima de este *pickup*, aparecerá la pantalla que se puede observar en la figura 4.24.

Figura 4.24: *Elegir o no una carta de esbirro*

En esta pantalla el jugador podrá elegir si quiere añadir alguna de las cartas que se le muestran a su mazo de esbirros. También, podrá decidir no añadir ninguna.

Si hay más cartas de las que caben en la pantalla, el jugador podrá verlas todas arrastrando hacia la derecha o izquierda el ratón.

En el momento en el que el jugador haga clic sobre alguna carta, esta se añadirá a su mazo de esbirros.

***Pickup* de Artefacto**

El *pickup* de artefacto tendrá forma de carta. Esta carta rotará sobre sí misma, mostrando un dorso morado cuando se vea por detrás. En la imagen 4.25 se puede observar cómo se verá por delante y por detrás.

Figura 4.25: *Pickup de artefacto*

Además de rotar sobre sí misma, también subirá y bajará lentamente, dando la sensación de que está flotando.

Cuando el jugador pase por encima de este pickup, aparecerá la pantalla que se puede ver en la imagen 4.26.

Figura 4.26: *Elegir o no una carta de artefactos*

En esta pantalla el jugador podrá elegir si quiere añadir alguna de las cartas que se le muestran a su bolsa de artefactos o podrá decidir no añadir ninguna.

En caso de que haya más cartas de las que caben en la pantalla, el jugador podrá verlas todas arrastrando hacia la derecha o izquierda el ratón.

En el momento en el que el jugador haga clic sobre alguna carta, esta se añadirá a su bolsa de artefactos.

***Pickups* de Oro**

Habrá varios tipos de *pickups* de oro, pero todos funcionarán de la misma manera. Cuando el jugador pase por encima, desaparecerán usando un efecto de partículas y se añadirá una cantidad de oro al Inventario.

La cantidad de partículas que saldrán dependerá del tamaño de la pila de oro.

A continuación, se muestran imágenes de los seis diferentes tipos de *pickups* de oro.

Figura 4.27: *Pila de oro pequeña*

Figura 4.28: *Pila de oro mediana*

Figura 4.29: *Pila de oro grande*

Figura 4.30: *Pila de oro pequeña con joyas*

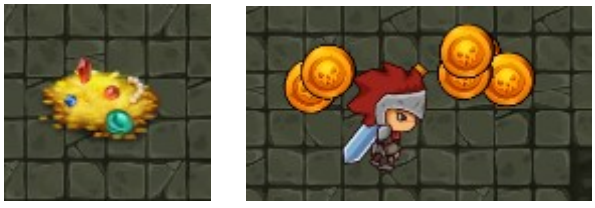

Figura 4.31: *Pila de oro mediana con joyas*

Figura 4.32: *Pila de oro grande con joyas*

El valor de las diferentes pilas de oro será de 5, 10, 20, 35, 50 y 100 respectivamente.

4.1.6. Pisos

Los diferentes pisos serán generados de manera aleatoria y procedural, esto significa que aunque llegue al mismo piso en partidas distintas, cada piso será diferente.

En una misma partida, cada vez que se supere un piso, el siguiente será más grande, con más habitaciones y la dificultad de estas aumentará.

En las siguientes imágenes se pueden ver varias generaciones de cada piso para que se aprecien las diferencias.

Figura 4.33: *Piso 1*

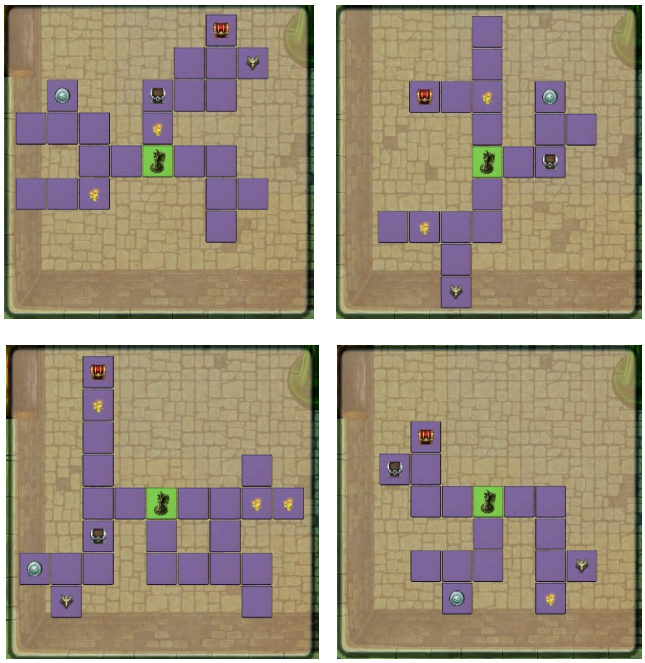

Figura 4.34: *Piso 2*

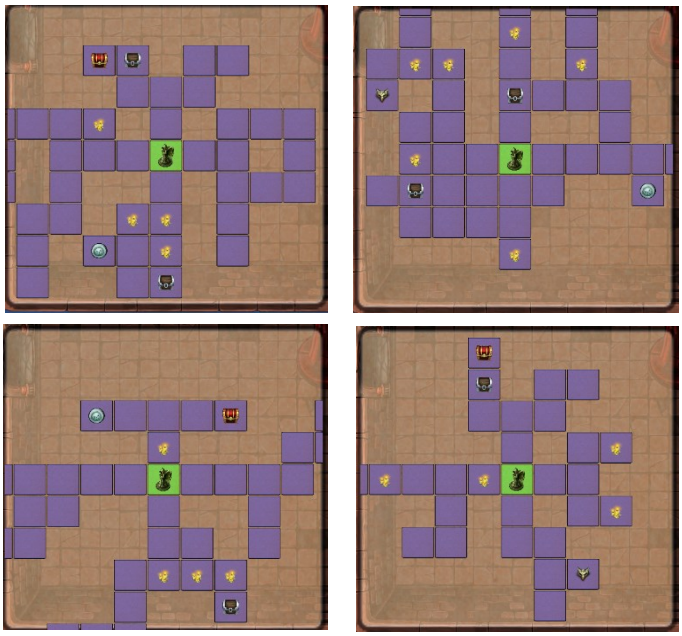

Figura 4.35: *Piso 3*

4.1.7. Habitaciones

Los diferentes pisos estarán compuestos por diferentes habitaciones. Estas habitaciones podrán ser de dos tipos: habitaciones normales o habitaciones especiales.

Las habitaciones normales no tendrán ningún icono en el minimapa. En ellas habrá enemigos. Se cerrarán las puertas cuando se entre en ellas y no se abrirán hasta que se derrote a los enemigos.

El sistema de combate lo realizará mi compañero Pau Vidal Ramón en su TFG. Como él no lo tiene hecho para esta entrega, se ha dejado una placa de presión como *placeholder* que al pulsarla se abren las puertas y te da la recompensa de haber derrotado a los enemigos.

Habrá diferentes tipos de habitaciones especiales:

Habitación de inicio

Estará señalada en el minimapa con el ícono de una estatua como se ve en la figura 4.36.

Figura 4.36: *Icono de la habitación de inicio*

Será la sala en la que empieza el personaje al inicio de cada piso. Se caracterizará por tener varias estatuas de piedra.

Las puertas estarán abiertas para que directamente se pueda ir a explorar el resto del piso.

Figura 4.37: *Habitación de inicio 1*

Figura 4.38: *Habitación de inicio 2*

Habitación de la Vela Mágica

Estará marcada en el minimapa con el icono de una vela.

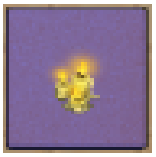

Figura 4.39: *Icono de la habitación de la Vela Mágica*

Puede haber más de una en cada piso. No será una sala hostil, por lo que las puertas estarán siempre abiertas.

Se caracterizará por tener una vela mágica gigante en el centro de la sala. Esta vela estará apagada y si el personaje se acerca, la vela se encenderá y curará al personaje un X % de la vida.

En las imágenes 4.40 y 4.41 se puede ver una habitación de la Vela Mágica con la vela encendida y apagada.

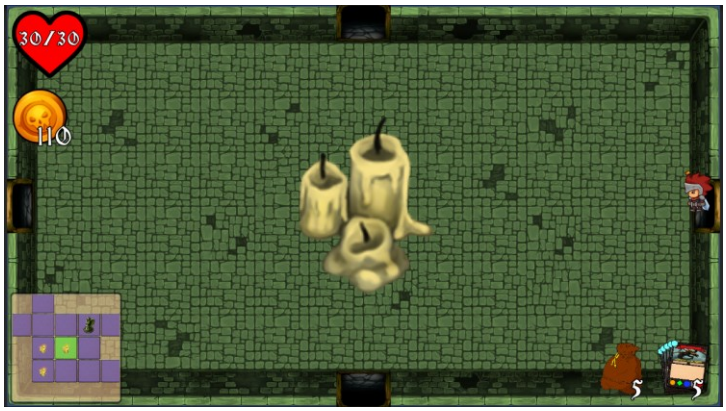

Figura 4.40: *Habitación de la Vela Mágica 1*

Figura 4.41: *Habitación de la Vela Mágica 2*

Habitación del Cofre

Estará marcada en el minimapa con un icono de un cofre marrón.

Figura 4.42: *Icono de la habitación del cofre*

Podrá haber más de una en cada piso. No será una sala hostil, por lo que las puertas estarán siempre abiertas.

Se caracterizará por tener un cofre de madera en el centro de la sala. El cofre estará cerrado y si el jugador se acerca, le dará recompensas, como oro, cartas y artefactos que podrá añadir a su mazo.

En las figuras 4.43 y 4.44 se puede ver una habitación del cofre con el cofre cerrado y abierto con las recompensas que ha soltado.

Figura 4.43: *Habitación del cofre 1*

Figura 4.44: *Habitación del cofre 2*

Habitación de la tienda

Estará marcada en el minimapa con un icono de una moneda plateada.

Figura 4.45: *Icono de la habitación de la tienda*

Es única por piso y siempre estará generada en un "callejón sin salida", es decir, solo tendrá una habitación adyacente.

No será una sala hostil, por lo que las puertas estarán abiertas siempre.

En esta tienda se podrá intercambiar el oro que tenga el jugador por diferentes cartas.

El precio de cada objeto comprable se podrá ver debajo del mismo.

Para comprar los objetos simplemente habrá que caminar por encima de ellos.

En las figuras 4.46, 4.47 y 4.48 se pueden observar diferentes tiendas.

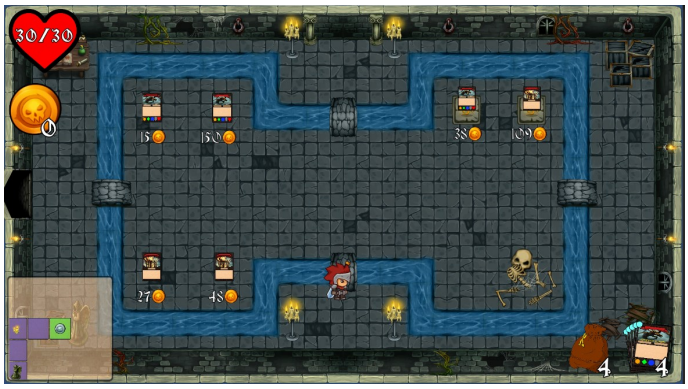

Figura 4.46: *Habitación de una tienda 1*

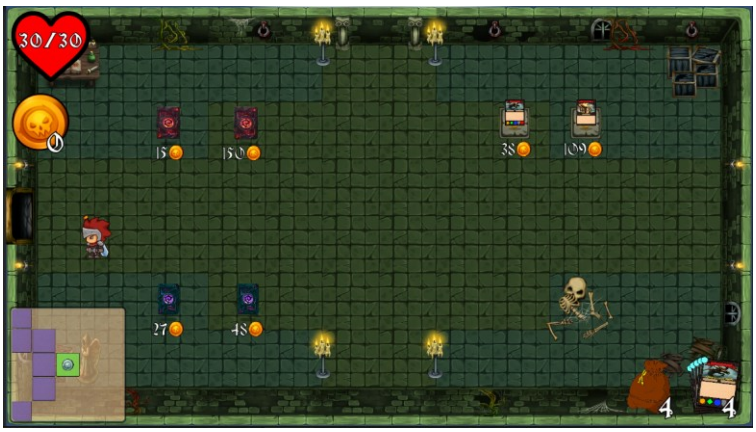

Figura 4.47: *Habitación de una tienda 2*

Figura 4.48: *Habitación de una tienda 3*

Habitación del Tesoro

Estará marcada en el minimapa con un icono de un cofre dorado y rojo.

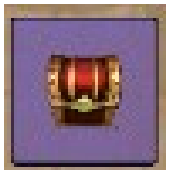

Figura 4.49: *Icono de la habitación del tesoro*

Es única por piso y siempre estará generada en un “callejón sin salida”, es decir, solo tendrá una habitación adyacente.

No será una sala hostil, por lo que las puertas estarán abiertas siempre.

Se caracterizará por tener un cofre dorado y rojo gigante en el centro de la sala.

El cofre estará cerrado y si el jugador se acerca, le dará recompensas al jugador, como oro, cartas y artefactos que podrá añadir a su mazo. Estas recompensas tenderán a ser mejores que las de los cofres normales.

En las figuras 4.50 y 4.51 se puede ver una habitación del tesoro con el cofre cerrado y abierto con las recompensas que ha soltado.

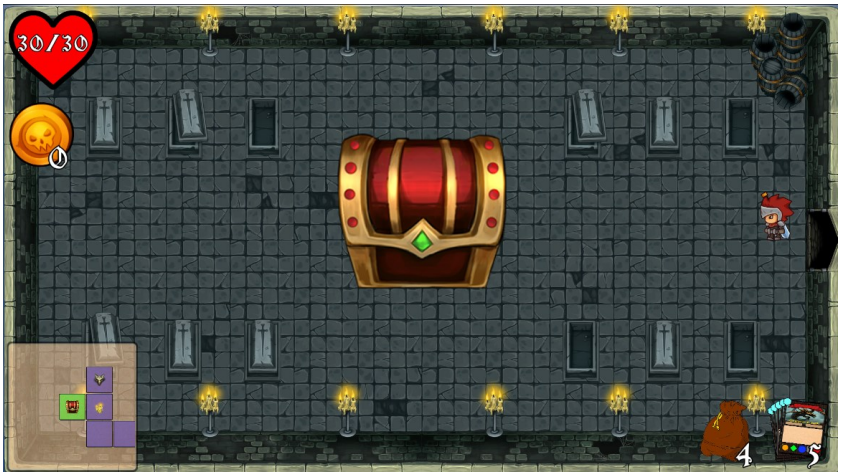

Figura 4.50: *Habitación del tesoro 1*

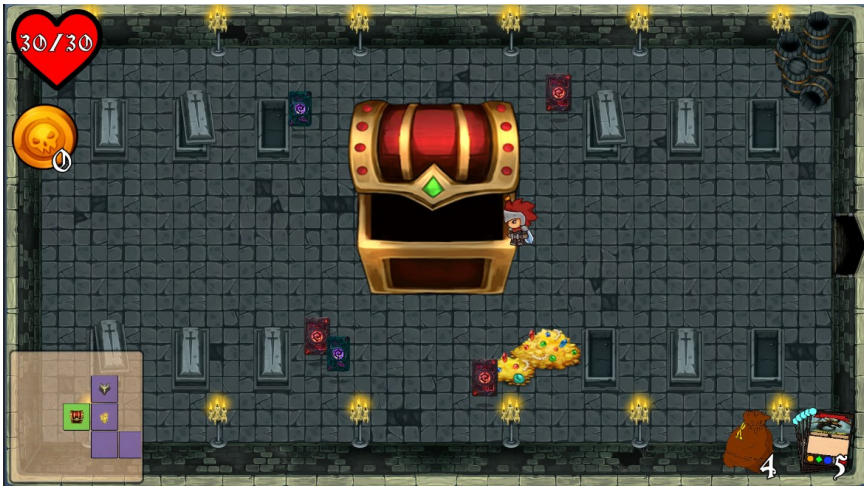

Figura 4.51: *Habitación del tesoro 2*

Habitación del Jefe

Estará marcada en el minimapa con el icono de la figura 4.52.

Figura 4.52: *Icono de la habitación del Jefe*

La habitación del Jefe actuará como una habitación de enemigos normal, pero solo habrá un enemigo que será más grande y fuerte de lo habitual.

Al ser una habitación hostil, las puertas se cerrarán cuando el jugador entre y no se abrirán hasta que derrote al Jefe del piso.

Cuando se derrote al Jefe, aparte de abrirse las puertas y soltar recompensas, también aparecerán unas escaleras que, al entrar en ellas, llevarán al siguiente piso.

En caso de entrar por estas escaleras, no se podrá retroceder.

Si el jugador se encuentra en el último piso y derrota al Jefe, aparecerá un cofre especial que al abrirlo surgirá un letrero de “Victoria!”, y pasados unos segundos, el usuario volverá al menú principal.

Al igual que en las habitaciones normales, el sistema de combate de esta habitación no está implementado, ya que pertenece al trabajo final de grado de mi compañero Pau Vidal Ramón, que lo presentará más adelante.

En su defecto, se ha utilizado una placa de presión como *placeholder* que cuando el jugador la pise, simulará que se ha derrotado al jefe y se abrirán las puertas, soltando recompensas y aparecerán las escaleras hacia el siguiente piso.

En la figura 4.53 se pueden observar distintos tipos de escaleras para bajar de piso, y en la figura 4.54, el cofre especial con el que se acaba la partida.

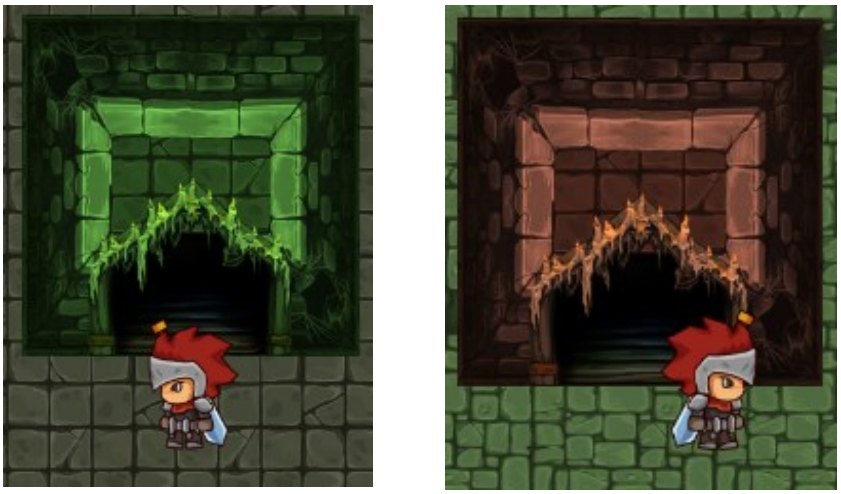

Figura 4.53: *Escaleras para bajar de piso*

Figura 4.54: *Cofre especial, fin de partida*

4.1.8. Sistema de Combate

Aunque el sistema de combate no esté implementado, ya que pertenece al trabajo final de grado de mi compañero Pau Vidal Ramón, es importante tener una idea general de cómo va a funcionar para poder entender este proyecto en su totalidad.

Cuando se entre en una sala hostil, se iniciará este sistema de combate.

Se podrán ver los enemigos y sus posiciones de la sala en la que se encuentra el personaje.

El jugador robará una cantidad X de cartas de esbirros de su mazo de cartas de esbirro.

Se tendrá una cantidad máxima de maná para usar. Usar una carta costará la cantidad de maná que indique la propia carta.

Las cartas de esbirro se usarán para crear al esbirro en la habitación. Una vez se creen los esbirros se le podrá dar al botón de “Luchar”.

En ese momento, los enemigos y los esbirros creados empezarán a luchar entre ellos por sí mismos, sin necesidad de que el jugador interactúe.

La forma en la que actúan los esbirros dependerá de sus estadísticas, así como el daño que hacen, la vida que tienen, etc.

El jugador no podrá volver a interactuar hasta que se mueran o todos los esbirros del usuario o todos los enemigos.

Si se mueren todos los enemigos, se saldrá del sistema de combate y las puertas de la sala se abrirán. También, se recibirán recompensas.

En caso de que mueran todos los esbirros del personaje, el jugador recibirá una cantidad de daño determinada por el número de enemigos que queden con vida. El jugador volverá a

robar cartas de su mazo y volverá a tener todo el maná. Pudiendo invocar a nuevos esbirros para que terminen de rematar a los enemigos que queden con vida.

Este ciclo continuará hasta qué o mueran todos los enemigos o el personaje se quede sin vida, en cuyo caso se acabará la partida.

En caso de que se acaben las cartas del mazo de esbirros, este se volverá a mezclar y se seguirán robando cartas.

4.1.9. Acabar la partida

Habrá dos formas de acabar la partida, Morir o Ganar.

En caso de que la vida del personaje sea igual o inferior a cero, el personaje morirá y saldrá un letrero de “Has Muerto”. Pasados unos segundos, enviará al jugador al menú principal que no podrá continuar la partida, solo empezar una nueva.

Figura 4.55: *Muerte del personaje*

Si el jugador superá el último piso, aparecerá un cofre especial que al abrirlo saldrá un letrero de “Victoria!”. Pasados unos segundos, lo enviará al menú principal y no podrá continuar con la partida, solo empezar una nueva.

Figura 4.56: *Victoria del jugador*

4.1.10. Menú de Pausa

En cualquier momento de la partida, el usuario podrá acceder al Menú de pausa pulsando la tecla “Escape”.

Esté menú, solo tendrá dos botones grandes: “Continuar Jugando” y “Salir al Menú”.

Figura 4.57: *Menú de pausa abierto*

En caso de apretar el botón de “Continuar Jugando”, el menú de pausa se cerrará.

Si se pulsa el botón de “Salir al Menú”, el juego llevará al menú principal con el botón de “Continuar Partida” habilitado.

4.1.11. Debugger

Para poder hacer pruebas y ver que todo funciona bien, se implementará un modo administrador o un *Debugger*.

El *Debugger* actuará como si fuese una consola de comandos o una terminal, el usuario podrá insertar comandos y estos se activarán.

Habrá una línea de texto que informará de si el comando se ha ejecutado correctamente o ha habido algún error por parte del usuario, como haber escrito mal el comando.

A continuación, se listarán todos los posibles comandos, junto con imágenes de todas las posibles respuestas de esta consola de comandos:

Lo que se encuentre entre los símbolos “<” y “>” indica que se espera una variable, por ejemplo, <número> indica que el comando espera un número.

- **ADDRANDOMCARD <número>**: Añade un número determinado de cartas aleatorias de esbirro al mazo de Esbirros.

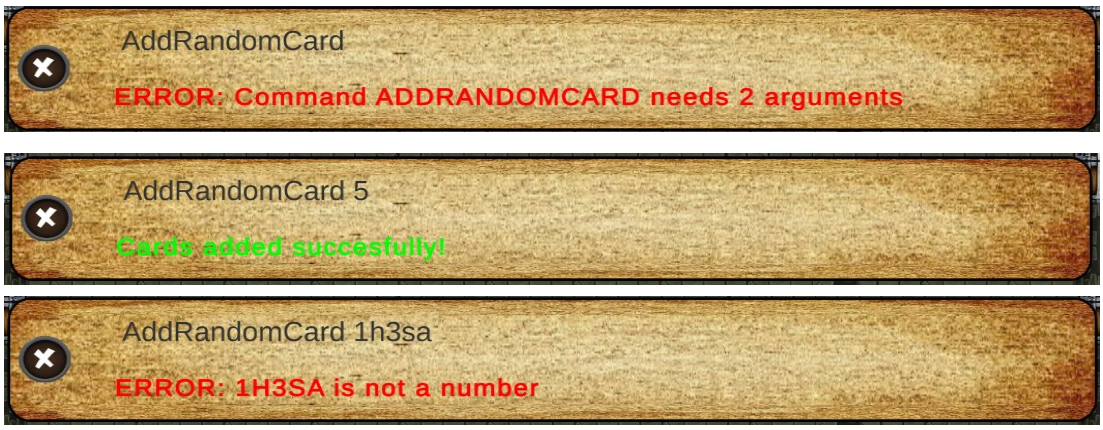

Figura 4.58: *Comando AddRandomCard*

- **ADDRANDOMARTIFACT <número>**: Añade un número determinado de artefactos aleatorios a la bolsa de Artefactos.

Figura 4.59: *Comando AddRandomArtifact*

- **ADDGOLD <número>**: Añade el número indicado de oro al inventario, puede ser negativo para restar oro.

Figura 4.60: *Comando AddGold*

- **SPAWN <nombre>**: Crea encima del personaje el objeto determinado por la *string* (el nombre del objeto a crear).

Figura 4.61: *Comando Spawn*

Estos serán los objetos a *spawnear*:

  - **ArtifactPickup**: *Pickup* con Artefactos a elegir para añadir a la bolsa de artefactos.
  - **CardPickup**: *Pickup* con Esbirros a elegir para añadir al mazo de esbirros.
  - **GSmallPickup**: *Pickup* de oro para recoger, con un valor total de 5 de oro.
  - **GMidPickup**: *Pickup* de oro para recoger, con un valor total de 10 de oro.
  - **GBigPickup**: *Pickup* de oro para recoger, con un valor total de 20 de oro.
  - **GSmallJewelPickup**: *Pickup* de oro para recoger, con un valor total de 35de oro.
  - **GMidJewelPickup**: *Pickup* de oro para recoger, con un valor total de 50de oro.
  - **GBigJewelPickup**: *Pickup* de oro para recoger, con un valor total de 100de oro.
  - **ArtifactSellable**: Carta de artefacto que puedes ver para elegir si comprarla o no.
  - **CardSellable**: Carta de Esbirro que puedes ver para elegir si comprarla o no.
  - **SellChooseArtifactPickUp**: ArtifactPickup pero con precio pensado para que se encuentre en la tienda.
  - **SellChooseCardPickUp**: CardPickup pero con precio pensado para que se encuentre en la tienda.

- **OPENALLDOORS**: Abre todas las puertas del piso en el que se encuentra el personaje.

Figura 4. 62: *Comando OpenAllDoors*

- **CLOSEALLDOORS**: Abre todas las puertas del piso en el que se encuentra el personaje.

Figura 4.63: *Comando CloseAllDoors*

- **CLEARMAP**: Borra el minimapa dejándolo vacío. Está pensado para comprobar que funcionan bien las actualizaciones del minimapa cuando se cambia de piso.

Figura 4.64: *Comando ClearMap*

- **HEAL <número>**: Cura el número determinado de vida al jugador.

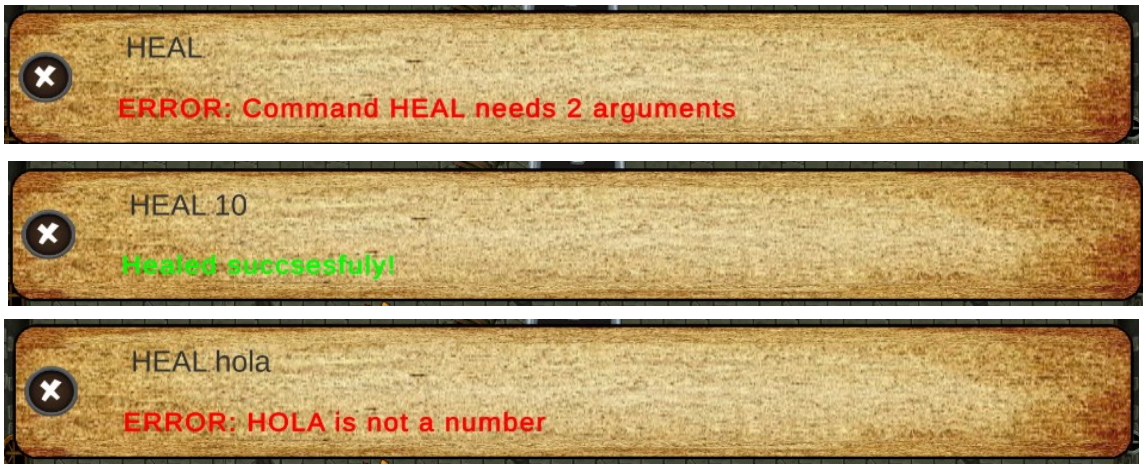

Figura 4.65: *Comando Heal*

- **DAMAGE <número>**: te hace X puntos de daño.

Figura 4.66: *Comando Damage*

- **ADDMAXHEALTH <número>:** Añade X puntos de vida máxima, puede ser negativo. La vida máxima está limitada a 99 y no puede ser menor de 5.

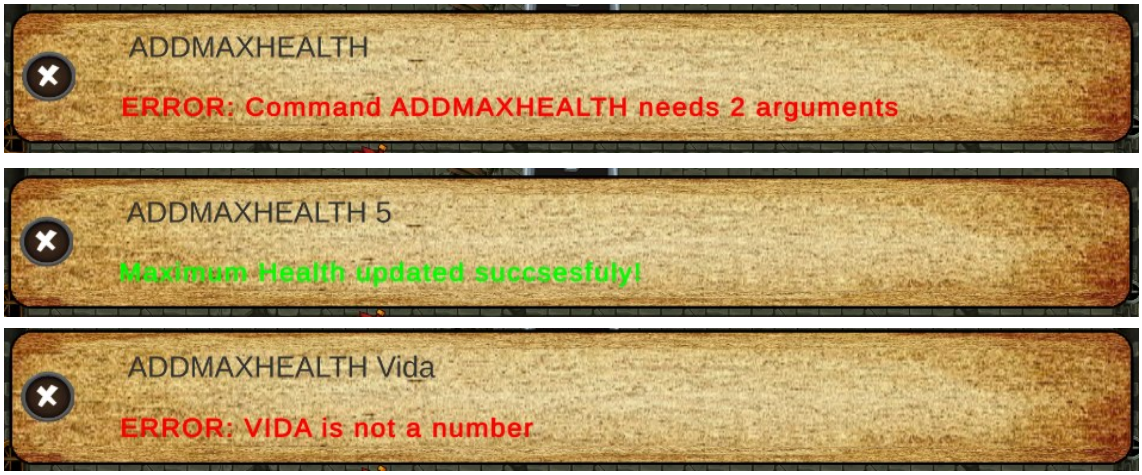

Figura 4.67: *Comando AddMaxHealth*

4.2. Tecnología utilizada

En este apartado se mostrará las diferentes herramientas y tecnologías que han sido empleadas en el proyecto y la justificación del porqué de esas herramientas y no otras.

4.2.1. Motor de juegos Unity

Se ha elegido Unity y no otro como Unreal Engine por varios motivos:

- En primer lugar, la idea de hacer este proyecto se formó gracias a la asignatura de *Desarrollo de Videojuegos 3D*, en la cual me di cuenta de lo mucho que me gustaba hacer videojuegos y aprendí a utilizar este motor.
- En segundo lugar, en *Apolo Kid*s, el lugar donde he estado haciendo prácticas, también utilizamos este motor de juegos.
- En último lugar, Unity es un motor que es muy potente y a la vez, es relativamente simple. Por lo que al tener ya bastante experiencia con este motor sabía que le podía sacar bastante partido.

4.2.2.Lenguaje de programación C#

Respecto al lenguaje de programación, lo normal en videojuegos es utilizar lenguajes que faciliten la Programación Orientada a Objetos. Los más utilizados son C/C++, Java o C# principalmente.

Los dos grandes motores de juegos son Unreal Engine y Unity. Con Unreal Engine se suele programar en C++, ya que da unas ventajas de rendimiento y velocidad considerables, sin embargo, este proyecto no es tan ambicioso ni necesita semejante inversión de tiempo y aprendizaje para que sea rentable utilizar Unreal con C++.

Con Unity se suele programar en C#, porque es su lenguaje predeterminado y tiene muchas facilidades e integraciones ya hechas para este lenguaje.

Teniendo en cuenta que el trabajo se va a realizar en Unity, la conclusión lógica es llevar a cabo este proyecto en C#.

4.2.3.Editor de Código: Visual Studio 2019

Al igual que Unity está preparado para trabajar con C#, también tiene integraciones para Visual Studio, inclinando la balanza a utilizar este editor.

Pero Visual Studio también tiene muchas otras ventajas como la facilidad de creación de código, el resaltado de código con Unity, control de versiones, depuración simple y fácil, etc.

Por todo esto, la opción lógica es utilizar Visual Studio.

4.2.4.Editor de Imágenes 2D: Paint.NET

Aunque muchos de los *sprites* que han sido utilizados o bien han sido comprados o bien descargados de Internet con licencias de libre uso, muchas veces se ha necesitado o bien modificar algunas imágenes existentes o bien crear otras desde cero.

Se ha elegido esta herramienta, ya que ni es demasiado simple como podría ser el *Paint* de Windows, que es muy simple pero muy poco potente. Ni es demasiado compleja como el Photoshop, que tiene mucho potencial, pero es muy compleja.

Al estar en ese punto intermedio, se ha seleccionado esa herramienta ya que proporciona lo necesario para poder llevar a cabo este proyecto.

4.2.5.Generador de Imágenes por IA: Midjourney

Para la creación de imágenes complejas como los fondos o Imágenes de las cartas, se ha optado por generarlas mediante inteligencia artificial, en concreto utilizando la inteligencia artificial Midjourney.

Finalmente, se ha elegido esta inteligencia artificial gracias a un proceso de prueba y error, en el que se generaron varias imágenes con las diferentes Inteligencias artificiales que generan imágenes actuales y con la que mejores resultados se obtuvieron fue con Midjourney.

Los factores que llevaron a la elección de utilizar Inteligencia Artificial para generar imágenes fueron principalmente:

- Se han obtenido muy buenos resultados que encajan perfectamente con lo que se buscaba.
- Las imágenes no tienen licencia, por lo que se pueden utilizar sin problema.
- La facilidad para generar imágenes.

5. Desarrollo de la solución propuesta

Este proyecto se ha llevado a cabo mediante la metodología ágil *Feature Driven Development* (FDD).

En este apartado se va a explicar cómo se han desarrollado las diferentes funcionalidades. Como algunas funcionalidades se explican mejor junto a otras, se van a agrupar en módulos.

5.1. Sistema de Perfiles

Archivos .Json

El objetivo de implementar un sistema de perfiles es conseguir tener persistencia en el juego, para poder así guardar el progreso.

Para guardar el progreso, se crean una serie de archivos .Json en los que se guarda todo lo necesario.

En primer lugar, se tiene un archivo Perfiles.Json que no se puede eliminar nunca, ya que es el encargado de almacenar la cantidad de perfiles que hay creados y sus nombres. Este Perfiles.Json simplemente almacena una lista de *strings* o nombres.

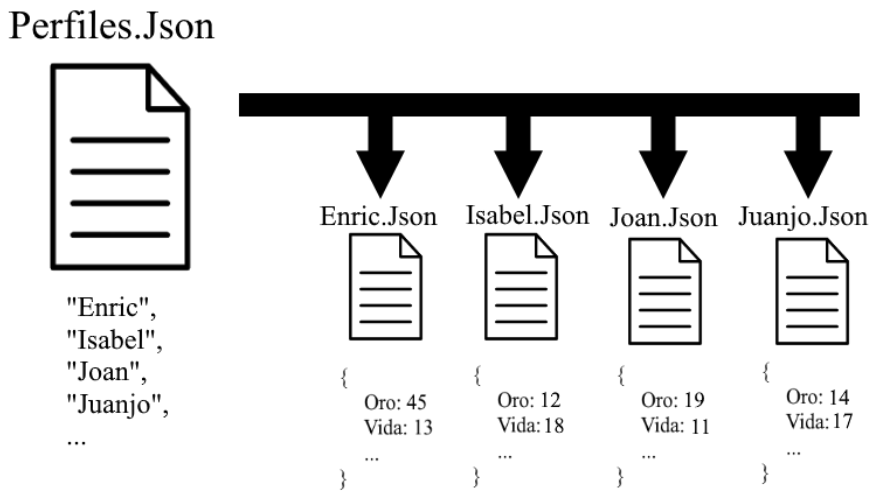

Figura 5.1: *Perfiles.Json*

Siempre que cree un perfil, se creará otro archivo .Json con el nombre del perfil. Este archivo .Json se encargará de guardar todos los elementos necesarios para la persistencia en cada perfil. En concreto, este archivo Json guarda: el Oro de la partida, la vida actual del Personaje, la vida máxima del personaje, una lista con los IDs (números) de las cartas de esbirro del personaje, una lista con los IDs (números) de las cartas de artefacto del personaje, un *bool*

(true/false) de si hay una partida en curso o no, el nombre del Piso en el que se encontraba el personaje, una lista de booleanos que representa los logros conseguidos o no y una lista de una estructura que contiene toda la información de las habitaciones del piso.

Gracias a esta última lista, se puede recrear el mismo piso en el que se encontraba el personaje.

En caso de eliminar un perfil, se eliminará de la lista del .Json principal y se eliminará el archivo .Json asociado a ese perfil.

Cuando se cree un perfil se comprobará que no exista ya ninguno con el mismo nombre.

Menú de perfiles

A nivel de maquetación de este menú, se pueden observar varios componentes.

Figura 5.2: *Menú de perfiles*

En primer lugar, se tiene el fondo, que se mantiene a lo largo de todo el sistema de perfiles y el menú principal. Este fondo ha sido generado mediante Inteligencia Artificial, concretamente utilizando Midjourney.

En la parte superior se encuentra el título. La tipografía del título se mantendrá a lo largo de todo el juego, ya que le da un toque original y fantástico al videojuego. Esta tipografía no es original, se descargó por Internet y no tiene licencia, así que puede ser usada sin problema.

En la parte inferior se encuentra un botón de "Crear Perfil", al pulsar este botón llevará al usuario a un *popup* de crear perfil que se verá más adelante en este mismo apartado.

En el centro está el apartado en el que se observan los diferentes perfiles. Este apartado está preparado para poder manejar todo el número de perfiles que haga falta.

Para empezar, hay un componente invisible con forma de rectángulo que tiene un componente "Máscara". Con este componente, si hay más perfiles de los que caben en el centro, estos perfiles no se ven. En las imágenes 5.3 y 5.4 se puede observar la diferencia entre tener el componente activado o no.

Figura 5.3: *Componente Máscara activado*

Figura 5.4: *Componente Máscara desactivado*

Además, también se le añade un componente "*Scroll Rect*" que sirve para poder arrastrar el ratón hacia arriba y hacia abajo para poder acceder a todos los perfiles.

***Popup* de crear perfiles**

En el momento en el que se apriete el botón de "Crear Perfil", aparecerá el *popup* que se puede observar en la Figura 5.5.

Figura 5.5: *Popup de crear perfiles*

En este *popup* se puede observar un botón en la parte izquierda con forma de X. En caso de apretarlo, se cerrará el *popup* sin crear ningún perfil nuevo.

En la parte central se tiene un objeto rectangular con un componente "*Input Field*" que permite introducir texto. Destacar que todas las texturas rugosas y con colores que se observan en estos menús, han sido hechas a mano con Paint.Net.

A la parte derecha del *popup* existe un botón de "Crear", cuando sea pulsado creará un nuevo archivo .Json con el nombre que haya en el "*Input Field*", se instanciará un *prefab* de perfil en el centro junto a los demás perfiles (en el siguiente apartado se explicará que es un *prefab* de perfil) y se añadirá el nombre que haya en el "*Input Field*" a la lista de Perfiles.Json.

En caso de que el nombre este repetido, no lo dejará crear y saldrá el mensaje de error sobre el "*Input Field*" como se ve en la Figura 5.6.

Figura 5.6: *Nombre de perfil repetido*

***Prefab* de Perfil**

Al entrar en el menú de perfiles, se creará un *prefab* de perfil por cada perfil que haya guardado en Perfiles.json

Este *prefab* está compuesto por un fondo rojo y dos botones.

El botón de la derecha tiene una X en el centro y cuando se le aprieta, borra el perfil en cuestión.

El botón de la izquierda es mucho más alargado y como texto tendrá el nombre del perfil. Cuando es apretado, cambia de escena al menú principal cargando todos los datos del .Json del perfil seleccionado.

Figura 5.7: *Prefab de Perfil*

5.2. Menú Principal

Cuando se carga el menú principal, lo primero que se hace es comprobar si hay alguna partida en curso. Esto se comprueba gracias a una variable booleana *(true/false)* que se ha extraído del .Json del perfil cargado.

Si hay una partida en curso, el botón de "Continuar Partida" estará activado como se observa en la figura 5.8.

Figura 5.8: *Botón "Continuar partida" activado*

En caso de que la variable booleana sea "*false*", el botón de "Continuar Partida" estará desactivado. Al estar desactivado simplemente se cambia el objeto de este botón por otro similar de color gris, el cual no se puede pulsar. Se puede observar cómo se vería en la figura 5.9.

Figura 5.9: *Botón "Continuar partida" desactivado*

Hay un total de cuatro botones. A continuación, se explicará qué ocurre cuando se pulsa en cada uno de ellos.

Empezar Partida

Cuando se aprieta el botón de empezar partida, lo primero que se hace internamente es eliminar todos los datos que pueda haber guardados de alguna partida anterior y *resetear* esos valores a los valores por defecto.

Al oro se le instancia un valor de 0, a la vida un valor de 30, a la vida máxima un valor de 30, a la lista de las cartas de esbirro se añaden los esbirros con ID 1, 2, 3 y 4, a la lista de cartas de artefactos se añaden los artefactos con ID 1, 2, 3 y 4, se instancia el booleano de que hay una partida guardada a “true” y se vacía la lista con la información de todas las habitaciones que hay y sus posiciones.

Una vez se tienen todos los valores instanciados, se añade la escena que genera el terreno proceduralmente y la escena del juego. Una vez se cargan estas escenas, se cierra la escena del menú principal.

Mientras ocurre todo el cambio de escenas, se puede observar una pantalla de carga como la de la figura 5.10.

Figura 5.10: *Pantalla de carga*

Continuar Partida

Si se puede apretar el botón de continuar partida es porque había datos guardados de una partida en curso.

En el momento en que se aprieta este botón, se añade la escena del juego y además se llama a la escena que genera el terreno proceduralmente, pero esta vez se le informa a través de un booleano que no tiene que generar un nuevo terreno desde cero, sino generar el terreno que está guardado en la lista del archivo .Json.

Cuando estas escenas están cargadas, se cierra la escena del menú principal.

Logros

Al apretar este botón, se abre el *popup* de logros. Este *popup* se puede observar en la figura 5.11.

Figura 5.11: *Popup de logros*

A nivel de maquetación, este *popup* es muy parecido al sistema de elegir perfil.

En primer lugar, se encuentra el título del *popup* en la parte superior, que simplemente es un texto estático con la tipografía cambiada.

En la parte superior izquierda se encuentra un botón marcado con una X, que en caso de ser pulsado se cierra el *popup* de logros.

En el centro se encuentra la lista con todos los posibles logros, que en total hay tres. Estos logros o bien aparecen en verde si han sido superados, o en gris y negro si aún no se han conseguido.

Para tener controlados los posibles logros y poder añadir nuevos, en el menú existe un objeto vacío que almacena una lista de *strings* o frases que representa el número de logros. Actualmente, esta lista solo contiene las *strings* "Supera el primer piso", "Supera el segundo piso" y "Supera el tercer piso".

Cuando se abre este *popup*, se recorre una Lista de booleanos de las variables del archivo .Json, que representa si estos logros han sido superados. Si la lista es {"*true*", "*false*", "*false*"}, solo se activa en verde el primer elemento de la lista de *strings*.

En caso de añadir nuevos elementos a la lista de *strings* para implementar nuevos logros, el juego ya está preparado para que cuando se intente comprobar si el logro está obtenido en la lista de booleanos, esta misma actualice el número de elementos para quedar con los mismos elementos de la lista de *strings*, evidentemente estos nuevos elementos quedan instanciados en false.

Aunque ahora mismo solo hay implementados tres logros, el sistema está preparado para poder aumentar el número de logros sin que haya ningún problema. Esto se consigue gracias a los complementos "*Scroll Rect*" y "Máscara" que contiene este *popup*, en el que si se añaden

más “*Prefabs* Logro” (se explicarán más adelante en este mismo apartado) de los que caben en el *popup*, se pueden ver todos ellos arrastrando el ratón hacia arriba y hacia abajo.

En el juego solo hay implementados tres logros, pero para que se pueda observar bien cómo funciona, el componente “Máscara” está preparado para soportar más logros, se van a añadir 4 más provisionalmente para que se entienda bien observando imágenes. Estas imágenes se pueden observar en la figura 5.12 (componente mascara activado) y figura 5.13 (componente mascara desactivado).

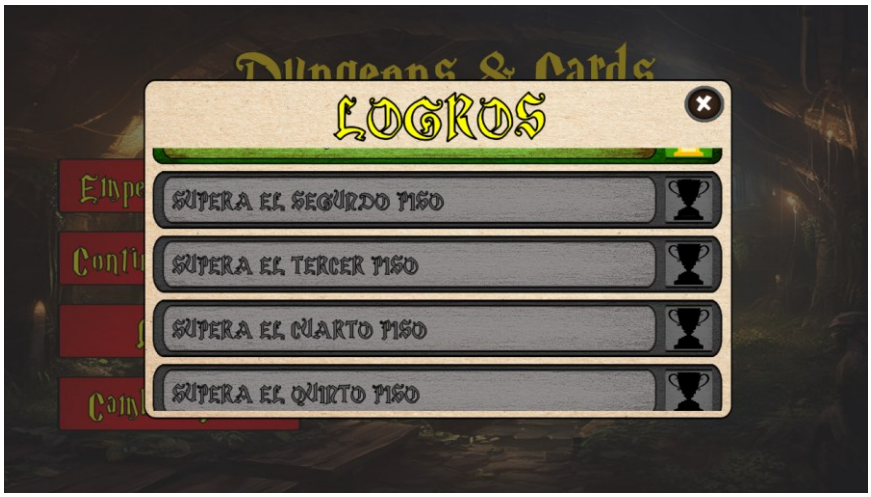

Figura 5.12: *Componente máscara activado*

Figura 5.13: *Componente máscara desactivado*

Estos logros nuevos han sido eliminados después de hacer capturas de estas figuras, ya que han sido implementados solo para mostrar el funcionamiento del *popup* en esta memoria.

Respecto al *prefab* del logro, puede estar activado o estar desactivado.

Si está activado, tiene un fondo verde oscuro. En la parte derecha, se encuentra un símbolo de trofeo de color amarillo con un fondo verde más brillante. En la parte izquierda y central hay un fondo verde también brillante y con el texto del logro de color blanco.

Figura 5.14: *Prefab del logro activado*

Si está desactivado, tiene un fondo gris oscuro. En la parte derecha hay un símbolo de trofeo de color negro con un fondo gris un poco menos oscuro. En la parte izquierda y central, se encuentra un fondo grisáceo y con el texto del logro de gris más oscuro.

Figura 5.15: *Prefab del logro desactivado*

Recalcar que las texturas rugosas y los elementos de estos *popups* han sido hechos a mano usando Paint.NET.

Cambiar Perfil

Al apretar el botón de cambiar perfil, se carga la escena de seleccionar perfil y se eliminan todo el resto de las escenas, quedando así el juego como si estuviese recién iniciado.

5.3. Generación Procedural del Terreno

Script DungeonGeneratorData

En primer lugar, se ha creado un script llamado DungeonGeneratorData que es un *ScriptableObject*, esto significa que se pueden crear instancias de este script desde Unity para poder tener diferentes instancias con distintos valores en sus atributos.

Estos atributos son el número de caminos del piso, el número de habitaciones mínimo que tendrá cada camino, el número de habitaciones máximo que tendrá cada camino y el nombre del piso que se está generando. Actualmente, solo hay tres pisos implementados cuyos nombres son “Basement”, “Jungle” y “Volcano”.

La funcionalidad de este *script* es poder indicar estos atributos a los *scripts* que se encargan de generar el terreno, de esta manera se podrá cambiar el tamaño y la forma de los pisos generados simplemente utilizando una instancia del DungeonGeneratorData u otra.

Script DungeonCrawler

Se ha hecho un s*cript* llamado DungeonCrawler, aunque por simplicidad y para que sea más fácil entender por el lector de esta memoria de que se está hablando, a partir de ahora se va a referir a este s*cript* como “Camino”.

El s*cript* “Camino” tiene un Vector2Int que guarda su posición (por ejemplo, (0,0)) y un método principal llamado Move.

Siempre que se llame al método Move, la posición se moverá en 1 en una dirección aleatoria (arriba, abajo, izquierda o derecha) y devolverá esta posición. Por ejemplo, si llamamos al método Move cuando el "Camino" está en la posición (0,0), el método devolverá (-1,0), (1,0), (0,1) o (0,-1) de forma aleatoria. El Vector2Int posición del "Camino" también se actualizará.

El Método Move no elige la dirección de forma 100% aleatoria, cuando elige la dirección a la que moverse, comprueba el número de habitaciones vecinas que tendría si elige esa dirección. Si este número es mayor que 1, el 98% de las veces elegirá otra dirección aleatoria. Esto se debe a que se intenta minimizar el número de *loops* que puedan darse en el terreno para que los caminos queden más claros y estos den una sensación de ramas de un árbol.

Hay dos motivos principales para que esto ocurra solo el 98% de las veces y no el 100%.

El primero es que, aunque queremos evitar los posibles *loops*, que haya variedad en los pisos y que de vez en cuando el jugador se encuentre alguno, le da más vida y mayor factor de aleatoriedad al juego.

El segundo motivo es que podría darse la situación de que las cuatro posibles direcciones del método Move tengan más de una habitación adyacente, por lo que entraría en un bucle infinito de intentar cambiar de dirección, cosa que se evita con el 98%.

***Script* DungeonCrawlerController**

Se ha creado un *script* llamado DungeonCrawlerController, y por simplicidad y para que sea más fácil entender por el lector de esta memoria, a partir de ahora se va a referir a este Script como "ControladorCaminos".

La finalidad de este s*cript* es generar una lista con todas las posiciones de las habitaciones del piso.

Cuando se llama a este s*cript*, se le pasa como argumento la instancia de un DungeonGeneratorData con los datos necesarios para crear las posiciones del piso.

Este script tiene una lista llamada "posicionesVisitadas" que se inicializa vacía.

Cuando se llama a este script, se crean tantas instancias del script "Camino" con posición (0,0) como indique el DungeonGeneratorData.

Por cada camino, se elige el número de iteraciones que se llamará a su método Move, este número es un número aleatorio entre el número de iteraciones mínimo que indica el DungeonGeneratorData y el máximo que indica el DungeonGeneratorData.

Luego, se llama al método Move tantas veces como se ha decidido y se van guardando los valores que devuelve en "posicionesVisitadas".

Si simplemente se hiciera esto por cada "Camino" que se ha creado, se quedaría un piso con una serie de caminos rectos que no se ramificarían. Para que se ramifiquen y haya un poco más de variedad en el piso, cuando se acaba un camino, dependiendo del número de iteraciones de este se elige el número de ramas. Cuando más grande sea el camino, más ramas tendrá.

Una vez se tiene el número de ramas que tendrá el camino, por cada rama se crea una nueva instancia del *script* "Camino" en una posición aleatoria de las que hay en el camino actual y se llama a su método Move un número determinado de veces.

Ese número determinado de veces se calcula de la siguiente manera, si se decide ramificar en la habitación 3 del camino y en total el número de iteraciones del camino es 8, se llamaría un total de 8-3 (5) veces al método Move de la rama, de esta manera las dos ramas que se quedan tienen la misma longitud.

El proceso de ramificación se repite tantas veces como se ha declarado antes y todo el proceso se repite por cada número de caminos que hay en el *script* DungeonGeneratorData.

Una vez ha terminado el proceso, el *script* devuelve la lista de "posicionesVisitadas" que ha sido calculada. Esta lista simplemente tiene todas las posiciones en las que habrá habitaciones en el piso, pero no indica que habitación habrá en cada posición.

***Script* DungeonGenerator**

Se ha implementado un *script* llamado DungeonGenerator, este *script* se encarga de recibir la lista de Vector2Int que genera el *script* DungeonCrawlerController y elegir qué tipo de habitación va en cada posición.

Lo primero que hace el *script*, una vez que ha recibido esta lista, es crear una nueva lista, pero esta vez ordenando las posiciones de las habitaciones por distancia. Esto se consigue aplicando un algoritmo BFS.

Para conseguir esta lista ordenada por distancia, lo primero que se hace es crear un diccionario donde la llave es un Vector2Int (una posición) y el valor es un *Int* (un número), este diccionario guardará la distancia a la que está cada habitación.

Para inicializar los valores de este diccionario se aplica un algoritmo BFS, se empieza inicializando a 0 la distancia de la habitación (0, 0) y añade a una cola.

A continuación, por cada elemento de la cola, se quita el primer elemento y a cada habitación adyacente, que aún no tenga posición, se le asigna la posición del elemento que se esté procesando más uno. Después, se añade cada habitación adyacente a la cola si aún no ha sido procesada.

Esto se repite hasta que la cola esté vacía y se hayan procesado todas las habitaciones.

En el momento en el que se piensa cómo calcular las distancias, puede parecer que lo más sencillo es sumar el valor absoluto de las coordenadas de cada habitación. Esto puede ser erróneo en caso de que el piso presente "curvas", por ese motivo se ha decidido aplicar un algoritmo BFS. En las siguientes imágenes se puede ver la comparación entre usar el algoritmo BFS para calcular distancias (Figura 5.16) y sumar el valor absoluto de las coordenadas (Figura 5.17). Observando las figuras podemos ver que la habitación marcada en amarillo presentaría una distancia errónea en caso de sumar el valor absoluto de las coordenadas.

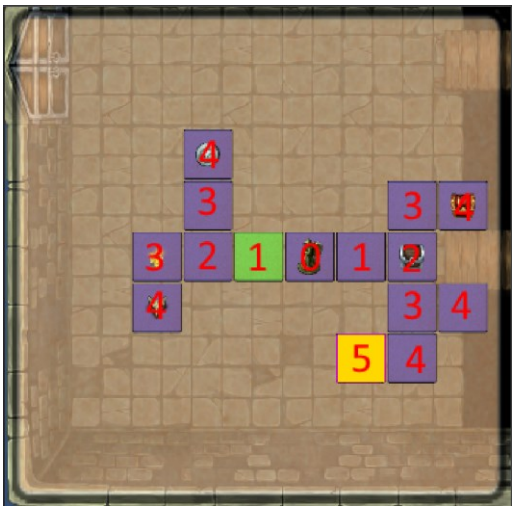

Figura 5.16: *Algoritmo BFS*

Figura 5.17: *Suma del valor absoluto de las coordenadas*

Cuando el *script* tiene las diferentes distancias guardadas en el diccionario, se crea una lista llamada "deadEndsByDistance" en la que se obtienen los diferentes "callejones sin salida" o *dead ends* (habitaciones con solo una habitación adyacente). Esta lista se obtiene utilizando la librería LINQ y con una sola línea de código:

```
deadEndsByDistance = dungeonRooms.Distinct().Where(o => numOfNeighbors(o) == 1).OrderByDescending(o => distanceRoom[o]).ToList();
```

Una vez se tiene esta lista, se comprueba el número de elementos que contiene y si es menor a 3, se vuelve a llamar al script DungeonCrawlerController para generar un nuevo piso, ya que se necesitan al menos 3 *dead ends*.

Cuando ya se tenga la lista "deadEndsByDistance" y tenga 3 o más elementos, se empieza a llamar al *script* RoomController, que es el encargado de recibir la información de donde va cada habitación e instanciarla. La información enviada a este script en cada ocasión es: Nombre de la habitación, posición X y posición Y.

Siempre que se envía algún elemento al *script* RoomController, se quita la posición enviada de las listas para evitar instanciar dos habitaciones en un mismo lugar.

La primera habitación que se envía es la inicial con la posición (0, 0).

La segunda habitación que se envía, es la del Jefe, que siempre estará en el *dead end* más alejado del inicio, por lo que se envía el nombre de la habitación del Jefe con la posición del primer elemento de la lista "deadEndsByDistance".

La tercera habitación enviada es la del Tesoro y se envía con una posición aleatoria de la lista del "deadEndsByDistance".

La cuarta habitación enviada es la de la Tienda que se envía con una posición aleatoria de los *dead ends* que quedan.

Después, se calcula el número de habitaciones de Hoguera a instanciar, se calcula con la fórmula:

*Habitaciones Totales * 0.15 + (1 un 15% de las veces) - 1 (un 5% de las veces)*

De esta manera se le añade un componente de aleatoriedad. Una vez se tiene el número, se envían ese número de habitaciones de Hoguera al *script* RoomController.

A continuación, se calcula el número de habitaciones de Cofre a instanciar, se calcula con la fórmula:

*Habitaciones Totales * 0.075 + (1 un 10% de las veces) - 1 (un 1% de las veces)*

De esta manera se le añade un componente de aleatoriedad. Una vez se obtiene el número, se envían ese número de habitaciones de Cofre al *script* RoomController.

Para las posiciones que queden por cubrir, se elige una habitación de enemigos aleatoria y se llama al *script* RoomController con esa posición.

***Script* RoomController**

El script RoomController es el que se encarga de recibir el nombre de la habitación a crear y sus posiciones e instanciarlas.

Siempre que recibe el nombre de una habitación, se encarga de añadirle delante el nombre del piso actual. Por ejemplo, existen un total de tres salas de la tienda, BasementShop, JungleShop y VolcanoShop. pero el *script* DungeonGenerator solo le ha enviado "Shop" y una posición. El *script* RoomController se encarga de saber qué piso se está creando e instanciar las habitaciones que hacen falta.

Además, una vez ha instanciado las habitaciones y sabe sus posiciones, indica a todas las habitaciones que vecinos tienen y cuáles no, para que estas eliminen las puertas de las habitaciones que no conecten con ninguna.

5.4. Personaje

Movimiento

Para el movimiento de personaje se obtienen dos números “InputVertical” y “InputHorizaontal”, cuyos valores se encuentran entre -1 y 1.

Estos números se obtienen utilizando el sistema de *Input* de Unity, de esta manera si el usuario aprieta la tecla “W” o la flecha hacia arriba, “InputVertical” sería 1 y “InputHorizaontal” 0.

Se crea un Vector2 utilizando estos valores y se normaliza, así el movimiento será uniforme y el personaje no se moverá más si se mueve en diagonal.

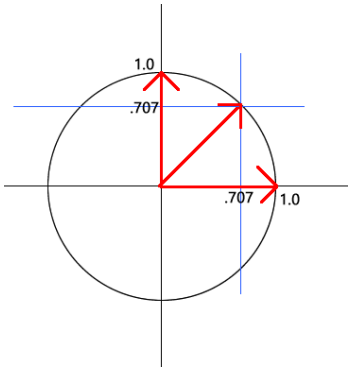

Figura 5.18: *Vector2*

Este Vector se suma a la posición del jugador multiplicando por una variable velocidad y por el intervalo en segundos desde que se procesa un *frame* del juego y otro. Debido a esto, el personaje se moverá igual de rápido tanto en un ordenador con mucha potencia como con un ordenador con menos potencia.

Gracias a la variable velocidad, se puede modificar la velocidad del personaje en caso de querer cambiarla en algún momento. Por ejemplo, si el jugador encontrase un objeto que aumente su velocidad, se cambiaría esta variable.

Además, el objeto del personaje tiene un componente llamado “Box Collider 2D”. Este componente “choca” con los *colliders* de otros objetos con el fin de que no los atraviese, como pueden ser las paredes. Este componente está representado con el rectángulo verde en la figura 5.19.

Figura 5.19: *Componente Box Collider 2D*

Animaciones

El personaje tiene dos animaciones principales, la animación *Idle* y la animación *Walk*. La animación *Idle* es una sucesión de *sprites* que se pueden ver en la figura 5.20.

Figura 5.20: *Animación Idle*

La animación *Walk* también es una sucesión de *sprites* que se puede ver en la figura 5.21.

Figura 5.21: *Animación Walk*

Para cambiar de una animación a otra se utiliza el Animator de Unity. Este Animator funciona como una máquina de estados. En la figura 5.22 se pueden observar los diferentes posibles estados y sus transiciones.

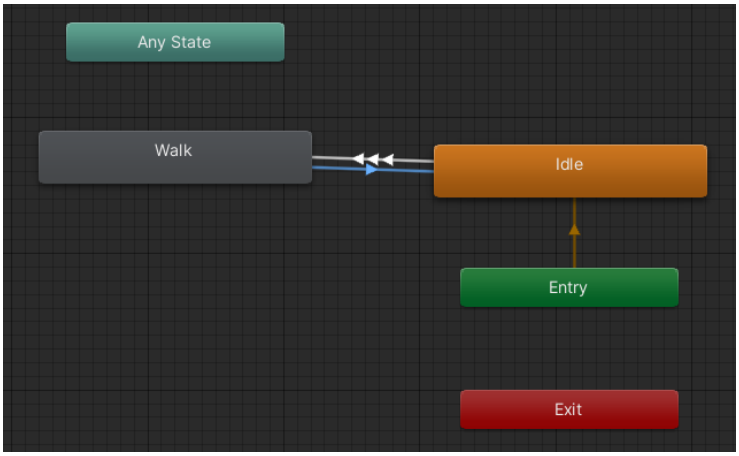

Figura 5.22: *Posibles estados y sus transiciones*

La transición de *Idle* a *Walk* se llevará a cabo siempre que el *float* Magnitud del vector *Input ("InputHorizontal","InputVertical")* sea mayor a 0.

La transición de *Walk* a *Idle* se llevará a cabo siempre que el *float* Magnitud del vector *Input ("InputHorizontal","InputVertical")* sea mayor a -0.01 y menor a 0.01.

Como se ha podido observar en las figuras 5.20 y 5.21 el personaje siempre estará orientado a la derecha. Para orientarlo a la izquierda, cuando el usuario decida moverse hacia la izquierda, se escala el *sprite* del personaje a -1 en la X quedando así orientado a la izquierda. En caso de volver a moverse a la derecha, se volverá a escalar a 1.

5.5. Puertas y Habitaciones

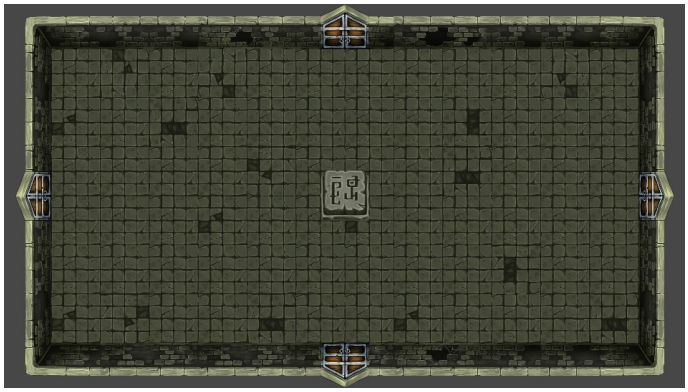

Figura 5.23: *Habitación 1*

A nivel gráfico, las habitaciones, el suelo y las paredes han sido creados utilizando *Tilemaps* de Unity.

Estos *Tilemaps* son una cuadrícula que se puede rellenar o pintar con los diferentes *Tiles* que se tengan. Estos *Tiles* han sido comprados en la *Unity Store*, ya que crear este tipo de *asset* sin tener conocimiento artístico es muy complicado y actualmente es imposible obtener buenos resultados mediante Inteligencia Artificial.

En las figuras 5.24 y 5.25 se pueden observar los diferentes *Tiles* a los que se ha tenido acceso gracias a esta compra.

Figura 5.24: *Tiles 1*

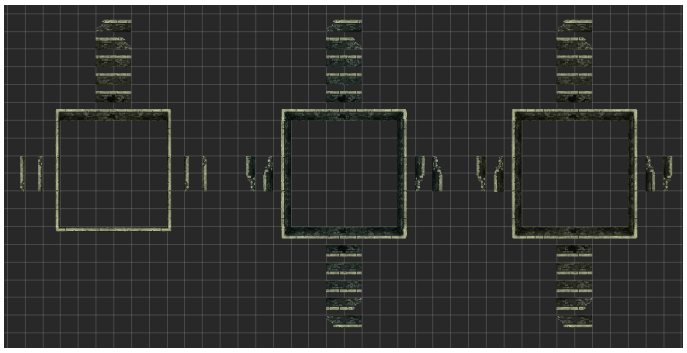

Figura 5.25: *Tiles 2*

Aunque solo se tuviese acceso a 4 tipos diferentes de suelo, se ha jugado aplicando filtros de colores para tener más variedad. Un buen ejemplo se puede observar en la figura 5.26.

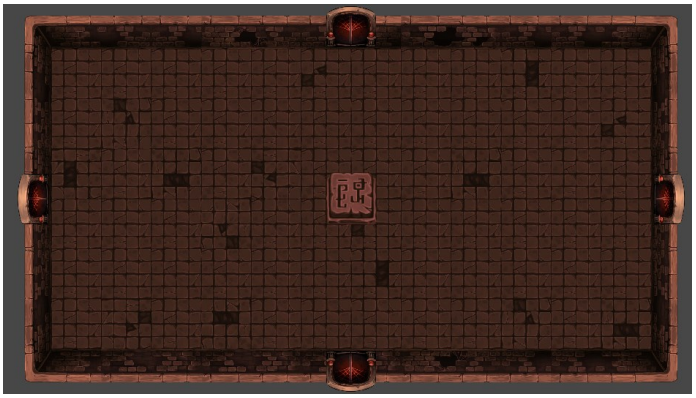

Figura 5.26: *Habitación 2*

Los gráficos de las puertas también han sido comprados. Cada puerta tiene 3 gráficos, puerta abierta, puerta cerrada y el fondo como se puede observar en la figura 5.27.

Figura 5.27: *Puerta*

Esto ocurre en todos los tipos de puertas y en todas las direcciones posibles (arriba, abajo, izquierda y derecha).

Estos tres *sprites* se colocan en la misma posición, pero se ven en el siguiente orden: “Puerta Cerrada”, “Puerta Abierta”, “Fondo”.

Para decidir abrir una puerta, simplemente se desactiva el *sprite* “Puerta Cerrada”. El resultado se puede observar en la figura 5.28.

Figura 5.28: *Puerta cerrada y puerta abierta*

A nivel mecánico, las habitaciones tienen un objeto invisible con una serie de *colliders* para que el jugador no pueda atravesar paredes. Estos *colliders* se pueden ver resaltados en líneas verdes en la figura 5.29.

Figura 5.29: *Colliders*

Los *colliders* resaltados en líneas azules en la figura 5.29 están vinculados a cada puerta, de esta manera, cuando la puerta recibe la orden de abrirse, desactiva su *sprite* "Puerta Cerrada" y desactiva el *collider* correspondiente para que el jugador pueda pasar por ahí.

5.6. Minimapa

Gráficamente, el minimapa tiene dos objetos principales, el cuadrado con el componente "Máscara" y el contenido.

El objeto Contenido cuando se crea el piso, recibe toda la información de las posiciones de las habitaciones e instancia los diferentes objetos cuadrados (Figura 5.30) en ciertas posiciones simulando la disposición de las habitaciones del piso.

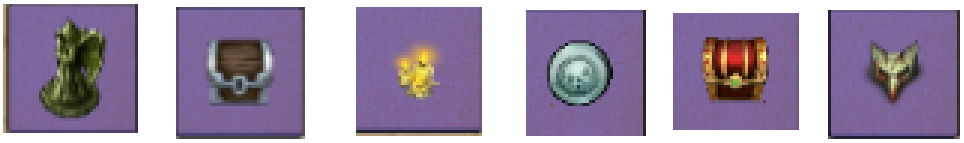

Figura 5.30: *Objetos cuadrados*

El Objeto máscara tiene el fondo que se ve del minimapa y gracias a un componente "Máscara" impide que el contenido se salga del minimapa. En la figura 5.31 se puede observar la diferencia entre tener este componente y no tenerlo.

Figura 5.31: *Efecto del componente Máscara*

Siempre que se entra a una nueva habitación, se cambia la posición del contenido para que la habitación que se vea en el centro sea la habitación actual. Además, se cambia el color de la habitación en la que se encuentra el personaje a verde.

Pulsando el botón Tabulador, se cambia el tamaño del componente máscara, pudiendo así hacer más grande el minimapa para que el usuario pueda observar más habitaciones del piso. Esto se puede ver en la figura 5.32.

Figura 5.32: *Minimapa ampliado*

Si se vuelve a pulsar el tabulador, el minimapa volverá al tamaño original.

5.7. Cartas

En este proyecto se encuentran dos tipos principales de cartas: Cartas de artefactos y Cartas de esbirros.

En la figura 5.33 se puede observar un ejemplo de cada tipo de carta.

Figura 5.33: *Carta de Esbirro y carta de Artefacto*

A nivel maquetación son muy similares. Ambas tienen el nombre de la carta en la parte superior, una imagen y una descripción.

Pero las cartas de esbirros también tienen una raza (Dragón, Humano, Elfo, Enano…) y una clase (Druida, Mago, Brujo, Guerrero…).

Además, las cartas de esbirro también tienen una serie de estadísticas. Estas estadísticas serán: el maná, la fuerza, la destreza, la magia, la armadura y la vida.

Para elegir las estadísticas de cada carta, inicialmente se tenía pensado hacer mediante el uso de una Ontología, sin embargo, al no ser un sistema muy complejo, se ha optado por programar una estructura de estadísticas manualmente.

Cada clase y cada raza tendrá un valor de cada estadística asociado, siempre que se cree una carta, está comprobará su raza y clase y obtendrá las estadísticas automáticamente sumando las estadísticas de su raza y de su clase que obtiene a partir de la estructura. Por ejemplo, la raza Dragón tiene guardada una fuerza de 4 en la estructura, y la clase Druida tiene guardado una fuerza de 2, por lo que una carta Dragón/Druida tendrá una fuerza de 6 (2+4). Esto se repite para todas las estadísticas.

En la figura 5.34 se puede observar los valores de las estadísticas de cada clase y raza.

	Draconido	Dragón	Elfo	Enano	Gnomo	Humano	Mediano	Semielfo	Semiorco
Maná	1	3	1	1	0	1	0	1	1
Fuerza	2	4	4	5	2	3	1	4	4
Destreza	1	5	3	2	2	3	4	3	0
Magia	3	7	5	0	0	3	0	1	0
Armadura	1	4	1	5	2	3	1	3	4
Vida	3	6	3	3	3	3	4	5	5

	Bardo	Brujo	Bárbaro	Clerigo	Druida	Explorador	Guerrero	Hechizero	Mago	Monje	Paladín	Picaro
Maná	0	0	0	0	0	0	0	0	1	0	2	0
Fuerza	0	0	4	1	2	1	3	0	0	0	3	2
Destreza	1	0	0	2	3	5	0	1	0	3	0	4
Magia	3	3	0	3	1	0	0	3	5	3	3	0
Armadura	0	0	2	0	0	0	4	0	0	0	3	0
Vida	0	2	2	0	2	0	4	0	0	0	3	0

Figura 5.34: *Estadística de cada clase y raza*

Se ha decidido utilizar este tipo de estructura para facilitar la creación de nuevas cartas. Para crear nuevas cartas simplemente hay que crear una copia del objeto "Carta Base", añadirle una nueva imagen, añadirle el nombre y la descripción y seleccionar su raza y su clase, de las estadísticas se ocupa el propio juego.

Cada carta también tiene un ID propio.

5.8. Sistema de Inventario

Cartas y artefactos

Respecto al sistema de inventario se existen 2 inventarios principales, el Mazo de Esbirros donde se guardan las cartas de esbirro y la Bolsa de Artefactos donde se guardan las cartas de artefacto. Los dos funcionan de la misma manera.

Como las cartas de esbirro y las cartas de artefactos son *GameObjects* de Unity, hacer diccionarios y listas de estos puede generar muchos problemas. Por lo que se va a tener un objeto base llamado "CardManager" el cual funcionará como un diccionario donde las *Keys* son los IDs de las cartas (*int*) y el valor un *prefab* del *gameobject* de la carta en cuestión. "CardManager" funcionará de diccionario tanto de cartas de esbirro como de cartas de artefactos.

De esta manera, en el inventario de cada partida solo se debe tener una lista de *INTs* para saber qué esbirros se tiene en el mazo. Lo mismo ocurre con los artefactos.

A nivel de maquetación, tenemos dos botones en la esquina inferior derecha de la pantalla, uno abre el mazo y el otro la bolsa de artefactos.

Figura 5.35: *Mazo y bolsa de artefactos*

Cuando se aprietauno de estos dos botones, se abre un *popup* (que en verdad es un *GameObject*) el cual lee esta lista de *INTs* guardada en el Inventario y "genera" los *GameObjects* llamando al "CardManager" que funciona como diccionario.

Para que se vea bien el *popop*, se tiene un objeto que funciona como máscara (para que los objetos no se salgan del margen) y tenemos un "*Scroll Rect*" para que, en caso de tener demasiadas cartas, poder verlas todas bajando o subiendo como si fuese una página web.

En la figura 5.36 se puede observar la diferencia entre tener el componente máscara activado y desactivado.

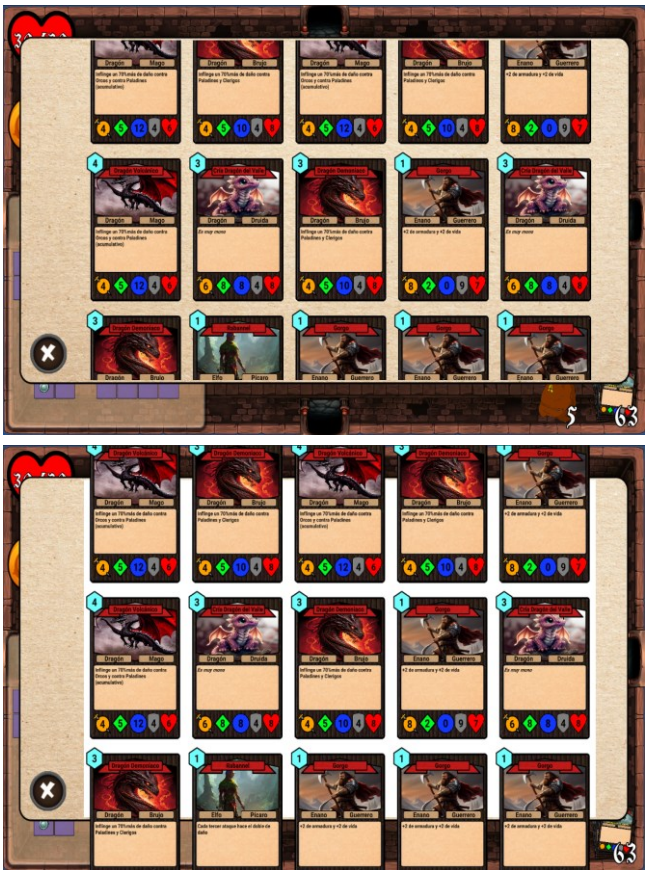

Figura 5.36: *Componente Máscara activado y desactivado*

Hay un *popup* como estos para Artefactos y otro para Esbirros.

Oro

El Inventario también guarda un *Int* que representa el número de Monedas de Oro que tiene el jugador.

Figura 5.37: *Oro*

Este número se puede saber en partida gracias a un icono con forma de moneda situado en la parte superior izquierda de la pantalla.

Siempre que se actualice este número, se lanza una animación hecha con el *Animator* de Unity en la cual el número del icono tiembla y parpadea varias veces, en color verde si se ha añadido dinero y en color rojo en caso de que se quite.

Vida

El Inventario también guarda dos *Ints* que representan la vida actual del personaje y la vida máxima del personaje.

Estos números se pueden saber en partida gracias a un icono con forma de corazón situado en la parte superior izquierda de la pantalla.

Figura 5.38: *Vida*

Siempre que se actualice cualquiera de los dos números, se lanza una animación hecha con el *Animator* de Unity y el número tiembla y parpadea varias veces, en color verde si el valor al que cambia es mayor y en color rojo en caso de que el nuevo valor sea menor.

Evidentemente, la vida nunca puede superar a la vida máxima y en caso de llegar a 0, se acaba la partida.

Cuando se llega a 0, se indica al .Json del perfil actual que no hay ninguna partida en curso, aparece una pantalla de “Has muerto” y se carga la escena del menú principal. Se puede observar la pantalla te has muerto en la figura 5.39.

Figura 5.39: *Has muerto*

La vida máxima está limitada a 99 y no puede ser menor a 5. En caso de que al quitar vida máxima al personaje la vida actual sea superior a la vida máxima, la vida actual se adaptará y será igual a la vida máxima.

5.9. Sistema de Recompensas

***Pickups* de Oro**

Habrá varios tipos de *pickups* de oro, pero todos funcionarán de la misma manera: Cuando el jugador pasa por encima (se detecta con un Box Collider), desaparecerán usando un efecto de partículas de Unity y se añadirá una cierta cantidad de oro al Inventario.

La cantidad de partículas que saldrán dependerá del tamaño de la pila de oro.

La figura 5.40 muestra los seis diferentes tipos de *pickups* de oro.

Figura 5.40: *PickUps de oro*

El sistema de partículas crea una cantidad determinada de partículas (dependiendo de lo grande que sea el *pickup* de oro) con forma de moneda de oro. Y les añade una velocidad inicial hacia arriba con forma de cono. También les añade un componente para que se le aplique gravedad. De esta manera, da la sensación de que "esparces" las monedas cuando el jugador pasa por encima.

En la figura 5.41 se pueden observar las monedas de este efecto de partículas.

Figura 5.41: Monedas de oro

***Pickup* de Esbirro**

El *pickup* de esbirro tiene forma de carta. Esta carta rota sobre sí misma utilizando el animador de Unity, mostrando un dorso rojo cuando se ve por detrás. En la figura 5.42 se puede ver como se ve por delante y por detrás.

Figura 5.42: *Pickup de Esbirro*

Además de rotar sobre sí misma, también se eleva y baja lentamente utilizando también el *Animator* de Unity, dando la sensación de que está flotando.

Cuando el jugador pasa por encima de este pickup (se detecta utilizando un Box Collider), aparece el *popup* que se puede observar en la figura 5.43.

Figura 5.43: *Popup 1*

En esta pantalla el jugador podrá elegir si quiere añadir alguna de las cartas que se le muestran a su mazo de esbirros. También podrá decidir no añadir ninguna.

Cuando se genera esta pantalla, se genera una lista de números aleatorios, por ejemplo, {1,2,3,4}, y se convierten en *GameObjects* de carta utilizando el diccionario de Esbirros del "CardManager" como se explica en el apartado 5.7.

Además de generar las cartas, se les añade un componente *Script* que hace que cuando se hace clic sobre ellas, se añaden al inventario y se cierra el *popup*.

Destacar que, aunque por defecto se generen cuatro cartas para elegir, está preparado para que, si en algún momento se necesita generar más o menos cartas, simplemente cambiando una

variable, esto ocurra. Está preparado para que en un futuro se pueda implementar un objeto que cambie la cantidad de cartas entre las cuales puedes elegir.

En caso de que haya más cartas de las que caben en la pantalla, el jugador podrá verlas todas arrastrando hacia la derecha e izquierda con el ratón gracias a un componente "*Scroll Rect*" de Unity. Las cartas que no se ven están ocultas utilizando un componente "máscara". En la figura 5.44 se puede observar la diferencia entre tener este componente activado o no.

Figura 5.44: *Componente Máscara activado y desactivado 1*

***Pickup* de Artefacto**

El *pickup* de artefacto tiene forma de carta. Esta carta rota sobre sí misma utilizando el animador de Unity, mostrando un dorso rojo cuando se ve por detrás. En la figura 5.45 se puede ver como se ve por delante y por detrás.

Figura 5.45: Pickup de Artefacto

Además de rotar sobre sí misma, también se eleva y baja lentamente utilizando también el *Animator* de Unity, dando la sensación de que está flotando.

Cuando el jugador pasa por encima de este *pickup* (se detecta utilizando un Box Collider), aparece el *popup* que se puede observar en la figura 5.46.

Figura 5.46: *Popup 2*

En esta pantalla el jugador podrá elegir si quiere añadir alguna de las cartas que se le muestran a su bolsa de artefactos. También podrá decidir no añadir ninguna.

Cuando se genera esta pantalla, se crea una lista de números aleatorios, por ejemplo, {1,2,3,4}, y se convierten en *GameObjects* de carta utilizando el diccionario de Artefactos del "CardManager" como se explica en el apartado 5.7.

Además de generar las cartas, se les añade un componente *Script* que hace que cuando se hace clic sobre ellas, se añaden al inventario y se cierra el *popup*.

Destacar que, aunque por defecto se creen cuatro cartas para elegir, está preparado para que, si en algún momento se necesitan generar más o menos cartas, simplemente cambiando una variable, esto ocurra. En un futuro se podría implementar un objeto que cambie la cantidad de cartas entre las cuales puedes elegir.

En caso de que haya más cartas de las que caben en la pantalla, el jugador podrá verlas todas arrastrando hacia la derecha o izquierda el ratón gracias a un componente "*Scroll Rect*" de Unity. Las cartas que no se ven, están ocultas utilizando un componente "máscara". En las siguientes figuras se puede observar la diferencia entre tener este componente activado o no.

Figura 5.47: *Componente Máscara activado y desactivado 2*

Probabilidad de Recompensas

Estos *pickups* pueden aparecer al jugador por tres motivos diferentes: Porque se ha superado una sala, porque se ha abierto un cofre o porque se ha abierto un cofre del tesoro.

Evidentemente, abrir un cofre del tesoro otorgará más recompensas al jugador que superar una sala. Para ello se ha creado un objeto "Dropper" que dándole una lista de recompensas y de probabilidades de cada recompensa, se encargue de generar las recompensas correspondientes.

En la figura 5.48 aparecen tablas donde se puede ver la probabilidad de la cantidad de recompensas a soltar en cada situación y la probabilidad de que *pickup* soltar en cada situación.

Superar una Sala	
Cantidad	Probabilidad
1	15%
2	50%
3	25%
4	10%
Tipo de PickUp	Probabilidad
Oro Pequeño	23%
Oro Mediano	18%
Oro Grande	13%
Oro Pequeño con joyas	9%
Oro Mediano con joyas	5%
Oro Grande con joyas	2%
Carta de Esbirro	20%
Carta de Artefacto	10%

Abrir Cofre Normal	
Cantidad	Probabilidad
3	15%
4	50%
5	25%
6	10%
Tipo de PickUp	Probabilidad
Oro Pequeño	5%
Oro Mediano	7%
Oro Grande	11%
Oro Pequeño con joyas	15%
Oro Mediano con joyas	8%
Oro Grande con joyas	4%
Carta de Esbirro	30%
Carta de Artefacto	20%

Abrir Coofre del Tesoro	
Cantidad	Probabilidad
6	15%
7	50%
8	25%
9	10%
Tipo de PickUp	Probabilidad
Oro Pequeño	0%
Oro Mediano	0%
Oro Grande	7%
Oro Pequeño con joyas	13%
Oro Mediano con joyas	15%
Oro Grande con joyas	10%
Carta de Esbirro	30%
Carta de Artefacto	25%

Figura 5.48: *Tablas de probabilidades*

5.10. Habitaciones

Los pisos están compuestos por diferentes habitaciones. Estas habitaciones podrán ser de dos tipos: habitaciones normales y habitaciones especiales.

Habitaciones Normales

Las habitaciones normales son aquellas que no tienen ningún icono en el minimapa. En ellas debería haber enemigos. Se cierran todas las puertas cuando se entre en ellas y no se abren hasta que se supere la habitación.

El sistema de combate estará hecho por mi compañero Pau Vidal Ramón en su TFG. Como él no lo tiene para esta entrega, se ha dejado una placa de presión como *placeholder* que cuando es pulsada, se abren las puertas y se otorga la recompensa de haber derrotado a los enemigos.

En la figura 5.49 se puede observar la placa de presión sin pulsar y pulsada junto con las recompensas que otorga.

Figura 5.49: *Placa de presión*

Habitaciones Especiales

Habitación de inicio

Está señalada en el minimapa con el ícono de una estatua que muestra la imagen 5.50.

Figura 5.50: *Icono de la habitación de inicio*

Es la sala en la que empieza el personaje al inicio de cada piso. Se caratcteriza por tener varias estatuas de piedra.

Las puertas estarán abiertas para que directamente se pueda ir a explorar el resto del piso.

Figura 5.51: *Habitación de inicio*

Habitación de la Vela Mágica

Está marcada en el minimapa con el icono de una vela.

Figura 5.52: *Icono de la habitación de la Vela Mágica*

Puede haber más de una en cada piso. No es una sala hostil, por lo que las puertas están siempre abiertas.

Se caracteriza por tener una vela mágica gigante en el centro de la sala. Esta vela empieza estando apagada y si el personaje se acerca, la vela se enciende y cura al personaje un 33% de la vida máxima.

La hoguera detecta que el jugador está cerca gracias a un componente "Circular Collider". Para curar la vida del personaje se llama al método Heal del *script* Inventario

En la figura 5.53 se muestra una habitación de la Vela Mágica con la vela encendida y apagada. Se puede ver el perímetro del "Circular Collider" en verde.

Las capturas de pantalla de la figura 5.53 han sido tomadas desde el Editor de Unity con el objetivo de que se pudiese observar el perímetro de "Circular Collider", en el juego no se observarán las líneas verdes ni los iconos.

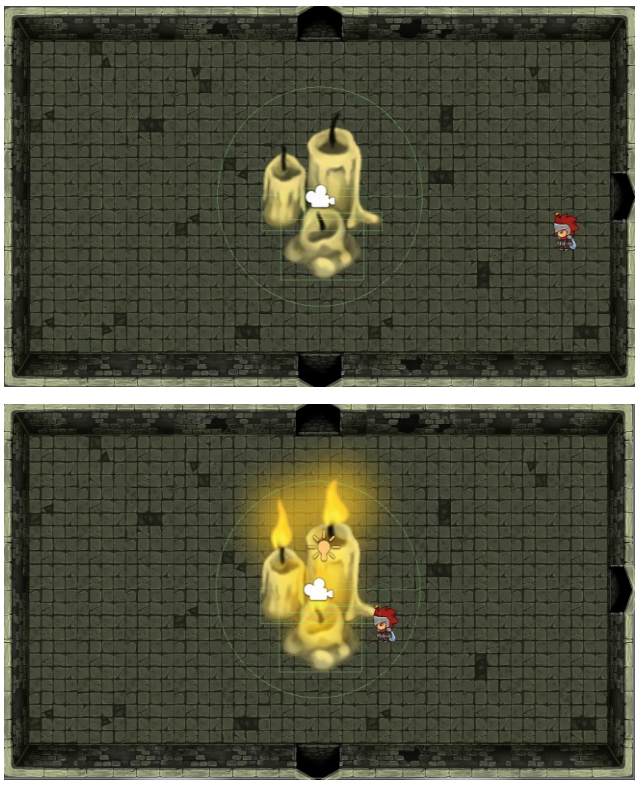

Figura 5.53: *Perímetro Circular Collider 1*

Habitación del Cofre

Está marcada en el minimapa con un icono de un cofre marrón.

Figura 5.54: *Icono de la habitación del Cofre*

Puede haber más de una en cada piso. No es una sala hostil, por lo que las puertas están siempre abiertas.

Se caracteriza por tener un cofre de madera en el centro de la sala. El cofre está cerrado y si el jugador se acerca, le dará las recompensas que se indican en el apartado 5.9.

El cofre detecta que el jugador está cerca gracias a un componente “Circular Collider”.

En la figura 5.55 se puede ver una habitación del Cofre con el cofre sin abrir y abierto. Se puede ver el perímetro del “Circular Collider” en verde.

Las capturas de pantalla de la figura 5.55 han sido tomadas desde el Editor de Unity con el objetivo de que se pudiese observar el perímetro de “Circular Collider”, en el juego no se observarán las líneas verdes ni los iconos.

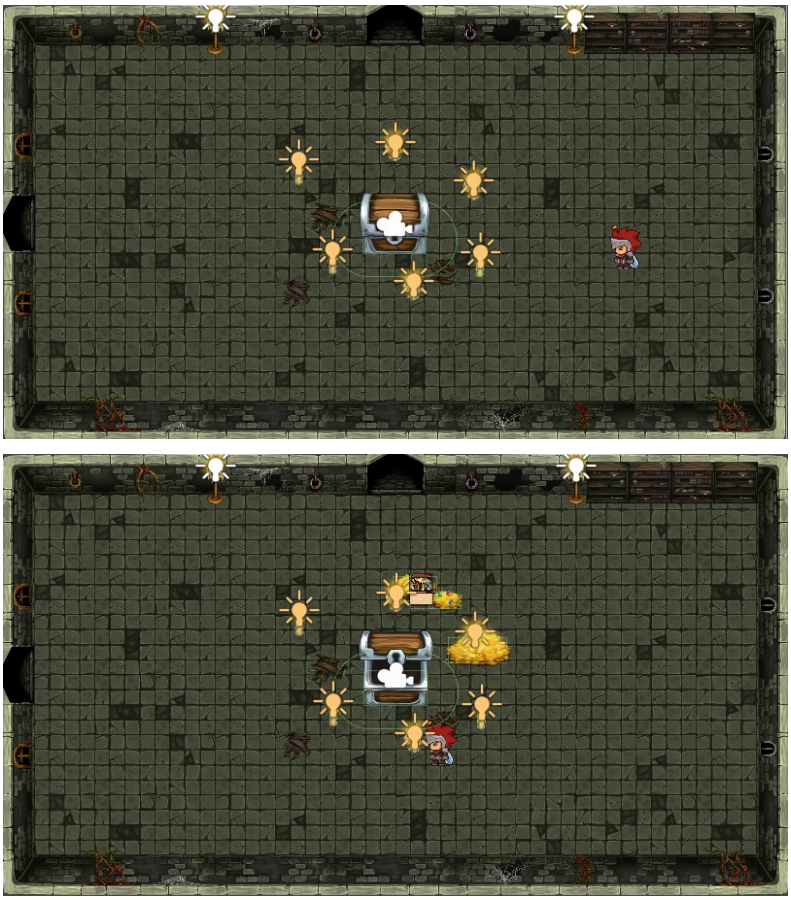

Figura 5.55: *Perímetro Circular Collider 2*

Habitación de la tienda

Está marcada en el minimapa con un icono de una moneda plateada.

Figura 5.56: *Icono de la habitación de la tienda*

Es única por piso y siempre está generada en un "callejón sin salida", es decir, solo tiene una habitación adyacente.

No es una sala hostil, por lo que las puertas están abiertas siempre.

En esta tienda se puede intercambiar el oro que tiene el jugador por diferentes *pickups* de cartas.

El precio de cada objeto que se puede comprar se puede ver debajo del mismo.

Para comprar los objetos simplemente hay que caminar por encima de ellos. Los objetos detectan que el jugador ha pasado por encima gracias a un "Box Collider"

Si el jugador pasa por encima de un objeto que no tiene suficiente dinero para comprar, pasará por encima y el número de oro que se ve en el *Overlay* parpadeará en rojo y temblará usando el *Animator* de Unity.

En las figuras 5.57, 5.58 y 5.59 se pueden observar diferentes tiendas.

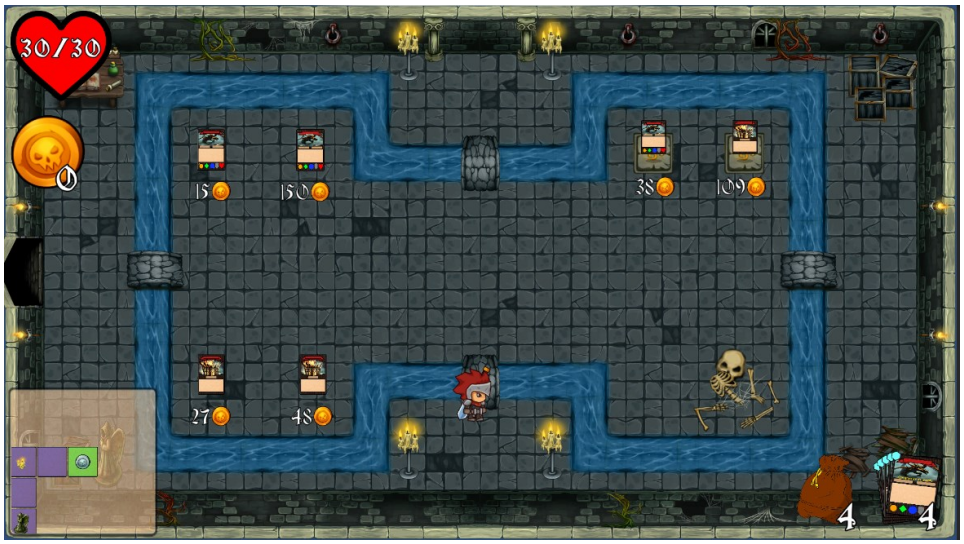

Figura 5.57: *Tienda 1*

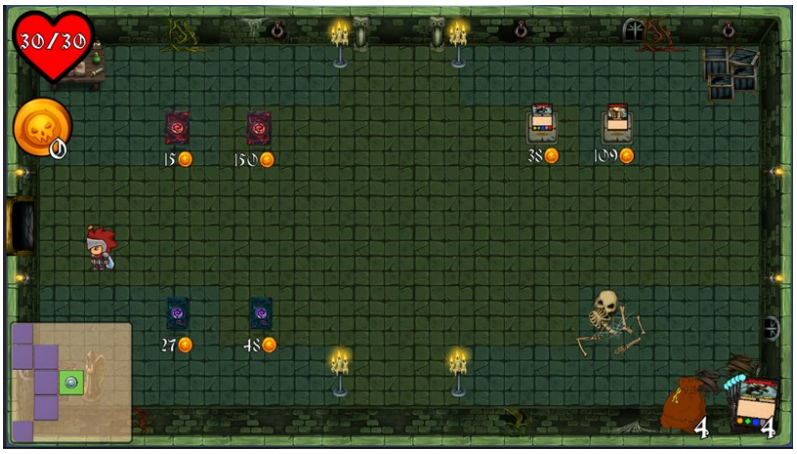

Figura 5.58: *Tienda 2*

Figura 5.59: *Tienda 3*

Habitación del Tesoro

Está marcada en el minimapa con un icono de un cofre dorado y rojo.

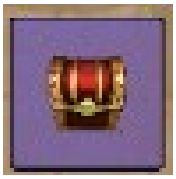

Figura 5.60: *Icono de la habitación del Tesoro*

Es única por piso y siempre estará generada en un "callejón sin salida", es decir, solo tendrá una habitación vecina.

No es una sala hostil, por lo que las puertas están siempre abiertas.

Se caracteriza por tener un cofre del tesoro en el centro de la sala. El cofre está cerrado y si el jugador se acerca, le dará las recompensas que se indican en el apartado 5.9.

El cofre detecta que el jugador está cerca gracias a un componente "Circular Collider".

En la figura 5.61 se puede ver una habitación del Cofre con el cofre sin abrir y abierto. Se puede ver el perímetro del “Circular Collider” en verde.

Las capturas de pantalla de la figura 5.61 han sido tomadas desde el Editor de Unity con el objetivo de que se pudiese observar el perímetro de “Circular Collider”, en el juego no se observarán las líneas verdes ni los iconos.

Figura 5.61: *Perímetro Circular Collider 3*

Habitación del Jefe

Está marcada en el minimapa con el icono de la figura 5.62.

Figura 5.62: *Icono de la habitación del Jefe*

La habitación del Jefe actúa como una habitación de enemigos normal, pero solo habrá un enemigo que será más grande y fuerte de lo normal.

Al ser una habitación hostil, las puertas se cierran cuando el jugador entra y no se abren, hasta que se derrote al Jefe del piso.

Cuando se derrote al Jefe, aparte de abrirse las puertas y soltar recompensas, también aparecen unas escaleras que, al entrar en ellas, llevan al siguiente piso.

Cuando se entra a las escaleras y se cambia de piso, se guarda toda la información relevante de la partida al .Json: el oro, la vida, la vida máxima, la lista de las cartas de esbirro, la lista de las cartas de artefacto y la información del piso generado.

En caso de entrar por estas escaleras, no se podrá retroceder.

Si el jugador se encuentra en el último piso y derrota al Jefe, aparecerá un cofre especial que al abrirlo surgirá un letrero de “Victoria!”, y pasados unos segundos, el usuario volverá al menú principal.

Al igual que en las habitaciones normales, el sistema de combate de esta habitación no está implementado, ya que pertenece al trabajo final de grado de mi compañero Pau Vidal Ramón, que lo presentará más adelante.

En su defecto, se ha utilizado una placa de presión como *placeholder* que cuando el jugador la pisa, simula que se ha derrotado al Jefe y se abren las puertas, soltando las recompensas mencionadas en el apartado 5.9 y aparecen las escaleras hacia el siguiente piso.

Las escaleras detectan que el jugador ha entrado en ellas gracias a un componente “box collider”.

En la figura 5.63 se puede observar un tipo de escalera vista desde el editor de Unity. Los “Box Collider” resaltados en azul actúan como pared para que el jugador no pueda pasar por ahí. El Box Collider en rojo actúa es el que detecta que el jugador ha entrado en él y se encarga de hacer que se avance al siguiente piso.

Figura 5.63: *Box Collider en una escalera*

El cofre especial detecta que el jugador está cerca gracias a un componente “Circular Collider”.

La captura de pantalla de la figura 5.64 ha sido tomada desde el Editor de Unity con el objetivo de que se pudiese observar el perímetro de “Circular Collider”, en el juego no se observarán las líneas verdes ni los iconos.

Figura 5.64: *Perímetro Circular 4*

En la figura 5.65 se puede observar la pantalla de Victoria previa al menú principal

Figura 5.65: *Victoria*

5.11. Menú de Pausa

Cuando el usuario apriete la tecla *Esc* se cierra el *popup* abierto en caso de haber uno. Si no hay ninguno abierto, se abrirá el menú de pausa.

Mientras haya abierto algún *popup* el personaje no puede moverse, incluyendo el menú de pausa. Esto se consigue teniendo un booleano "InputActivado" que se comprueba antes de hacer el movimiento del personaje.

El menú de pausa tiene dos botones: "Continuar Jugando" y "Salir al Menu".

En caso de apretar el botón de "Continuar Jugando", se cierra el menú de pausa y se sigue la partida.

Si se aprieta el botón de "Salir al Menu", se cargará la escena del menú principal y una vez activada, se cerrará la del juego. Destacar que no hace falta especificar en el .Json que hay

una partida en curso, ya que eso ha ocurrido en el momento en que se ha apretado a “Empezar Partida”.

En la figura 5.66 se puede observar el *popup* del menú de pausa.

Figura 5.66: *Popup del menú Pausa*

5.12. Debugger

Para poder hacer pruebas y ver que todo funciona bien, se ha implementado un modo administrador o un “Debugger”.

El Debugger actúa como si fuese una consola de comandos o una terminal, el usuario puede insertar comandos y estos se activan en consecuencia.

Hay una línea de texto que informa de si el comando se ha ejecutado correctamente o ha habido algún error por parte del usuario, como haber escrito mal el comando.

En la figura 4.67 se puede observar el *popup* del Debugger.

Figura 5.67: *Popup del Debugger*

Para instanciar un comando, simplemente se escribe y se aprieta *Intro*.

Cuando se aprieta *Intro*, el *script* del debugger recoge la *string* del comando y la separa por espacios. Luego utilizando un *Switch* se comprueba la primera palabra y si es un comando lo ejecuta. Si el comando necesita dos o más palabras, primero se comprueba la primera palabra (para saber que comando es) y luego la segunda.

A continuación, se listan todos los posibles comandos, junto con imágenes de todas las posibles respuestas de esta consola de comandos:

Lo que se encuentre entre los simbolos “<” y “>” indica que se espera una variable, por ejemplo, <número> indica que el comando espera un número.

- **ADDRANDOMCARD <número>**: Añade un número determinado de cartas aleatorias de esbirro al mazo de Esbirros.

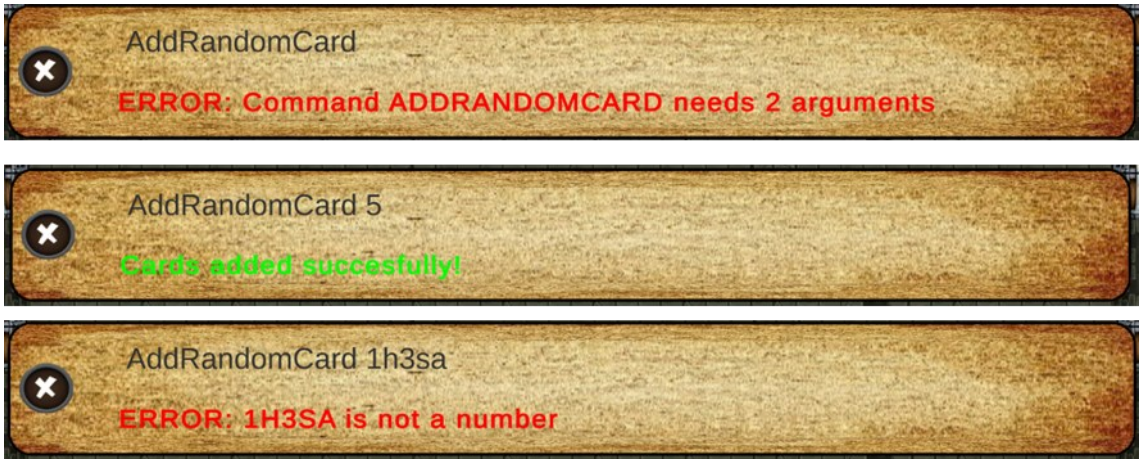

Figura 5.68: *Comando AddRandomCard*

- **ADDRANDOMARTIFACT <número>**: Añade un número determinado de artefactos aleatorios a la bolsa de Artefactos.

Figura 5.69: *Comando AddRandomArtifact*

- **ADDGOLD < número >**: Añade el número indicado de oro al inventario, puede ser negativo para restar oro.

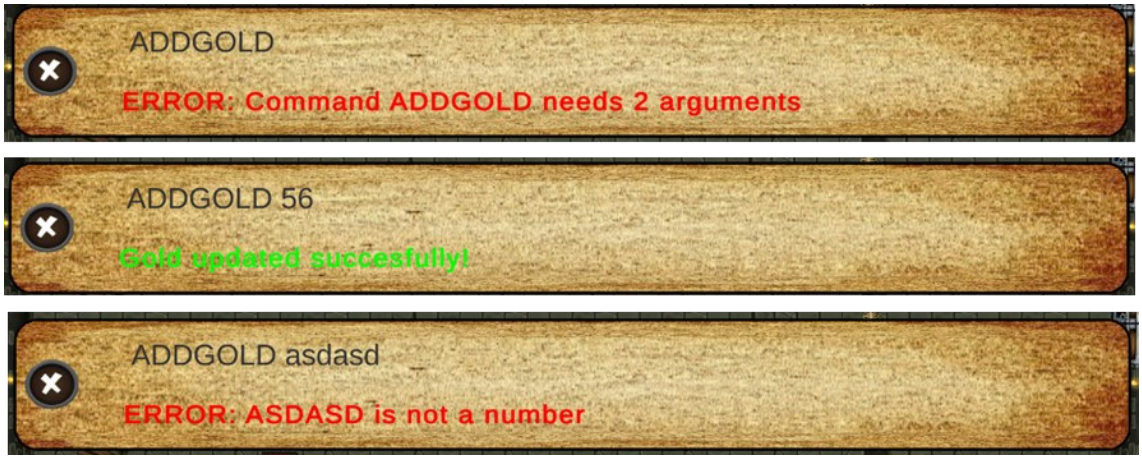

Figura 5.70: *Comando AddGold*

- **SPAWN <nombre>**: Crea encima del personaje el objeto determinado por la *string* (el nombre del objeto a crear).

Figura 5.71: *Comando Spawn*

Estos serán los objetos a *spawnear*:

- **ArtifactPickup**: *Pickup* con Artefactos a elegir para añadir a la bolsa de artefactos.
- **CardPickup**: *Pickup* con Esbirros a elegir para añadir al mazo de esbirros.
- **GSmallPickup:** *Pickup* de oro para recoger, con un valor total de 5 de oro.
- **GMidPickup:** *Pickup* de oro para recoger, con un valor total de 10 de oro.
- **GBigPickup:** *Pickup* de oro para recoger, con un valor total de 20 de oro.
- **GSmallJewelPickup**: *Pickup* de oro para recoger, con un valor total de 35d e oro.
- **GMidJewelPickup**: *Pickup* de oro para recoger, con un valor total de 50 de oro.
- **GBigJewelPickup**: *Pickup* de oro para recoger, con un valor total de 100 de oro.
- **ArtifactSellable**: Carta de artefacto que puedes ver para elegir si comprarla o no.
- **CardSellable**: Carta de Esbirro que puedes ver para elegir si comprarla o no
- **SellChooseArtifactPickUp**: ArtifactPickup pero con precio pensado para que se encuentre en la tienda.
- **SellChooseCardPickUp**: CardPickup pero con precio pensado para que se encuentre en la tienda.

- **OPENALLDOORS**: Abre todas las puertas del piso en el que se encuentra el personaje.

Figura 5. 72: *Comando OpenAllDoors*

- **CLOSEALLDOORS**: Abre todas las puertas del piso en el que se encuentra el personaje.

Figura 5. 73: *Comando CloseAllDoors*

- **CLEARMAP**: Borra el minimapa dejándolo vacío. Está pensado para comprobar que funcionan bien las actualizaciones del minimapa cuando se cambia de piso.

Figura 5.74: *Comando ClearMap*

- **HEAL <número>**: Cura el número determinado de vida al jugador.

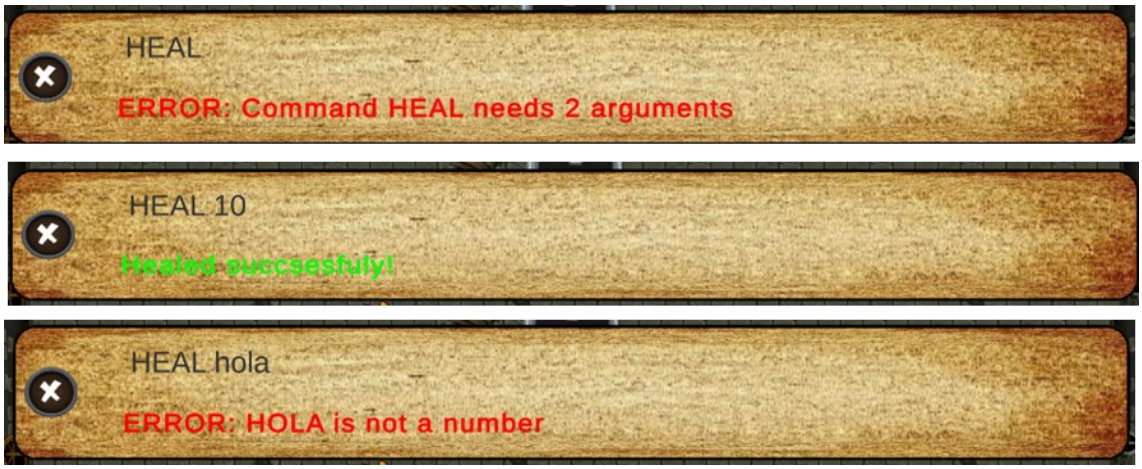

Figura 5.75: *Comando Heal*

- **DAMAGE <número>**: te hace el número indicado de puntos de daño.

Figura 5.76: *Comando Damage*

- **ADDMAXHEALTH <número>**: Añade el número indicado de puntos de vida máxima, puede ser negativo. La vida máxima está limitada a 99 y no puede ser menor de 5.

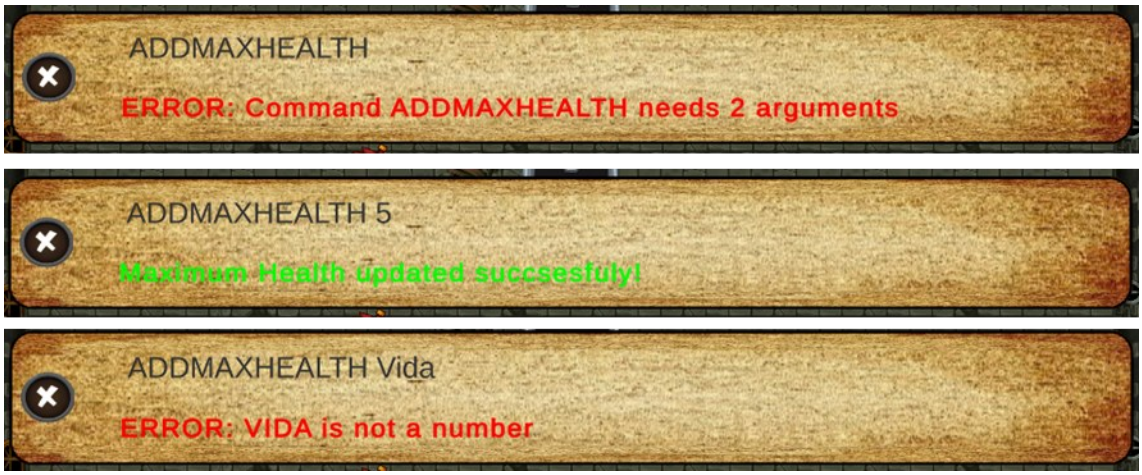

Figura 5.77: *Comando AddMaxHealth*

6. Implantación

Respecto a la implantación del Juego, una de las ventajas que tiene utilizar un motor de juego como Unity es que el propio motor te genera las *Builds* con las que ejecutar el juego.

Por lo que simplemente accediendo al apartado de *builds* en el propio Unity y preparando las escenas como las necesites, se crea una *Build* jugable.

7. Pruebas

Las pruebas de las diferentes funcionalidades se iban haciendo a medida que se iban creando estas funcionalidades. No se solía empezar una nueva funcionalidad hasta que no se hubiese comprobado que estuviese correctamente implementada en el juego y no diese ningún tipo de error.

Para probar estas funcionalidades, una herramienta que se ha utilizado es el propio Editor de Escenas e Inspector de Unity (que es muy útil para comprobar todas las situaciones).

Además, también se ha implementado un "Debugger" que actúa como consola de comandos para poder hacer todo tipo de pruebas. Esta herramienta ha sido muy útil ya que facilitaba el hacer pruebas dentro del mismo juego sin tener que ir cambiando de ventanas entre editor de escenas y juego.

El diseño de este debugger se explica en el apartado 4.1.11 de este mismo documento. La implementación del debugger se explica en el apartado 5.12 con más detalle.

8. Conclusiones

Finalmente, se ha conseguido obtener un resultado final jugable, que era el objetivo principal del proyecto.

El objetivo del presente Trabajo Final de Grado era el desarrollo de un videojuego de tipo *roguelike* centrado en la gestión de recursos y cartas. Este objetivo finalmente se ha logrado ya se ha obtenido una versión jugable del videojuego.

Aunque el proyecto no esté completo, ya que falta el módulo de combate que tiene que hacer Pau Vidal Ramón en su TFG, el proyecto tiene todo el resto de elementos del videojuego preparados para ser aplicados al módulo de combate cuando haga falta (soltar las recompensas de combate o hacer daño al jugador).

Cabe destacar que en el diseño del juego se ha prestado especial importancia a la facilidad de personalización y ampliación del mismo. Esto se ha conseguido mediante la creación de herramientas que facilitan la ampliación de contenido en caso de ser necesario. Por ejemplo, se pueden crear nuevos pisos con simplemente copiar un archivo y cambiar unas variables para elegir las características del piso. Lo mismo ocurre con las cartas de esbirro o las cartas de artefacto, con solo copiar un archivo "base", elegir una imagen para la carta y elegir su raza y su clase e inicializar variables como el nombre de la carta y la descripción, ya se ha creado una carta nueva. En el caso de las cartas de artefacto no habría que elegir raza y clase.

8.1. Relación del trabajo desarrollado con los estudios cursados

Evidentemente, en la creación de un videojuego se aplican conocimientos que se han visto a lo largo de toda la carrera. Los más significativos o que más pueden resaltar son:

- **Programación Orientada a Objetos**: Al final en un videojuego hay muchos objetos que deben tener acciones e interacciones con otros, por lo que las asignaturas que te enseñan esta programación orientada a objetos son importantes a la hora de hacer videojuegos.
- **Concurrencia**: En un videojuego hay muchos elementos que hacen sus acciones al mismo tiempo, por lo que todas las asignaturas que te enseñan concurrencia y manejar diferentes *threads* son importantes.
- **Interfaces Gráficas**: A lo largo de la carrera se han cursado varias asignaturas enfocadas a interfaces gráficas y manejar elementos gráfica y espacialmente, Evidentemente, esta área del conocimiento es muy importante en un videojuego.

8.2. Reflexiones

En este apartado voy a hablar con un tono más personal sobre lo que ha supuesto este proyecto para mí.

En cuanto a los conocimientos, gracias a este proyecto he consolidado muchos aspectos de la informática que he estudiado en las diferentes clases durante estos cuatro años de carrera.

Respecto a la forma de trabajar, este proyecto me ha ayudado a trabajar mejor en equipo sobre todo a la hora de decidir las ideas y plantear el videojuego, ya que tuvimos que llegar a un acuerdo mi compañero Pau Vidal y yo sobre cómo debería de ser el videojuego.

Finalmente, decir que estoy muy satisfecho con este proyecto y he disfrutado mucho llevándolo a cabo porque gracias a este he podido aprender y programar sobre géneros de los videojuegos que me encantan y de los que me considero apasionado, concretamente los videojuegos *roguelike* y los videojuegos de cartas.

9. Trabajos futuros

En caso de querer sacar este videojuego al mercado, se podrían hacer muchas ampliaciones. Algunas de ellas son:

- **Integrar el módulo de combate del TFG de mi compañero Pau Vidal Ramón:** Evidentemente, el juego sin este módulo de combate está incompleto, por lo que implementarlo es completamente necesario antes de salir al mercado.
- **Ampliar el catálogo de cartas:** Uno de los aspectos más importantes del género *Roguelike* es la rejugabilidad. Para eso es muy importante que haya una gran variedad de cartas para que no todas las partidas se sientan iguales o parecidas. Afortunadamente, como ya ha sido explicado, el juego ya está preparado para que la creación de cartas sea muy fácil y rápida.
- **Cambiar el apartado gráfico del juego:** En caso de querer profesionalizar aún más este videojuego, lo más recomendable sería contratar a algún artista para que cambie el apartado artístico. Hay que tener en cuenta que este proyecto ha sido hecho por una única persona que se ha centrado en la parte técnica y de la programación.
- **Añadir nuevos pisos:** Las partidas de este juego ahora mismo consisten en superar un total de tres pisos. Sin embargo, el juego también está preparado para que en caso de diseñar nuevos pisos, sea muy fácil su implementación.
- **Añadir nuevos personajes jugables:** Otra característica común de muchos *Roguelikes*, es que antes de empezar la partida te dejan elegir entre una serie de personajes diferentes. Está misma idea se podría implementar en este juego, cambiando así con qué cartas empiezas. Por ejemplo, si empiezas con un personaje que sea un elfo podrías empezar con 6 cartas de esbirro que sean elfos (aunque luego durante la partida acabes añadiendo cartas de todas las razas).

10. Referencias bibliográficas

[1] (S/f-a). Org.es. Recuperado el 30 de junio de 2023, de http://www.aevi.org.es/web/wp-content/uploads/2023/05/Anuario-AEVI-2022.pdf

[2] Vera, J. A. C. (2015). La dimensión social de los videojuegos "online": de las comunidades de jugadores a los "e-sports". Index.comunicación: Revista científica en el ámbito de la Comunicación Aplicada, 5(1), 39–51. https://dialnet.unirioja.es/servlet/articulo?codigo=5277303

[3] Rojas, J. (2022, junio 2). El impacto de la tecnología en el desarrollo de videojuegos. Telefónica. https://www.telefonica.com/es/sala-comunicacion/blog/tecnologia-en-desarrollo-de-videojuegos/

[4] Abadíe, B. (2021, agosto 1). Rogue-like, el género indie de moda que ha dado el salto a la industria mainstream. GQ España. https://www.revistagq.com/noticias/articulo/rogue-like-mejores-juegos-indie

[5] de 3DJuegos, E. E. (2020, agosto 14). 9 Juegos de cartas para un jugador para obsesionarse como con Slay the Spire. 3djuegos.com; 3DJuegos. https://www.3djuegos.com/juegos/slay-the-spire/noticias/9-juegos-de-cartas-para-un-jugador-para-obsesionarse-como-con-slay-the-spire-200814-1976

[6] Campana, N. (2022, marzo 23). Metodologías ágiles de desarrollo de software [Guía Completa]. Freelancer Blog; freelancermap.com. https://www.freelancermap.com/blog/es/metodologias-agiles-desarrollo-software/

[7] Metodología de desarrollo basado en funciones (FDD). (s/f). Tecnologias-informacion.com. Recuperado el 30 de junio de 2023, de https://www.tecnologias-informacion.com/metodologia-funciones.html

[8] *Historia de los juegos de cartas desde sus orígenes hasta hoy*. (s/f). Dupalu.com. Recuperado el 12 de junio de 2023, de https://www.dupalu.com/2021/10/historia-de-los-juegos-de-cartas-desde.html

[9] Euroinnova Business School. (2023, mayo 31). juegos de cartas. Euroinnova Business School. https://www.euroinnova.edu.es/juegos-de-cartas

[10] Morales, R. (2021, julio 8). Los mejores juegos de cartas para PC. Alfa Beta Juega. https://www.mundodeportivo.com/alfabeta/listas/mejores-juegos-cartas-pc

[11] Alex, C. D. (2019, julio 1).*Qué es el Auto Chess y por qué se está convirtiendo en el nuevo género de moda*. Vidaextra.com; Vida Extra. https://www.vidaextra.com/estrategia/que-auto-chess-que-se-esta-convirtiendo-nuevo-genero-moda

[12] Leonetix. (2018, septiembre 20). *[Artículo] Historia del roguelike: Nacimiento de lo aleatorio*. GaminGuardian. https://gaminguardian.com/articulo-historia-del-roguelike-nacimiento-de-lo-aleatorio/

[13] *Berlin interpretation*. (s/f). Roguebasin.com. Recuperado el 25 de junio de 2023, de https://www.roguebasin.com/index.php?title=Berlin_Interpretation

[14] Leonetix. (2018, septiembre 20). [Artículo] Historia del roguelike: Nacimiento de lo aleatorio. GaminGuardian. https://gaminguardian.com/articulo-historia-del-roguelike-nacimiento-de-lo-aleatorio/

[15] Roguelike. (s/f). Academia-lab.com. Recuperado el 30 de junio de 2023, de https://academia-lab.com/enciclopedia/roguelike/

[16] (S/f-b). Archive.ph. Recuperado el 30 de junio de 2023, de https://archive.ph/sx9EW

[17] (S/f-c). Recuperado el 30 de junio de 2023, de http://]https://www.destinorpg.es/2020/03/la-interpretacion-de-berlin-2008-el.html

[18] ¿Qué es un roguelike?¿Y un roguelite? (2020, septiembre 28). JuegosADN. https://juegosadn.es/que-es-un-rogue-lite-y-un-rogue-like-ar-3790/

ANEXO 1

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**	

El 25 de septiembre de 2015, los líderes internacionales acordaron un conjunto de metas globales con el propósito de eliminar la pobreza, preservar el medio ambiente y garantizar el bienestar para todos, como parte de una nueva agenda de desarrollo sostenible. Cada uno de estos objetivos establece metas concretas que deben cumplirse en los próximos 15 años.

Aunque haya videojuegos que sí que podrían aportas a alguno de estos objetivos, ya sea porque son educativos o porque tratan estos temas de forma directa, este proyecto no trata de ninguna forma alguno de los temas de los ODS.