Universitat Oberta de Catalunya

Desarrollo de un videojuego 2D: The Dungeon Rules

Autor: Iván Rodríguez Moraleja

Tutor: Jordi Duch Gavaldà

Profesor: Joan Arnedo Moreno

Diseño y Programación de Videojuegos

Diseño de experiencias de juego

05/06/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo de un videojuego 2D: The Dungeon Rules*
Nombre del autor:	*Iván Rodríguez Moraleja*
Nombre del colaborador/a docente :	*Jordi Duch Gavaldà*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	*06/2022*
Titulación o programa:	*Diseño y programación de videojuegos*
Área del Trabajo Final:	*Diseño de experiencias de juego*
Idioma del trabajo:	*Castellano*
Palabras clave:	*Videojuego, dungeon, puzle, sokoban, cartas*

Resumen:

El presente trabajo tiene como objetivo el desarrollo de un videojuego 2D llamado The Dungeon Rules.

The Dungeon Rules es un juego casual que tiene como principal característica la combinación de dos subgéneros muy conocidos en el mundo de los videojuegos: la exploración de mazmorras y los puzles de tipo Sokoban.

Cada nivel del juego es una mazmorra. Cada mazmorra ocupa una pantalla. El jugador comienza en la entrada de la mazmorra, y para superarla debe llegar a la salida. La salida le lleva a la siguiente mazmorra.

Para abrirse camino, el jugador tendrá que mover obstáculos de forma táctica, activar mecanismos y desactivar trampas. El movimiento se lleva a cabo por casillas y está regulado por turnos. Las mazmorras están plagadas de enemigos, a los que se deberá enfrentar mediante combates con cartas. Pero las mazmorras también contienen tesoros, ítems que ayudarán al jugador a salir victorioso.

Las principales herramientas utilizadas para llevar a cabo el trabajo son: Unity (motor de juego con editor gráfico), Photoshop (edición de imagen), Reaper (edición de sonido) y Github (control de versiones).

Se utilizan assets gráficos y assets de sonido de terceros, que solo necesitan ser editados según las necesidades del proyecto.

El resultado final del trabajo es la obtención de un MVP (producto mínimo viable) realizado en un plazo de 15 semanas, que puede ser ejecutado sobre Microsoft Windows.

Abstract:

The aim of this work is the development of a 2D videogame called The Dungeon Rules.

The Dungeon Rules is a casual game whose main feature is the combination of two subgenres well known in the world of video games: dungeon crawler and Sokoban type puzzles.

Each level of the game is a dungeon. Each dungeon occupies one screen. The player starts at the entrance of the dungeon, and to overcome it he must reach the exit. The exit leads to the next dungeon.

To make his way through, the player will have to tactically move obstacles, activate mechanisms and deactivate traps. The movement is carried out by squares and is turn-based. The dungeons are full of enemies, which must be confronted by cards combat. But the dungeons also contain treasures, items that will help the player to emerge victorious.

The main tools used to carry out the work are: Unity (game engine with graphic editor), Photoshop (image editor), Reaper (audio editor) and Github (version control).

Third party graphic assets and sound assets are used, which only need to be edited according to the needs of the project.

The final result of the work is an MVP (minimum viable product), made in 15 weeks, which can be executed on Microsoft Windows.

Dedicatoria

A Camino, la princesa que me rescató de la mazmorra.

A Abril y Olivia, las hadas que me tienen hechizado.

Agradecimientos

Quiero dar las gracias a Jordi Duch Gavaldà, tutor de este proyecto, por sus sugerencias y comentarios, y por los ánimos que me ha profesado durante la realización del trabajo.

También a Joan Arnedo Moreno, y a todas las profesoras y profesores de la UOC que me han impartido docencia en este master, y de los que he recibido muy buenos conocimientos y mucha dedicación.

Ha sido un placer haber sido alumno de todos ellos.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción

1.1. Introducción

The Dungeon Rules es un videojuego 2D que combina dos subgéneros que tienen una larga trayectoria en la historia de los videojuegos: la exploración de mazmorras y el puzle.

Existen muchos juegos de exploración de mazmorras que pueden incluir la resolución de puzles en algún momento del juego, sin embargo no es una de las mecánicas principales. En The Dungeon Rules la componente de puzle cobra la misma importancia que la propia exploración.

Los juegos de exploración de mazmorras aparecen ya en los inicios del videojuego. Uno de los pioneros del género fue Rogue, desarrollado por Michael Toy y Glenn Wichman en 1980, y distribuido por Epyx años después. El éxito de este juego da lugar a multitud de variaciones y clones como Hack (Jay Fenlason, 1982), Moria (Robert Koeneke, 1983) o Angband (Alex Cutler y Andy Astrand, 1990).

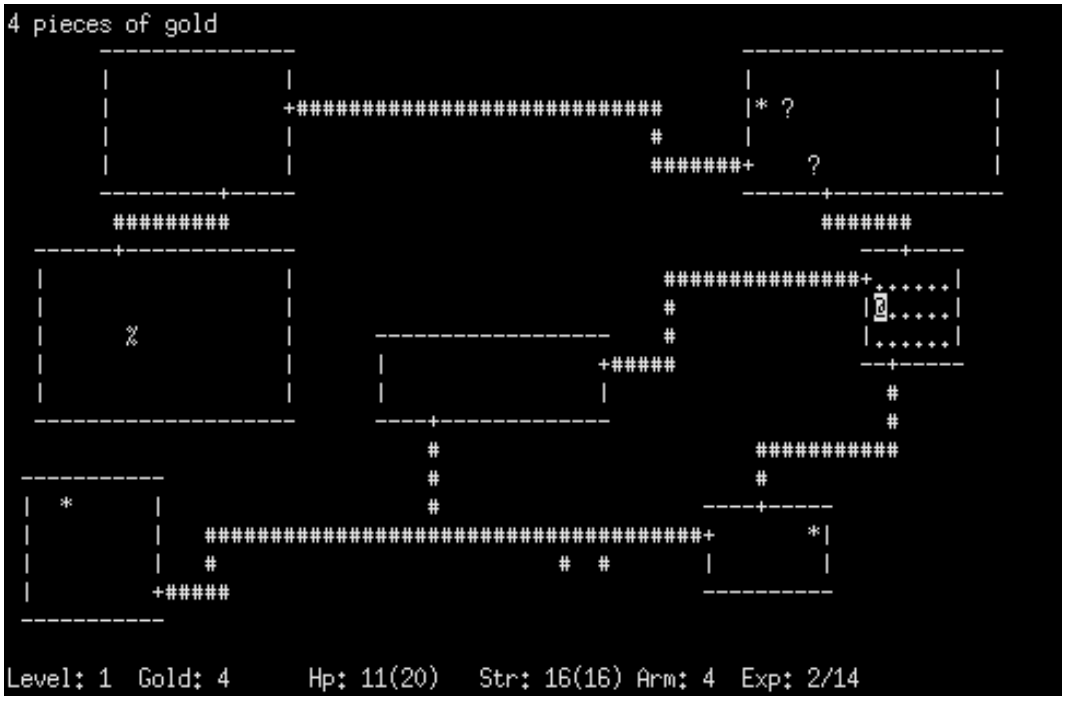

Figura 1: Rogue. Fuente: https://en.wikipedia.org.

Todos estos juegos terminan creando un género bautizado como roguelike. Esta etiqueta hace referencia a los juegos de tipo Rogue. El término roguelike engloba actualmente a juegos que tienen una serie de mecánicas concretas, pero que pueden diferir enormemente entre ellos.

Por otro lado, los puzles de The Dungeon Rules son principalmente de tipo Sokoban. Éste juego fue creado por Hiroyuki Imabayashi en 1981. La primera versión comercial es publicada en 1982 por Thinking Rabbits, para el ordenador NEC PC-8801. Desde entonces se han publicado, y se siguen publicando, multitud de juegos con la misma mecánica que Sokoban o con variaciones de todo tipo.

El juego consiste en empujar cajas dentro de un laberinto para colocarlas en los puntos especificados. La dificultad se encuentra en evitar que las cajas queden bloqueadas de forma que no se puedan mover hacia su destino.

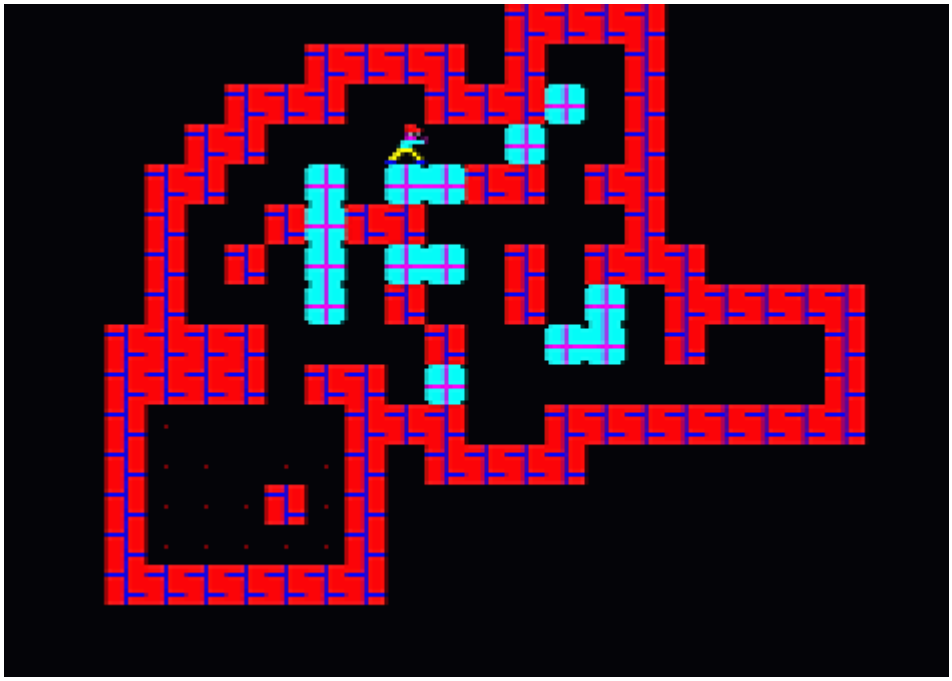

Figura 2: Sokoban. Fuente: https://www.uvlist.net

En el caso de The Dungeon Rules, se utiliza la mecánica básica de empujar objetos para abrirse camino, desactivar trampas, bloquear enemigos u otros fines que permitan avanzar en el juego.

El resultado de combinar estos géneros es un juego casual en el que se aprovechan las características de ambos géneros para motivar al jugador:

- Por un lado se puede disfrutar de la búsqueda de aventura, el descubrimiento y el enfrentamiento de peligros que proporciona la exploración de mazmorras.
- Y por otro se puede obtener la satisfacción que supone resolver un puzle complejo.

1.2. Descripción/Definición

The Dungeon Rules es un videojuego 2D que combina la exploración de mazmorras con la resolución de puzles.

Las mecánicas principales del juego son las siguientes:

a) Exploración de mazmorras: cada mazmorra ocupa una pantalla.
   En cada mazmorra el jugador comienza en la puerta de entrada a la estancia y tiene que alcanzar la puerta de salida para acceder a la siguiente mazmorra.
   Dentro de la mazmorra el jugador encontrará tesoros, trampas y enemigos.
   El movimiento se realiza por casillas y está regulado por turnos.
b) Resolución de puzles: para llegar a la salida de la mazmorra el jugador deberá solucionar puzles, especialmente de tipo Sokoban (Thinking Rabbit, 1982).
   El jugador se encontrará con obstáculos que deberá mover de manera táctica para abrirse camino sin quedar bloqueado. Del mismo modo, deberá accionar palancas y otros mecanismos para abrir puertas y desactivar trampas.
c) Para resolver el enfrentamiento contra los enemigos se utilizará un sistema de combate por cartas.
   En un principio se pensó en un combate de acción por turnos, pero esto suponía dotar al juego de mecánicas RPG con atributos o habilidades escalables.
   Debido a que no se deseaba que este aspecto se convirtiera en la mecánica central del juego, se implementa un sistema de combate por cartas muy sencillo.
d) Del mismo modo, los tesoros o recompensas que obtiene el personaje son cartas de magia, que puede utilizar durante la exploración y que le ayudarán a resolver los puzles para llegar a la salida.
   Por lo tanto hay dos tipos de cartas: cartas de magia, que solo se pueden usar durante la exploración, y cartas de combate que se pueden usar durante el combate.

Otros aspectos fundamentales del juego son:

- El jugador puede reiniciar cuando quiera la mazmorra en la que se encuentra, porque puede llegar a una situación de bloqueo, en la que el puzle no tenga solución.
- La muerte no es permanente. Si el jugador muere en combate se reinicia la mazmorra en la que se encuentra.

El resultado es un juego casual, donde cada mazmorra es un puzle único e independiente del resto, diseñado manualmente. La continuidad entre los distintos puzles viene dada por la mecánica de exploración, que permite ir acumulando objetos y habilidades en forma de cartas, y que pueden ser utilizados en el momento que el jugador estime oportuno.

1.3. Objetivos generales

El objetivo general de este trabajo es el de desarrollar un producto mínimo viable (MVP).
El resultado final que se persigue es obtener una aplicación totalmente funcional que consiste en un videojuego 2D que puede ejecutarse en un PC sobre el sistema operativo Microsoft Windows.

1.3.1. Objetivos principales

El videojuego presenta un acabado que incluye la implementación de un núcleo básico que permita, en un futuro, una escalabilidad muy sencilla que consiste en añadir nuevos niveles y nuevos ítems (en este caso en forma de nuevas cartas).

Las características que se implementan son:

- Menú principal con estas opciones: nueva partida, continuar partida, configuración, créditos y salir.
- Sistema de persistencia de datos para continuar la última partida o reiniciar un nivel.
- Tablero de juego estructurado en casillas con zona caminable y zona no caminable.
- Gestión de turnos de los actores: player y enemigos.
- Movimiento del player.
- IA que controla el movimiento y el combate de los enemigos.
- Gestión de un inventario de cartas.
- Sistema de combate: acciones que puede realizar el player durante el combate en función de las cartas que posee.

- Sistema de magia: acciones que puede realizar el player durante la exploración en función de las cartas que posee.
- HUD de combate y HUD de exploración.
- Edición de imágenes y elaboración de animaciones básicas partiendo de assets de terceros.
- Diseño sonoro de las acciones principales.

Una vez implementados estos puntos se diseñan 12 niveles.

1.3.2. Objetivos secundarios

Como objetivo secundario se establece la implementación de las siguientes características, cuyo desarrollo depende del tiempo disponible y del buen progreso de los objetivos principales:

- Tutorial formado por varios niveles.
- Narrativa: elaboración de una secuencia inicial animada.
- Iluminación del entorno.
- Animación de un mayor número de elementos.
- Diseño sonoro de un mayor número de elementos.
- Contenido musical.

1.4. Metodología y proceso de trabajo

La estrategia llevada a cabo para realizar el trabajo es la de adaptar características y mecánicas de diferentes juegos y combinarlas para crear un juego nuevo.
Esto requiere del análisis de los juegos de los que se extraen las mecánicas a combinar.

La siguiente tabla muestra los principales juegos que se han analizado y las características que se han tomado.

Juego	**Mecánicas adoptadas**
Portal (Valve, 2007)	Estancias independientes. El jugador debe alcanzar la salida para progresar a la siguiente estancia. La estancia se reinicia si el jugador muere.

	El jugador puede reiniciar la estancia si queda bloqueado.
Stoneshard (Ink Stains Games, 2020)	Exploración de mazmorras. Descubrimiento de tesoros. Encuentro con enemigos. Movimiento por turnos y basado en casillas.
Sokoban (Thinking Rabbit, 1982)	Movimiento de obstáculos de forma táctica para abrir el camino y no quedar bloqueado.
Slay the Spire (Mega Crit Games, 2019)	Combate con cartas

Tabla 1: Juegos analizados.

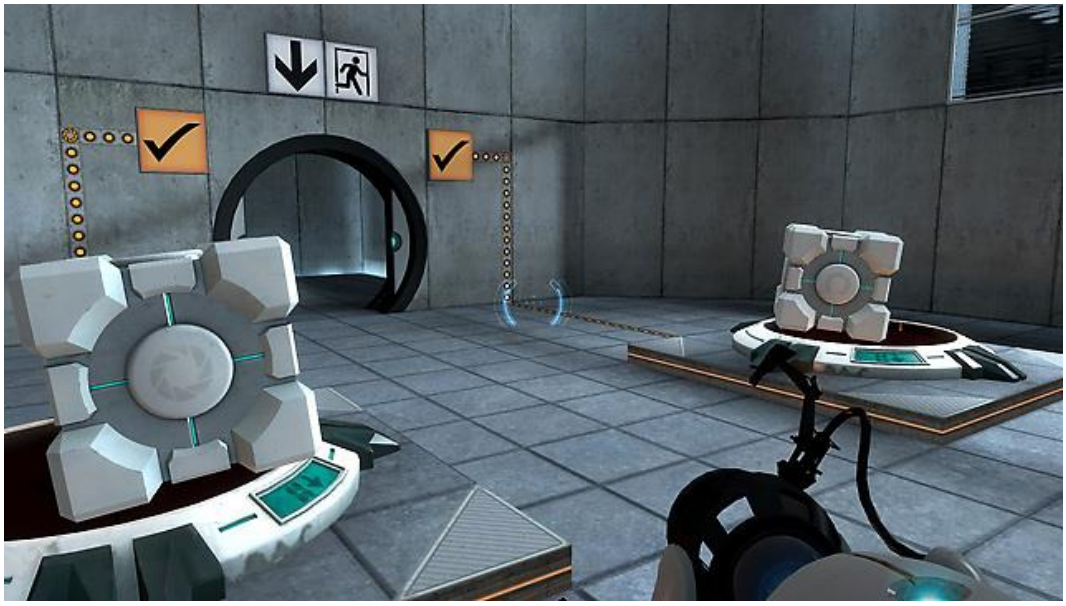

Figura 3: Portal (Valve). Fuente: https://store.steampowered.com.

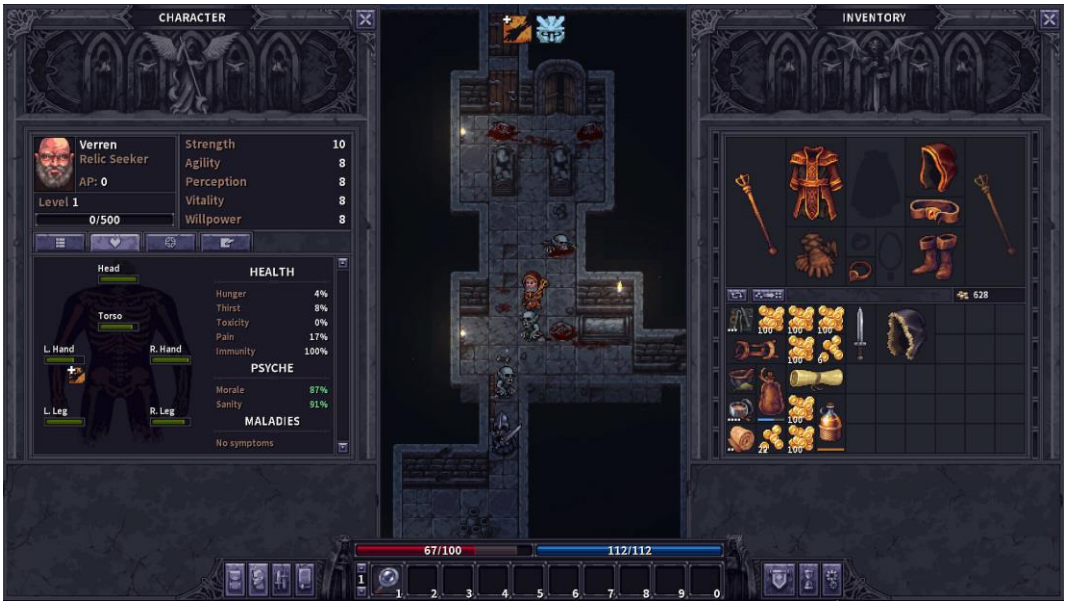

Figura 4: Stoneshard (Ink Stains Games). Fuente: https://store.steampowered.com.

Figura 5: Slay the Spire (Mega Crit Games). Fuente: https://store.steampowered.com.

Se ha utilizado una metodología incremental para el desarrollo software, de modo que el proyecto se divide en funcionalidades que se implementan y agregan progresivamente.

Las principales funcionalidades del núcleo jugable son:

- Movimiento del player sobre el tablero.
- Obstáculos y trampas.

- Gestión de turnos y movimiento de enemigos.
- Inventario de cartas.
- Sistema de combate.
- Sistema de magia.
- Progreso de niveles y persistencia de datos.
- Menús.

La implementación de cada una de estas funcionalidades incluye:

1) Análisis.
2) Diseño.
3) Programación.
4) Test.

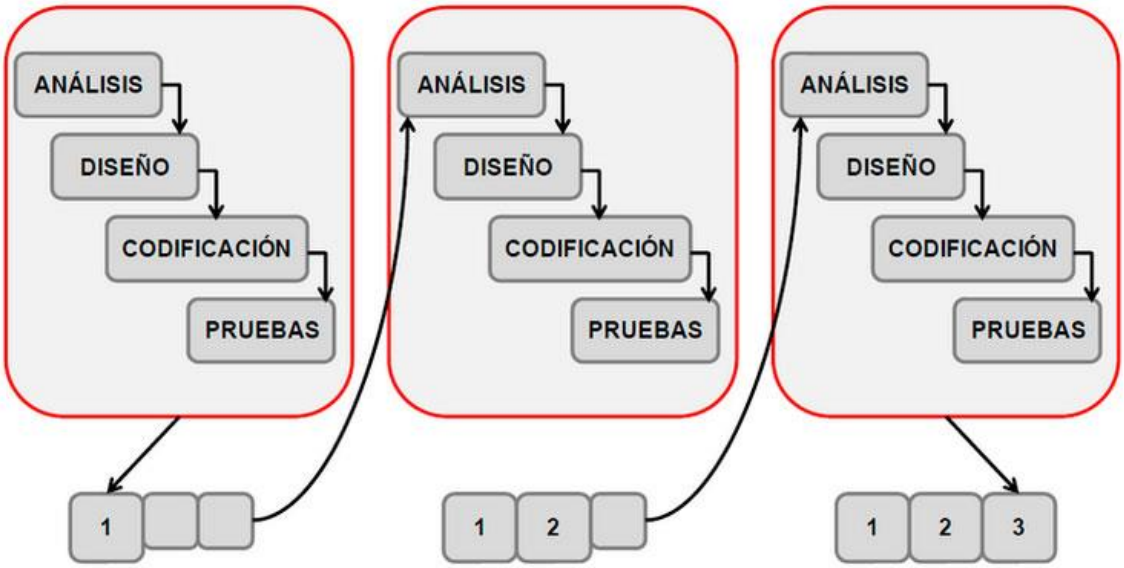

Figura 6: Modelo incremental de desarrollo software. Fuente: https://efectodigital.online.

1.5. Planificación

La planificación del proyecto está dividida en 4 fases. Cada una de estas fases se corresponde con cada PEC de la asignatura.

Las tareas que se han listado en esta planificación coinciden con las funcionalidades principales descritas en el apartado anterior. Y como se especificó en dicho apartado, cada tarea incluye su propio análisis, diseño, implementación y pruebas.

Se observan dos hechos:

- Edición de assets. Los assets se van editando según se van necesitando en la implementación. Este es uno de los pocos casos en los que se va a trabajar en paralelo.
- Entrega de la documentación. Esta tarea se realiza la última semana de cada fase. Esa semana se dedica también a ultimar detalles de esa fase.

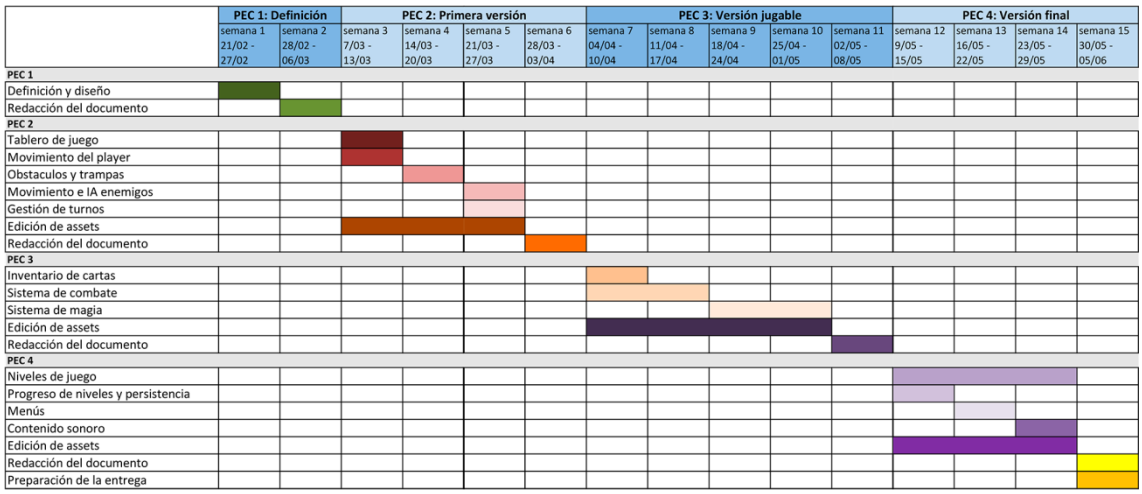

Figura 7: Diagrama de Gantt del proyecto.

	PEC 1: Definición		PEC 2: Primera versión			
	semana 1 21/02 - 27/02	semana 2 28/02 - 06/03	semana 3 7/03 - 13/03	semana 4 14/03 - 20/03	semana 5 21/03 - 27/03	semana 6 28/03 - 03/04
PEC 1						
Definición y diseño						
Redacción del documento						
PEC 2						
Tablero de juego						
Movimiento del player						
Obstaculos y trampas						
Movimiento e IA enemigos						
Gestión de turnos						
Edición de assets						
Redacción del documento						
PEC 3						
Inventario de cartas						
Sistema de combate						
Sistema de magia						
Edición de assets						
Redacción del documento						
PEC 4						
Niveles de juego						
Progreso de niveles y persistencia						
Menús						
Contenido sonoro						
Edición de assets						
Redacción del documento						
Preparación de la entrega						

Figura 8: Detalle del diagrama de Gantt del proyecto: PEC1 y PEC2.

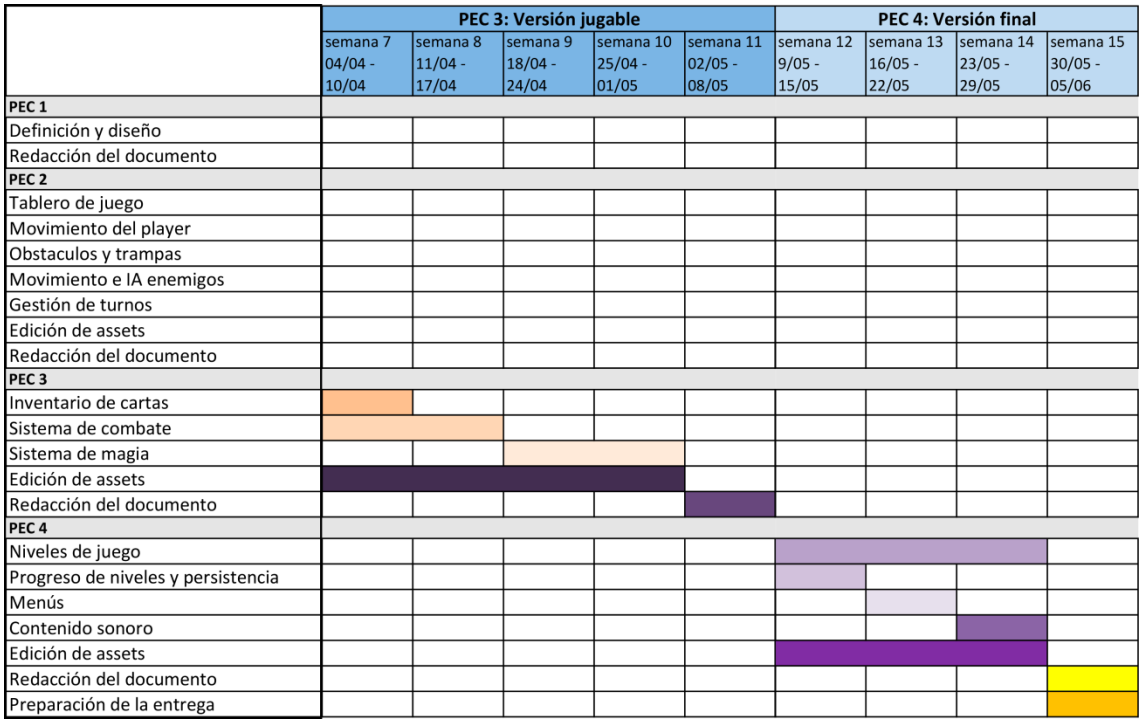

Figura 9: Detalle del diagrama de Gantt del proyecto: PEC3 y PEC4.

1.6. Presupuesto

El presupuesto refleja el desarrollo del producto que se ha llevado a cabo, realizado por una sola persona y en el tiempo real de ejecución.

De cara a la adquisición de determinado software, como Unity o Reaper, se considera que no hubo ingresos en los últimos 12 meses.

RECURSOS HUMANOS			
Concepto	**Precio (€) / Tiempo**	**Tiempo**	**Total (€)**
Desarrollador	2500 / mes	4 meses	10000
RECURSOS TÉCNICOS			
Software			
Concepto	**Precio (€) / Tiempo**	**Tiempo**	**Total (€)**
Unity	0	4 meses	0
Photoshop	24,19 / mes	4 meses	96,76
Reaper	60	4 meses	60
DaVinci Resolve	0	4 meses	0
Libre Office	0	4 meses	0
HARDWARE			
Concepto	**Precio (€) / Unidad**	**Unidades**	**Total (€)**
PC (con S.O. y periféricos)	1000	1	1000
Monitor de vídeo	175	2	350
Interfaz de audio	60	1	60
Monitor de audio	100	2	200
ASSETS			
Concepto	**Precio (€) / Unidad**	**Unidades**	**Total (€)**
Assets gráficos: Dungeon Tileset	15	1	18,15
Assets gráficos: Fantasy Heroes Editor	32,42	1	32,42
Assets gráficos: Undead Heroes Editor	16,20	1	16,20
Efectos sonoros: Soundly	14,99 / mes	1 mes	14,99
INMUEBLES			
Concepto	**Precio (€) / Tiempo**	**Tiempo**	**Total (€)**
Alquiler oficina	500 / mes	4 meses	2000
Gastos: comunidad, agua, luz, calefacción, conexión a internet.	200 / mes	4 meses	800
		TOTAL (€)	**14.648,52**

Tabla 2: Presupuesto del proyecto.

1.7. Estructura del resto del documento

Los contenidos de los siguientes capítulos son:

- **Capítulo 2: Análisis de mercado**. En este capítulo se hace un análisis del mercado en el que se enmarca el producto desarrollado. Se analiza la audiencia potencial, la competencia y se hace un análisis DAFO.
- **Capítulo 3: Propuesta**. En este capítulo se detallan las características del producto desarrollado y se describe la estrategia de difusión y promoción del mismo.
- **Capítulo 4: Diseño**. En este capítulo se describe el diseño del juego, haciendo especial hincapié en los componentes de software que se han desarrollado. También se muestran los niveles del juego.
- **Capítulo 5: Implementación**. En este capítulo se especifican las instrucciones de instalación de la aplicación y los requisitos mínimos requeridos.
- **Capítulo 6: Guía de usuario**. Este capítulo es un manual de usuario para hacer uso de la aplicación.
- **Capítulo 7: Conclusiones y líneas de futuro**. En este capítulo se hace una retrospectiva del trabajo realizado y se analizan los resultados obtenidos. También se especifican las características de futuras versiones del producto.

2. Análisis de mercado

2.1. Público objetivo.

Ya se ha comentado que The Dungeon Rules es un juego que combina dos subgéneros muy conocidos: la exploración de mazmorras (dungeon crawler) y los puzles de tipo Sokoban.

En la International Roguelike Development Conference celebrada en Berlín en 2008 se enunciaron un listado de mecánicas asociadas al género roguelike. La lista de mecánicas indica cómo de roguelike es un juego. Un juego que no tenga todas las mecánicas no implica que no pueda considerarse un roguelike. Del mismo modo, un juego que incluya muchas de estas mecánicas no tiene porqué ser de género roguelike. De hecho, también se ha popularizado la etiqueta roguelite para hacer referencia a juegos que tienen muchas mecánicas roguelike, pero que no se consideran como tal.

La siguiente tabla muestra cuales de las características establecidas en la conferencia de Berlín se cumplen en The Dungeon Rules.

MECÁNICA	**SÍ / NO**	**OBSERVACIONES**
Factores de alto valor		
Generación procedural del entorno	NO	
Muerte permanente	NO	
Acciones reguladas por turnos	SÍ	
Movimiento por cuadrículas	SÍ	
No modal	NO	No modal se produce cuando todas las acciones se realizan dentro del mismo modo de juego. En este caso los combates se llevan a cabo en un modo diferente.
Complejidad	SÍ	Diferentes soluciones para resolver el mismo problema.
Gestión de recursos	SÍ	
Hack and slash	SÍ	Esta característica se refiere al hecho de destruir enemigos, no a la mecánica de combate en sí.
Exploración y descubrimiento	SÍ	
Factores de bajo valor		
El jugador maneja un solo personaje	SÍ	

Los enemigos son iguales que el player	SÍ	
Desafío táctico	SÍ	
Gráficos ASCII	NO	
Mazmorras	SÍ	
Cuantificación numérica	NO	Características del personaje cuantificadas con números, algo habitual en los juegos RPG

Tabla 3: Mecánicas roguelike de The Dungeon Rules.

Aunque The Dungeon Rules cumple con la mayoría de características enunciadas en la conferencia de Berlín, no se utilizará la etiqueta roguelike para definirlo. Los motivos son los siguientes:

- El género roguelike engloba juegos con características muy diferentes.
- La comunidad de jugadores asocia principalmente el género roguelike con la generación procedural del mapa y la muerte permanente, dos mecánicas que The Dungeon Rules no tiene.
- El motivo anterior también implica que el género roguelike esté más vinculado con jugadores “hardcore”. Por contra, se pretende fomentar la imagen de The Dungeon Rules como un juego casual de puzzles.

Una vez que se ha enmarcado el juego como un juego casual de puzzles se pasa a determinar el público objetivo. Para ello se hace uso del administrador de anuncios de Meta (https://www.facebook.com/business/tools/ads-manager). Esta herramienta incluye un creador de audiencias que permite estimar cuánta gente, con determinadas características, está interesada en un tema concreto.

Las características de los usuarios que se han utilizado son:

- Lugar de residencia.
- Rango de edad.
- Género.

Se establece como tema de interés la intersección de “videojuego casual” y “puzzles de lógica”. Después de hacer diversas pruebas para residentes en EEUU y Europa, se llega a la conclusión de que el rango de edad que representa el grueso de la audiencia está entre los 15 y los 35 años. Otra conclusión importante es que el resultado es muy similar entre hombres y mujeres.

Tamaño de la audiencia estimada:

434 400 - 511 100

Las estimaciones pueden variar considerablemente a lo largo del tiempo según tus selecciones de segmentación y los datos disponibles.

Características de la audiencia:

- Lugar de residencia:
  - Estados Unidos
  - Europe
- Edad:
  - 15-35
- Sexo:
  - Mujer
- Personas que coinciden con:
  - Intereses: Logic puzzles
- Y deben también ver:
  - Intereses: Videojuego casual

Tamaño de la audiencia estimada:

498 400 - 586 400

Las estimaciones pueden variar considerablemente a lo largo del tiempo según tus selecciones de segmentación y los datos disponibles.

Características de la audiencia:

- Lugar de residencia:
  - Estados Unidos
  - Europe
- Edad:
  - 15-35
- Sexo:
  - Masculino
- Personas que coinciden con:
  - Intereses: Logic puzzles
- Y deben también ver:
  - Intereses: Videojuego casual

Figura 10: Audiencia estimada proporcionada por el administrador de anuncios de Meta.

Se determina entonces, que el público objetivo de The Dungeon Rules son hombres y mujeres de 15 a 35 años.

2.2. Estado del arte

A continuación se examinan juegos de los géneros de exploración de mazmorras y de puzles de tipo Sokoban, y que son relativamente actuales.

También se añaden algunas notas sobre un tercer tipo de juegos, los que utilizan la mecánica de combate con cartas. El motivo es la proliferación de juegos que utilizan esta mecánica en los últimos años.

2.2.1. Juegos de exploración de mazmorras

Para realizar un análisis de mercado se hace un listado de juegos con características similares. Los juegos se seleccionan de la siguiente forma:

- Se utiliza el buscador de Steam (https://store.steampowered.com).
- Se filtra según estos criterios:
  1. Incluir juegos con la etiqueta “Roguelike tradicional”.

2. Excluir juegos con la etiqueta “Generación procedural”
3. Excluir juegos con la etiqueta “Muerte permanente”

- Se ordenan por “Relevancia”.
  Steam publica la siguiente información: “El algoritmo «ordenar por relevancia» también asignará mayor peso a las etiquetas que hayas solicitado, lo que significa que verás resultados más relevantes al principio, lo que facilitará la búsqueda de títulos atractivos, independientemente de si son populares.” (https://store.steampowered.com/news/app/593110/view/1714119088658959583).
- Tomando los veinte primeros de la lista, se seleccionan los ocho juegos con más reseñas de usuarios. Steam no publica el número de ventas de un juego, pero existe una correlación entre ventas y número de reseñas.
- Por último, de esta lista de ocho se seleccionan manualmente aquellos cuatro que se considera que tienen mayor similitud con The Dungeon Rules.

La siguiente tabla muestra los juegos seleccionados junto con algunos datos de interés para este análisis:

NOMBRE	**EDITOR (AÑO)**	**PRECIO (EUROS) Fuente:steam**	**RESEÑAS Fuente:steam**	**ESTIMACIÓN DE VENTAS Fuente: steamdb**
Tangledeep	Impact Gameworks (2018)	12,49	899	30600 - 73300
Dungeonmans	Adventurepro Games (2014)	14,99	787	32100 - 82500
Rogue’s Tale	Epixx (2014)	6,49	474	19800 - 50900
Ultimate ADOM	Assemble Entertainment (2021)	19,99	272	6700 - 18300
VALORES MEDIOS		**13,49**	**608**	**22300 - 56250**

Tabla 4: Datos de juegos similares a The Dungeon Rules.

Los datos de la tabla son de Steam, excepto la estimación de ventas que se ha obtenido del sitio web https://steamdb.info. En steamdb tienen varios métodos de estimación de ventas. En la tabla se muestra la estimación de ventas basada en reseñas.

Estos datos ayudarán a:

- Definir un objetivo de ventas.
- Marcar un precio para el producto.
- Hacer una previsión de ganancias.
- Establecer la inversión máxima.

En este trabajo no se van a concretar estos aspectos económicos.

2.2.2. Juegos de puzles de tipo Sokoban

Después de hacer una búsqueda en Steam de juegos cuya mecánica principal es el puzle de tipo Sokoban, se concluye lo siguiente:

- Es un género recurrente. De los veinte primeros de la lista obtenida, la mayor parte de ellos fueron publicados desde 2018.
- Es un género con muchas menos ventas que la exploración de mazmorras. De los veinte primeros de la lista obtenida, la mayor parte de ellos tienen en torno a 100 reseñas.
- Tienen más éxito aquellos juegos que proponen una evolución de la mecánica Sokoban.

Se puede afirmar entonces que es un género con mucha oferta y en constante evolución. Sin embargo tiene un número de compradores muy reducido. Uno de los motivos podría ser que se puede jugar de forma gratuita a este tipo de juegos en sitios web y en aplicaciones móviles.

Especial atención merece el juego Baba Is You (Hempuli, 2019), un juego que cosechó gran éxito tanto en los usuarios como en la crítica especializada. A día de hoy tiene 13266 reseñas en Steam, y tiene una puntuación de 87 sobre 100 en el sitio web de reseñas Metacritic (https://www.metacritic.com/).

Con unos gráficos muy sencillos, Baba Is You ofrece un juego de tipo Sokoban con mecánicas muy originales, destacando la posibilidad de que el jugador cambie las reglas.

Tras analizar este juego, se decide que los puzzles de The Dungeon Rules deben ser lo más variado y original posible. Por ejemplo: empujar cajas sobre determinados lugares permite abrir/cerrar puertas, activar/desactivar trampas o bloquear el paso a enemigos para que no puedan perseguir al jugador. Las cajas se pueden destruir en determinadas circunstancias, e incluso se pueden empujar enemigos que hayan sido congelados con magia como si fueran cajas.

Figura 11: Baba is you (Hempuli). Fuente: https://store.steampowered.com.

2.2.3. Juegos de combate con cartas

El sistema de combate con cartas que implementa The Dungeon Rules es extremadamente sencillo. Permite resolver los encuentros con enemigos sin recurrir a métodos de tipo RPG. Son combates en los que el jugador debe tomar decisiones muy sencillas (atacar, defender o recuperar salud) en función de las cartas que tenga y de lo que tenga previsto hacer el enemigo.

El verdadero desafío para el jugador de The Dungeon Rules se encuentra en los puzles de tipo Sokoban que debe resolver mientras explora la mazmorra. Es por eso que no se analizan en profundidad los juegos basados en el combate con cartas.

Este tipo de juegos explota tras la publicación de Slay the Spire (Mega Crit Games, 2019), cuyas mecánicas han sido replicadas tal cual en muchos juegos posteriores. El combate tiene un componente muy táctico, de modo que el jugador debe seleccionar con mucho cuidado las cartas que utiliza en cada turno. El jugador tiene un mazo de cartas que se incrementa según se progresa en el juego. Las nuevas cartas tienen nuevos poderes, que a su vez se traducen en nuevas reglas. Esto aumenta la cantidad de acciones diferentes que puede llevar a cabo el jugador, aumentando también la complejidad de los combates y del juego.

The Dungeon Rules aprovecha una mecánica que está muy en alza actualmente, pero la implementa de una forma muy simplificada. En futuras versiones del juego se plantea la posibilidad de incrementar la complejidad de esta mecánica.

3. Propuesta

3.1. Especificaciones del producto

El objetivo del proyecto es el desarrollo de un videojuego con estas especificaciones:

- Género: explorador de mazmorras, puzles, combate con cartas.
- Gráficos 2D.
- Un jugador.
- Control con teclado y ratón.
- Idiomas: inglés.
- Compatible con Windows.
- Calificación de edad PEGI 7.

Las características que incluye el juego son:

- Estilo gráfico de tipo cartoon.
- Tutorial.
- 12 niveles de juego.
- 3 tipos de enemigos según su IA.
- 10 cartas de combate.
- 6 cartas de magia.
- Guardado de partida automático.
- Opciones: pantalla completa, resolución de vídeo y vibración de la cámara.

3.2. Estrategia de marketing

Darq (Unfold Games, 2019) es un pequeño juego independiente de puzles desarrollado por una sola persona, Wlad Marhulets. En éste, su primer juego, consigue muchos galardones, buenas críticas por parte de la prensa y mucha visibilidad en tiendas digitales como Steam. En 2020 Marhulets publica el libro “Gamedev: 10 steps to making your first game succesful” (Unfold Publishing, 2020). Se trata de una guía en la que explica paso a paso el proceso que siguió para convertir su idea en un juego comercial.

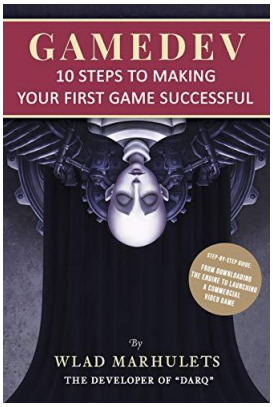

Figura 12: Portada de Gamedev: 10 steps to making your first game succesful. Fuente: amazon.es

Debido a que las circunstancias de desarrollo son similares a las de The Dungeon Rules (también es un pequeño juego independiente de puzles desarrollado por una sola persona), se toman como referencia las recomendaciones de Marhulets para llevar a cabo la campaña de difusión y marketing del juego.

Las acciones de difusión y marketing previstas se dividen en tres fases y están definidas por su objetivo principal:

Fase	**Objetivo principal**	**Acciones**
1	Presentación del juego.	Crear página web del juego. Elaborar un kit de prensa (se incluye en la web). Crear perfiles en redes sociales. Editar un trailer de presentación del juego.
2	Lanzamiento de la demo.	Crear página en tiendas digitales. Envíar notas de prensa. Envíar la demo a youtubers e influencers. Editar un trailer de la demo.
3	Lanzamiento del juego.	Envíar notas de prensa. Envíar el juego a youtubers e influencers. Editar el trailer definitivo del juego.

Tabla 5: Fases del plan de marketing.

Algunas observaciones sobre estas actuaciones son las siguientes:

- La fase de presentación empieza cuando se tiene una “vertical slice”. La página web y el trailer contienen un “call to action” para que los usuarios se suscriban a un “newsletter” y se hagan seguidores en redes sociales.

- La fase de lanzamiento de la demo empieza cuando se tiene una versión jugable. La demo se publica en tiendas digitales como Steam, Gog o Itchio. En esta fase el “call to action” de la página web y del trailer tiene como objetivo conseguir que los usuarios agreguen el juego a su “wishlist”. Estar en la “wishlist” de muchos usuarios aumenta la visibilidad del juego en la tienda.
- La fase de lanzamiento empieza cuando se publica la primera versión del juego. En esta fase el “call to action” de la página web y del trailer tiene como objetivo conseguir que los usuarios compren el juego.
- Se puede hablar también de una fase de pre-lanzamiento en la que se está muy activo en redes sociales. Se puede utilizar una cuenta atrás para incrementar la expectación, organizar un concurso que tenga como premio códigos del juego, y otros eventos similares.

4. Diseño

4.1. Entorno de desarrollo

4.1.1. Herramientas de desarrollo

El juego se ha desarrollado utilizando el siguiente hardware:

Equipo	Especificaciones
PC clónico	Procesador Intel Core i7 9700KF 3.6Ghz Placa base Gigabyte Z390 Aorus Pro Tarjeta gráfica Nvidia GeForce GTX 1660 Super Memoria RAM con capacidad de 16GB Almacenamiento NVMe M.2 con capacidad de 1TB
2 monitores de vídeo	Benq EW2420 Pantalla LCD 24 pulgadas Resolución 1920x1080
2 monitores de audio	Alesis M1 Active 520 Potencia 75W Altavoz de graves de 5 pulgadas (50W) Tweeter de 1 pulgada (25W) Respuesta en frecuencia de 56Hz a 20KHz Conexión XLR y TRS
Interfaz de audio	Behringer UMC22 Conexión USB Frecuencia de muestreo de 48KHz 2 entradas con conexión XLR y TRS 2 salidas con conexión TRS Alimentación fantasma de 48V
Teclado	Logitech
Ratón	Logitech

Tabla 6: Recursos hardware.

El juego se ha desarrollado utilizando las siguientes herramientas de software:

- Sistema operativo Windows 10 Pro. Se utiliza este sistema operativo porque es compatible con el resto de herramientas de software.
- Motor de juego Unity. El motivo principal por el que se utiliza este motor es porque el desarrollador tiene un buen conocimiento de él. Además, ofrece todo lo necesario para desarrollar un juego 2D como The Dungeon Rules. No se requieren grandes prestaciones porque los gráficos y animaciones del juego son muy sencillos.

- Adobe Photoshop. Es la herramienta de edición de imágenes más popular del mercado. En el desarrollo del juego también se utiliza para la creación de efectos, textos e incluso pequeñas animaciones.
- Reaper. Se decide utilizar Reaper para elaborar el diseño sonoro y la edición de audio. Es una herramienta muy económica y que cada vez se acerca más a las dos grandes DAWs que dominan el mercado de la producción de sonido, Protools (de Avid) y Nuendo (de Steinberg).
- DaVinci Resolve. Este software, propiedad del fabricante de equipos de vídeo BlackMagic, es muy conocido como herramienta para la corrección de color y el etalonaje. Pero en realidad es una suite que permite también la gestión de archivos multimedia, edición de vídeo, transcodificación y exportación. En el desarrollo del juego se utiliza para la edición de vídeo. Su versión gratuita permite trabajar con resoluciones de hasta Full HD.
- LibreOffice. Para todo el trabajo de ofimática se utiliza este paquete de aplicaciones que, además de ser gratuito, es compatible con los formatos más populares de procesamiento de textos, hojas de cálculo y documentos postscript.

4.1.2. Recursos y assets

El juego se ha desarrollado utilizando los siguientes assets de terceros:

- Puerta de castillo.
  Autor: Upklyak.
  https://www.freepik.es
- The dungeon – top down tileset.
  Autor: Zuhria Alfitra.
  https://assetstore.unity.com/packages/2d/environments/the-dungeon-top-down-tileset-112675
- Fantasy heroes: character editor.
  Autor: Hippo
  https://assetstore.unity.com/packages/2d/characters/fantasy-heroes-character-editor-90592
- Undead heroes: character editor.
  Autor: Hippo
  https://assetstore.unity.com/packages/2d/characters/undead-heroes-character-editor-100192

- A* Pathfinding Project.
  Autor: Aron Granberg
  https://arongranberg.com
- Los efectos sonoros, ambientes y loops musicales han sido obtenidos de la plataforma Soundly.
  https://getsoundly.com/

El resto de assets han sido creados por el desarrollador del juego.

4.2. Arquitectura general del juego: menús y pantallas.

4.2.1. Menú principal

Cuando la aplicación arranca muestra el menú principal, que permite ejecutar diferentes acciones o acceder a diferentes estados del juego.

El menú principal está formado por los siguientes botones:

- Salir: cierra la aplicación.
- Opciones: carga la escena que permite configurar las opciones del juego.
- Nueva partida: inicia una partida nueva.
- Continuar: continua una partida guardada previamente.
- Tutorial: inicia el tutorial.

Figura 13: Menú principal de The Dungeon Rules.

4.2.2. Nueva partida y continuar

El juego guarda de forma automática cuando el usuario supera un nivel.

La primera vez que el usuario arranca la aplicación no tiene disponible la opción de continuar. Esta opción queda habilitada cuando el usuario supera el primer nivel. A partir de este momento, si el usuario abandona la partida podrá retomarla en el último nivel alcanzado.

La opción de continuar también restaura el estado de juego que había en el último nivel superado, cargando el mismo valor de puntos de vida del jugador, pasos y cartas en el inventario.

La opción de nueva partida permite acceder al primer nivel del juego, pero además elimina la partida guardada, si la hubiera.

4.2.3. Tutorial

La opción de tutorial permite acceder al primer nivel del tutorial.

El tutorial es un modo de juego que tiene el mismo funcionamiento, estructura y objetos que el modo de juego normal. Su objetivo es enseñarle al usuario cómo se juega.

El tutorial está formado por diferentes niveles, y en cada uno de ellos se explica el funcionamiento de una o dos mecánicas del juego.

4.2.4. Opciones

El botón de opciones permite acceder a una pantalla que permite configurar los siguientes parámetros del juego:

- Full screen: permite seleccionar si la aplicación se muestra a pantalla completa o como ventana.
- Video resolution: permite seleccionar entre las diferentes resoluciones de vídeo disponibles en el hardware.
- Camera vibration: permite activar o desactivar la vibración de la cámara durante el juego.

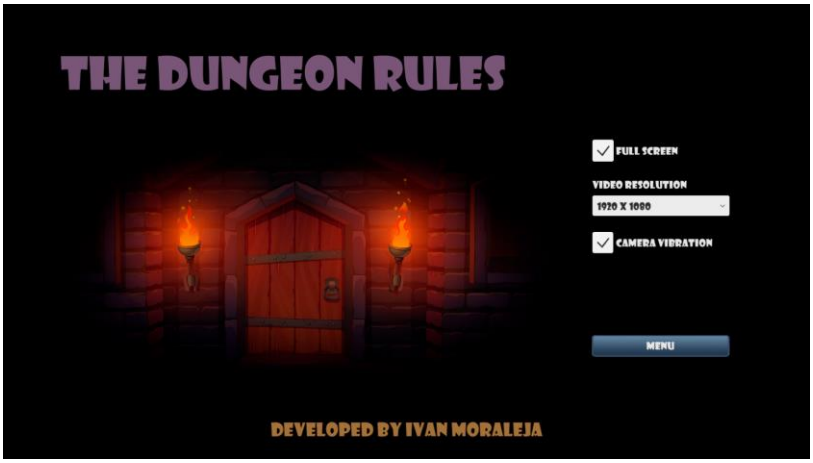

Figura 14: Pantalla de opciones de The Dungeon Rules.

4.2.5. Game Over

La pantalla de fin de partida aparece cuando el jugador pierde. El jugador solo puede perder cuando sus puntos de vida llegan a cero durante un combate.

Desde esta pantalla se pueden llevar a cabo dos acciones:

- Volver al menú principal.
- Continuar el juego de la misma manera que con la opción de continuar anteriormente explicada: carga el último nivel alcanzado y mantiene el estado del juego de ese momento.

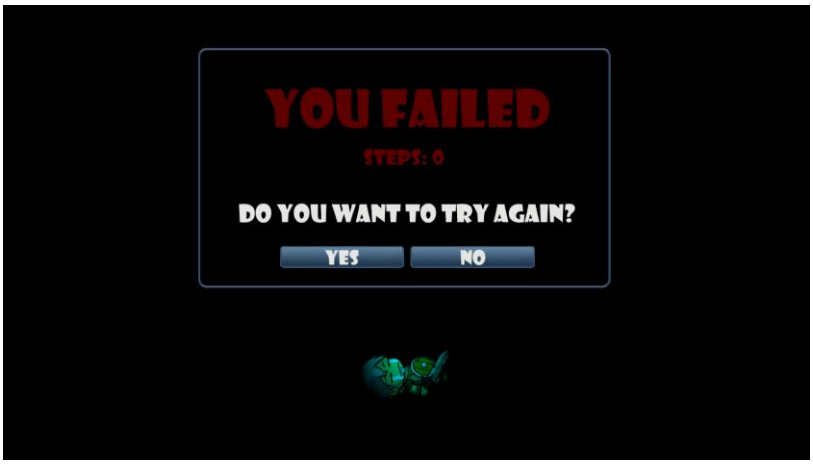

Figura 15: Pantalla de Game Over de The Dungeon Rules.

4.3. Componentes del juego

4.3.1. Controlador de juego

Cada nivel de juego tiene un objeto llamado GameManager que va a controlar el funcionamiento principal del juego.
A través de 3 scripts (GameController.cs, ScenesController.cs y TurnController.cs) realiza las siguientes tareas:

- Conmutar entre los dos modos de juego principales: exploración y combate. El modo habitual de juego, en el que el jugador se mueve por la mazmorra, es el modo exploración. Cuando el jugador se topa con un enemigo cambia la interfaz y se entra en modo combate. Del mismo modo, se vuelve al modo exploración cuando el jugador gana el combate.
- Activa/desactiva la interfaz de recompensa cuando el jugador encuentra un ítem.
- Gestiona los turnos, habilitando de forma sucesiva a cada personaje (jugador y enemigos) para que lleven a cabo su acción.
- Instancia el objeto Player en el lugar correspondiente (puerta de entrada a esa mazmorra).
- Almacena el estado de juego y carga el nivel correspondiente cuando se supera un nivel.

4.3.2. Mapa de nivel

El mapa de cada de nivel está estructurado en casillas, porque los personajes se mueven por casillas.
Se ha diseñado mediante Tilemaps. Se crea un objeto llamado TilemapGrid, que tiene un componente Grid rectangular, en el que cada casilla tiene un tamaño de 1x1 respecto a las medidas de referencia de Unity.
A su vez, el TilemapGrid contiene tres capas que son las siguientes:

- TilemapLimit: en este mapa se dibujan las paredes externas de la mazmorra. Tiene un componente TilemapCollider para que los personajes no puedan atravesar este tipo de bloques.
- TilemapWall: en este mapa se dibujan las paredes interiores de la mazmorra. Son iguales que los anteriores, pero podrán ser destruidos o atravesados con hechizos.
- TilemapFloor: en este mapa se dibuja el suelo. Solo tiene función estética.

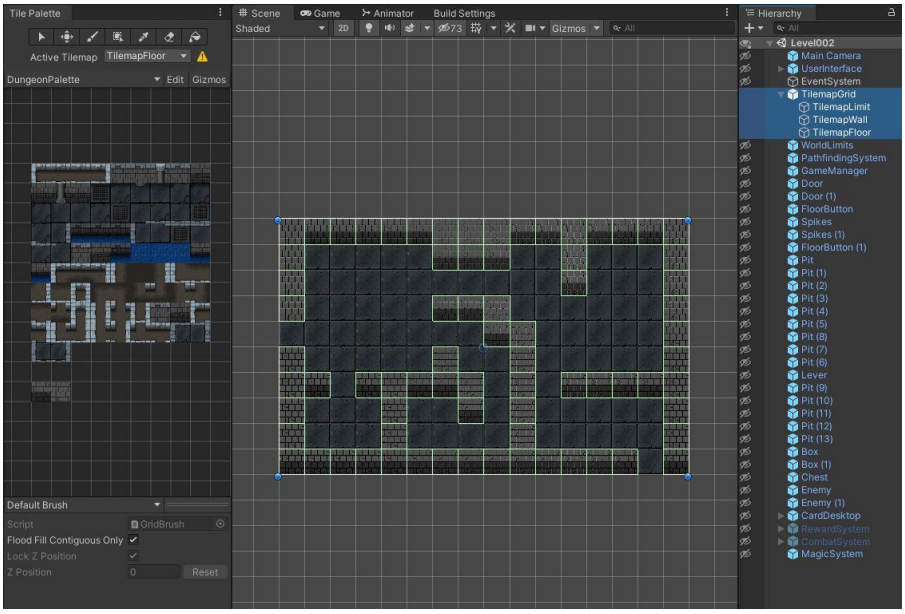

Figura 16: TilemapGrid de The Dungeon Rules.

4.3.3. El personaje jugador

Durante el modo de juego de exploración el usuario controla el objeto Player. Este objeto tiene el script PlayerMove.cs que tiene dos tareas principales:

- Procesar las entradas de teclado del usuario.
- Resolver la acción que ha solicitado el usuario.

El jugador se puede mover a razón de una casilla por turno. El usuario tiene cuatro direcciones posibles (arriba, abajo, izquierda y derecha), no pudiéndose desplazar en diagonal. Para ello se utilizan las teclas WASD o los cursores.

Cuando el usuario pulsa una tecla para desplazarse en una dirección se ejecuta un raycast en esa dirección a una casilla de distancia, y se analizan los objetos que pueda haber en esa casilla. De este modo, si la casilla está libre el personaje se desplaza a esa casilla, y pasa el turno.

Si por el contrario, se encuentra con un objeto que le impide el paso, como una pared, una puerta, una trampa de pinchos, una estatua o un foso, no ocurre nada y deberá mover en otra dirección porque no pierde el turno.

Si se desplaza hacia un objeto interactivo el personaje no se mueve, pero se produce una acción según el objeto del que se trate: la palanca se acciona, el botón de suelo se presiona, se empuja una caja o se abre un cofre. En estos casos también pasa el turno.

Si se desplaza hacia un enemigo el script emite el evento OnEnemyMeet, y el juego pasa de modo exploración a modo combate.

El objeto Player tiene un componente Animator para lanzar las animaciones de PlayerIdle y PlayerWalk, según esté parado o en movimiento. Se han diseñado dos objetos de tipo AnimatorController para el Player. Uno de ellos con espada y escudo, que es el que se utilizará habitualmente, y otro si armas. Este último se utiliza en el tutorial hasta que el personaje se equipa después de encontrar los objetos.

4.3.4. Objetos interactivos

4.3.4.1. Cajas

El jugador puede empujar cajas bajo estas condiciones:

- Que haya una casilla libre en esa dirección.
- Que haya un foso o una trampa de pinchos. En estos casos la caja queda destruida.

Cabe destacar dos hechos:

- Un enemigo también cuenta como obstáculo.
- El jugador no puede empujar más de una caja, de modo que si una caja topa con otra no podrán moverse.

Cuando el jugador interactúa con una caja (objeto Box), el script que lo controla (BoxController.cs) ejecuta un raycast, de la misma forma que ocurría con el Player, para analizar qué objetos hay en esa dirección a una casilla de distancia. En función del objeto que haya, y según las condiciones comentadas antes, puede ocurrir que la caja no pueda moverse, que quede destruida o que sí pueda moverse. En este último caso se emite el evento SetWalkable para que el Player también pueda moverse en esta dirección, detrás de la caja.

4.3.4.2. Palancas y botones de suelo

Las palancas y los botones de suelo (actuadores) tienen asociado un array de objetos sobre los que pueden actuar (target). Estos objetos pueden ser puertas, pinchos o puentes, en las combinaciones que se deseen.

Cuando se produce la interacción, los objetos target reaccionan, de modo que las puertas se abren o cierran, los pinchos suben o bajan y los puentes se pliegan o despliegan. Los scripts que controlan estos objetos target tienen métodos que se llaman igual (Block, Unblock, Invert) para que puedan ser ejecutados desde el objeto actuador independientemente de que sean puertas, pinchos o puente.

La única diferencia entre la palanca y el botón de suelo radica en que la palanca mantiene su estado después de que el jugador interaccione con ella, pero el botón necesita que un objeto esté sobre él para mantener el estado. Este objeto puede ser el Player, una caja o un enemigo. Cuando el objeto sale del botón, éste vuelve a su estado anterior.

Se ha implementado una mecánica en la que cuando una caja o enemigo está sobre una trampa de pinchos bajados o sobre un puente, y se actúa de modo que los pinchos suben, o el puente se pliega, la caja o el enemigo queda destruido.

4.3.4.3. Cofres y recompensas

Los cofres y las recompensas permiten al jugador obtener cartas de combate y de magia.

Los cofres se comportan como obstáculos. Cuando el jugador interactúa con un cofre, su script (ChestController.cs) emite el evento OnReward, para que GameManager active la interfaz de recompensas, que muestra la carta conseguida.

Cuando se derrota a un enemigo en combate aparece un objeto brillante en la casilla que ocupaba el enemigo. Cuando el jugador interactúa con él ocurre lo mismo que con el cofre: su script (RewardController.cs) emite el evento OnReward, para que GameManager active la interfaz de recompensas, que muestra la carta conseguida. En este caso, sin embargo, el objeto brillante desaparece.

4.3.4.4. Puertas

Las puertas se comportan como obstáculos cuando están cerradas y como casillas libres cuando están abiertas. Cada nivel tiene una puerta marcada como entrada (IsStart) y otra como salida (IsExit). No son objetos interactivos como tal, pero cabe destacar dos funciones importantes en el juego:

- La puerta de entrada siempre está abierta (porque narrativamente es la puerta de salida que se abrió en el nivel anterior). Es utilizada por GameManager para instanciar al Player en esa posición.
- La puerta de salida emite el evento OnExit cuando el jugador alcanza su posición. Este evento es utilizado por GameManager para cargar el siguiente nivel.

4.3.5. Enemigos

4.3.5.1. Tipos de enemigos

Existen 3 tipos de enemigos que se diferencian en su comportamiento y en otros aspectos que se explican a continuación.

El objeto Enemy es controlado por el script EnemyController.cs. Este script tiene definido un atributo de tipo EnemyData. La clase EnemyData.cs hereda de ScriptableObject, y permite determinar las características de los enemigos.

Los parámetros definidos en EnemyData son:

- Grade: especifica qué tipo de enemigo es.
- Animator y AnimatorCombat: permite asignarle una apariencia gráfica u otra según el tipo de enemigo.
- HPMax y HPMin: permite calcular sus puntos de vida (HP) de forma aleatoria entre estos dos valores.
- AttackMax y AttackMin: permite calcular sus puntos ataque de forma aleatoria entre estos dos valores en cada lance del combate.
- DefenseMax y DefenseMin: permite calcular sus puntos defensa de forma aleatoria entre estos dos valores en cada lance del combate.
- AttackProbability: es la probabilidad de que el enemigo ataque en su turno durante un combate.

Los enemigos pueden tener alguno de estos tres grados (Grade):

- Guardián (keeper).
- Patrullero (Patrolman).
- Depredador (Predator).

Para el juego se han creado tres esqueletos, uno con cada uno de estos grados.
Los parámetros que se les ha asignado son los siguientes:

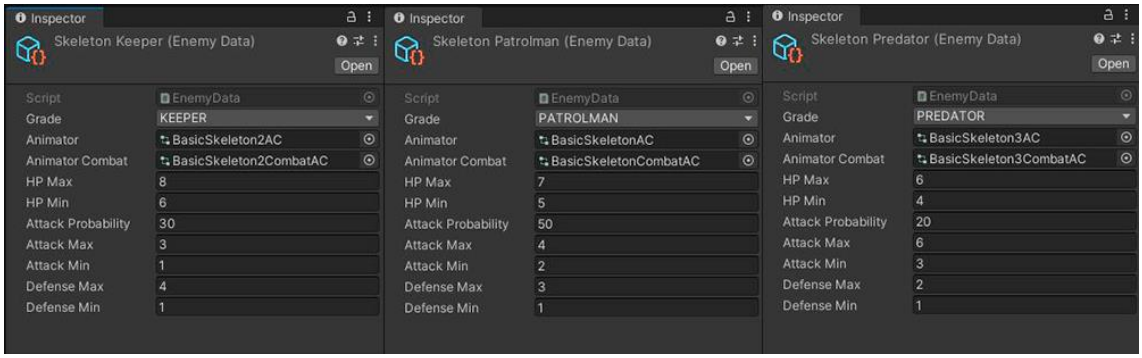

Figura 17: Tipos de enemigos de The Dungeon Rules.

- El esqueleto de tipo keeper es el que tiene más puntos de vida y más poder defensivo, pero menos poder ofensivo y probabilidad de ataque.
- Al contrario, el esqueleto de tipo predator es el que tiene menos puntos de vida y menos poder defensivo, pero más poder ofensivo y probabilidad de ataque.
- Por último, el esqueleto de tipo patrolman tiene valores intermedios en todos los parámetros.

4.3.5.2. IA de los enemigos

La IA de los enemigos se ha implementado mediante árboles de comportamiento. El comportamiento viene determinado por el grado que tenga cada enemigo. Los guardianes defienden una posición, los patrulleros se mueven por una zona y los depredadores persiguen al jugador.

<u>Guardián (keeper)</u>
El comportamiento de este enemigo es el siguiente:

- Se mantiene en su posición inicial.
- Si el jugador se acerca a una determinada distancia (alcance), el enemigo persigue al jugador.

- Cuando el jugador se aleja de esa distancia el enemigo vuelve a su posición.
- En caso de que el camino hasta el player esté obstaculizado, el enemigo vuelve a su posición inicial.
- En caso de que el camino a su posición esté obstaculizado, el enemigo finaliza su turno sin moverse.

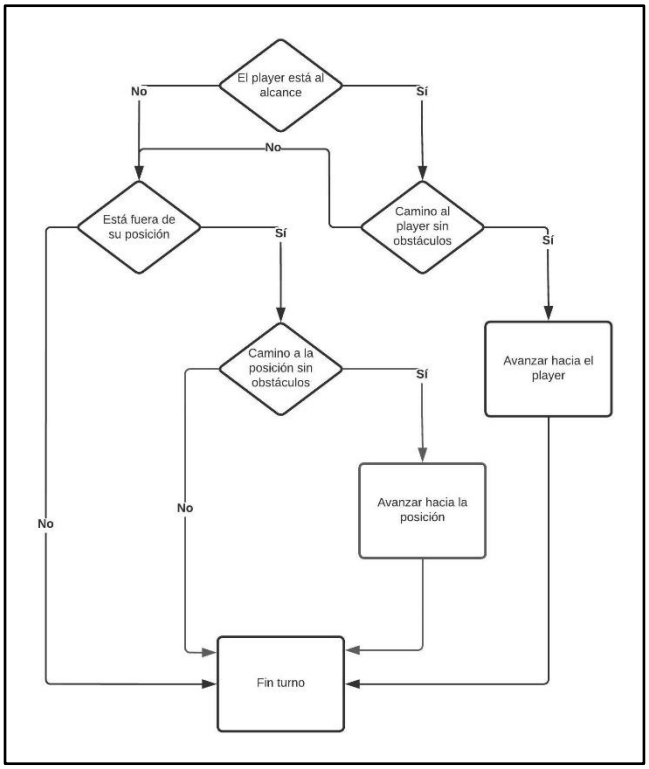

Figura 18: Árbol de comportamiento de enemigo de tipo guardián.

<u>Patrullero (patrolman)</u>

El comportamiento de este enemigo es el siguiente:

- Se genera una posición aleatoria no mayor de una distancia concreta (alcance) de su posición inicial.
- Si el jugador se acerca a esa distancia de la posición inicial, el enemigo persigue al jugador.
- Cuando el jugador se aleja a esa distancia de la posición inicial, el enemigo vuelve a dirigirse a la posición aleatoria.
- Si ya alcanzó la posición aleatoria se genera una nueva posición aleatoria, y se dirige a ella.
- En caso de que el camino hasta el player esté obstaculizado, el enemigo se dirige a la posición aleatoria.

- En caso de que el camino a la posición aleatoria esté obstaculizado, el enemigo finaliza su turno sin moverse.

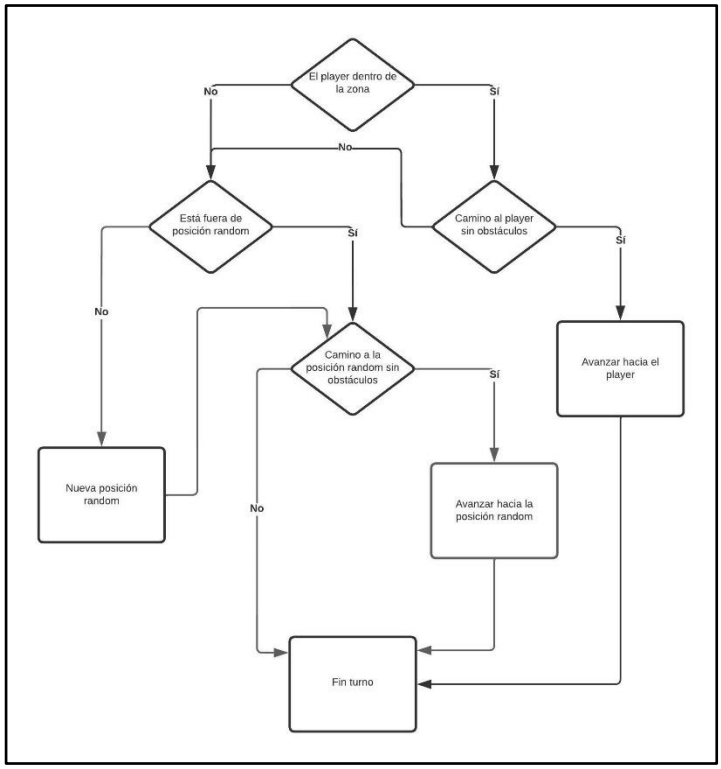

Figura 19: Árbol de comportamiento de enemigo de tipo patrullero.

<u>Depredador (predator)</u>

El comportamiento de este enemigo es el siguiente:

- Se mantiene en una posición.
- Si el jugador se acerca a una determinada distancia (alcance), el enemigo “ha visto” al jugador.
- Si el jugador ha sido visto alguna vez, el enemigo le persigue.
- En caso de que el camino hasta el player esté obstaculizado, el enemigo vuelve a su posición finaliza su turno.

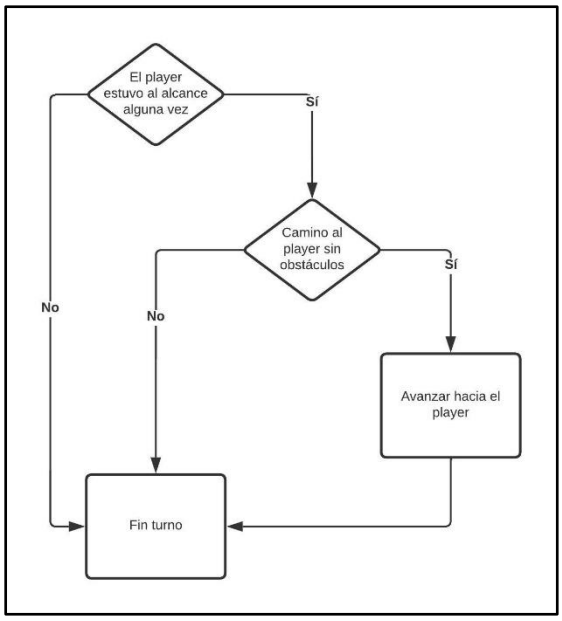

Figura 20: Árbol de comportamiento de enemigo de tipo depredador.

4.3.5.3. Pathfinding 2D

Para trazar los caminos que deben recorrer los enemigos para alcanzar su destino se ha utilizado la versión libre del asset “A* Pathfinding Project”. Este paquete proporciona clases que permiten trazar caminos en un entorno 2D de una manera sencilla.

Sobre un objeto (al que se ha llamado PathfindingSystem) se agrega el script Pathfinder.cs. Para configurar el pathfinder se crea la superficie (grid) sobre la que se van a hacer los cálculos, se especifica su tamaño y posición, y se indica que se va a trabajar en 2D.

También se activan las físicas 2D, se especifican las características del colisionador y se indica que los objetos no caminables están en la capa “obstacles”.

Los objetos que se van a incluir en la capa de “obstacles” son aquellos que los enemigos deben esquivar para trazar su camino. Estos objetos son: las paredes externas (TilemapLimit), las paredes internas (TilemapWall), las cajas (Box), los fosos (Pit), los pinchos cuando están arriba (SpikeUp), las puertas cerradas (DoorClose) y los cofres (Chest).

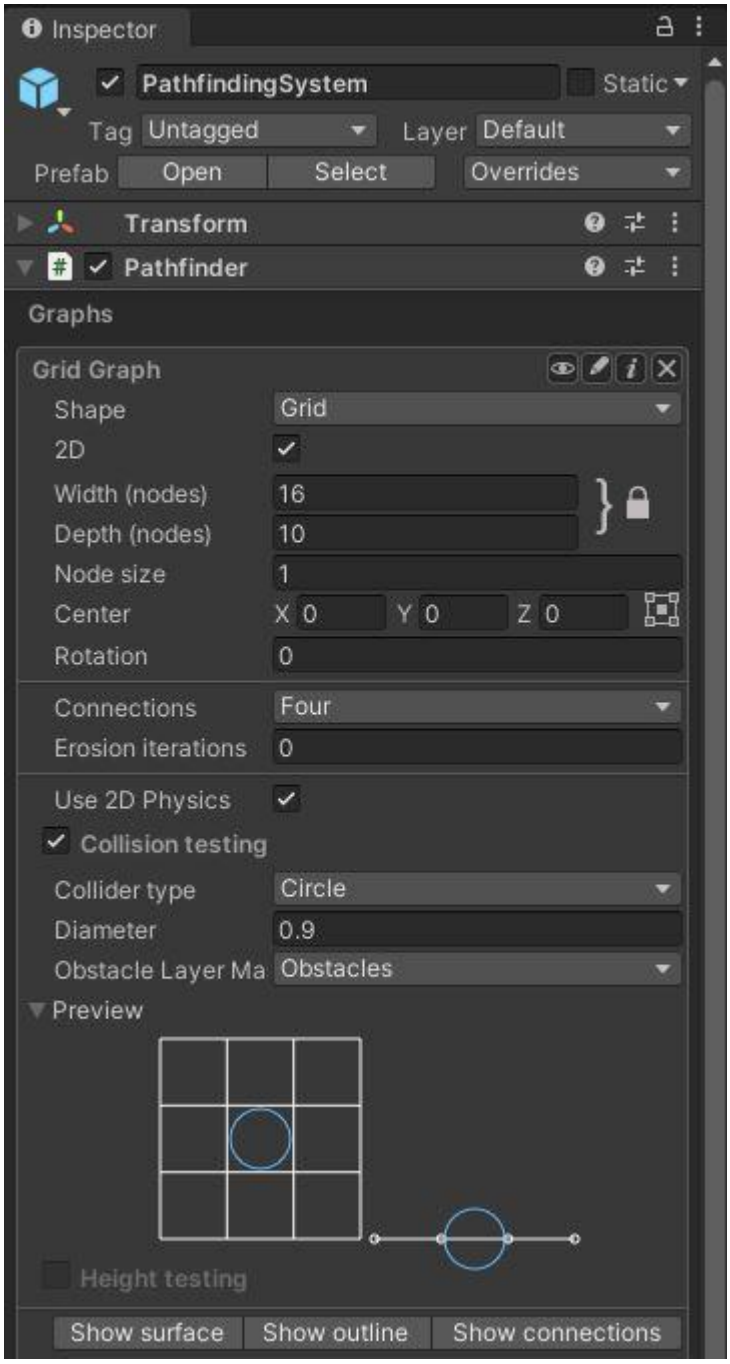

Figura 21: Configuración del Pathfinder.

A los enemigos se les agrega los scripts AIPath(2D, 3D).cs y Seeker.cs. Así se puede ejecutar el trazado del camino desde código. En el script EnemyController.cs se utiliza el método StartPath de la clase Seeker para calcular el camino. Este método proporciona un array con las posiciones del camino, desde la posición en la que se encuentra el enemigo hasta la posición de destino. Este array permite implementar los árboles de comportamiento de los diferentes tipos de enemigos que se han explicado anteriormente.

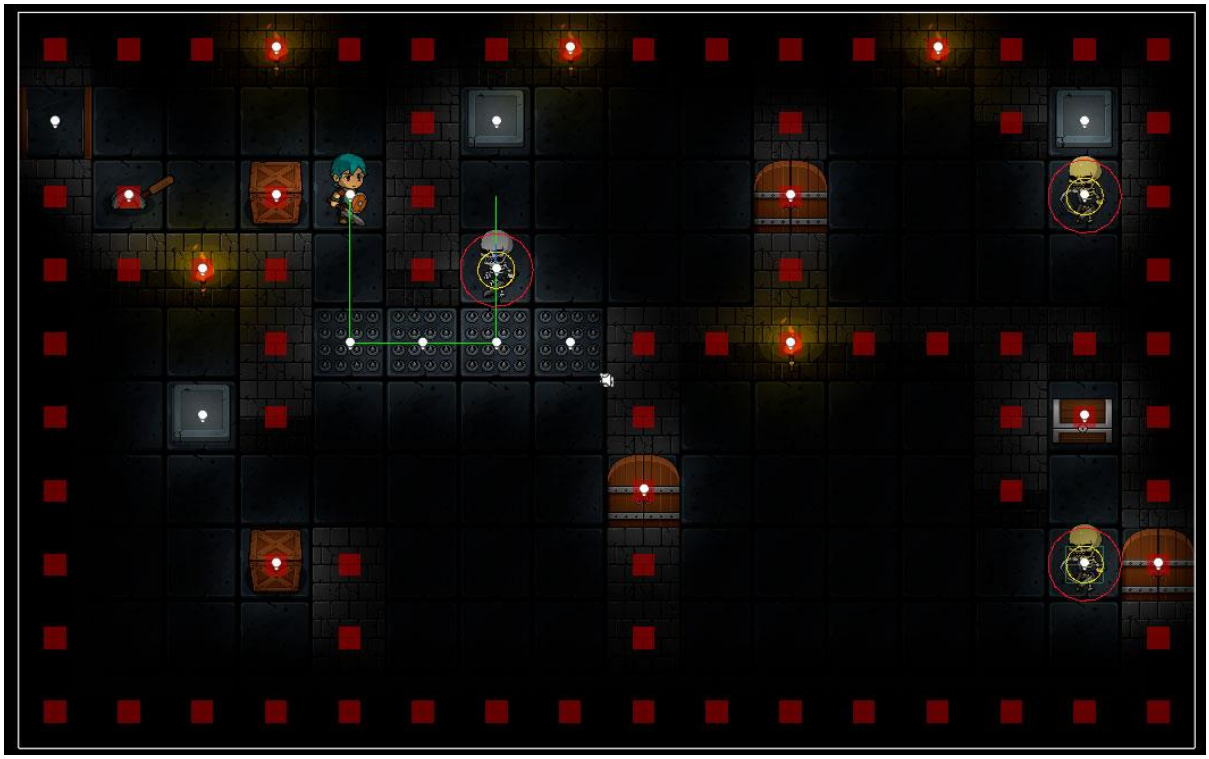

Figura 22: Información visual del script Pathfinder.

4.3.6. Cartas

Existen dos tipos de cartas: cartas de magia, que se pueden utilizar en el modo exploración, y cartas de combate, que se pueden utilizar en el modo combate.

Las cartas están representadas por el objeto CardPrefab, que se controla con el script CardController.cs. Este script tiene definido un atributo de tipo CardData. La clase CardData.cs hereda de ScriptableObject, y permite determinar las características de cada carta.

Los parámetros definidos en CardData son:

- CType: tipo de carta (combate o magia).
- CKind: las cartas de combate pueden ser de ataque, defensa o salud.
- CardName: nombre de la carta.
- Permanent: indica si la carta es permanente o de un solo uso.
- Range: indica si la carta tiene un valor constante o un rango entre dos valores.
- Min y Max: rango de valores de la carta.

- CImage: imagen de la carta que se muestra en pantalla.
- Description: descripción de la carta que se muestra en pantalla.

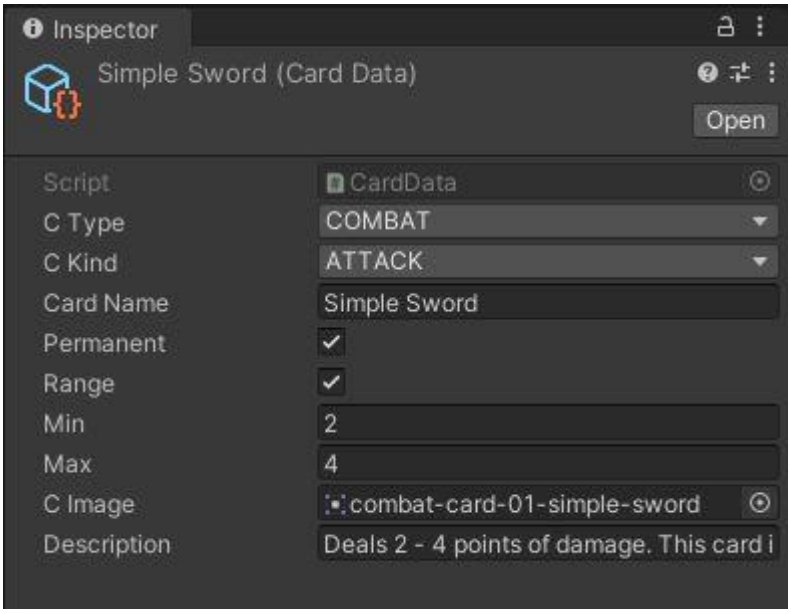

Figura 23: Ejemplo de CardData.

En tiempo de ejecución las cartas quedan almacenadas en dos listas de tipo CardData en la clase estática GameData. Estas listas son GameData.combatCards y GameData.magicCards.

El objeto CardDesktop muestra en pantalla una lista u otra en función del modo de juego en el que esté el usuario en ese momento: modo exploración o modo combate. Este objeto tiene un script llamado CardsManager.cs que se encarga de mostrar las cartas en pantalla, eliminarlas y seleccionarlas, entre otras cosas.

Figura 24: Ejemplo de CardDesktop en modo combate.

Figura 25: Ejemplo de CardDesktop en modo exploración.

4.3.7. Sistema de combate

Cuando el player se mueve hacia la casilla que ocupa un enemigo, o viceversa, GameManager conmuta al modo de combate y activa el objeto CombatSystem.

Este objeto tiene el script CombatManager.cs que se encarga de dirigir el combate. El combate se ejecuta mediante una sucesión de corrutinas que se van llamando unas a otras.

Las corrutinas que se ejecutan son las siguientes:

1. NextEnemyAction: muestra la acción que ejecutará el enemigo en su turno.
2. PlayerAction: después de que el jugador seleccione una carta se ejecuta esta corrutina que obtiene la acción del jugador (ataque, defensa o salud) y su valor. Llama a la corrutina ResolveStrike.
3. ResolveStrike: calcula el daño ocasionado por el personaje que tiene el turno. Llama a la corrutina CheckResult.
4. CheckResult: comprueba si los puntos de vida del player o del enemigo llegan a cero. Si no es el caso, pasa el turno y el combate continúa.
5. EnemyAction: ejecuta la acción mostrada en NextEnemyAction, y obtiene su valor. Llama a la corrutina ResolveStrike.

El interfaz muestra al usuario todos los datos del combate: acción que está ocurriendo en ese momento, puntos de vida de player y enemigo, siguiente acción del enemigo, acción de player y enemigo, y el resultado de la acción.

Figura 26: Interfaz de combate en el tutorial.

El combate termina cuando los puntos de vida de uno de los dos personajes llegan a cero. Si gana el jugador se emite el evento OnFinishCombat. Este evento lo recibe GameManager, que conmuta al modo de juego exploración y desactiva CombatSystem. Si gana el enemigo se carga la escena GameOverMenu.

Las cartas de combate implementadas son:

Simple sword	Twin swords	Thief knife	Big axe	Small potion
				
Ataque Daño de rango 2 – 4 Permanente	Ataque Daño de rango 1 - 8	Ataque Daño 3	Ataque Daño 5	Salud Cura el 50% de la vida máxima
Wood shield	**Steel shield**	**Soldier helmet**	**Knight armor**	**Big potion**
				
Defensa Protección de rango	Defensa Protección de rango	Defensa Protección 3	Defensa Protección 5	Salud Cura el 100% de la

1 -3 Permanente	1 - 8			vida máxima

Tabla 7: Cartas de combate.

4.3.8. Sistema de magia

Cuando el juego está en modo exploración GameManager activa el objeto MagicSystem. El objeto MagicSystem tiene el script MagicManager.cs que gestiona el lanzamiento de hechizos.

Cada hechizo tiene su propio script. Todos implementan la interfaz Spell, que tiene el método CastSpell. Cuando el usuario selecciona una carta de magia, el script del hechizo seleccionado se agrega en tiempo de ejecución al objeto MagicSystem, y se llama al método CastSpell. Después de que el hechizo sea ejecutado, el script se elimina de MagicSystem.

Los hechizos implementados son:

FireBall	Pass through wall	Destroy wall	Telekinesis	Stone statue	Master key
					
El jugador puede lanzar una bola de fuego que puede destruir un enemigo o una caja.	El jugador puede atravesar una pared.	El jugador puede romper un bloque de pared.	El jugador puede mover una caja desde lejos una posición en la dirección que desee.	El jugador puede convertir a un enemigo en una estatua de piedra.	El jugador puede abrir una puerta que esté cerrada.

Tabla 8: Cartas de magia.

4.3.9. Sistema de recompensas

Como se explicó en el apartado “Cofres y recompensas”, GameManager activa la interfaz de recompensas cuando se abre un cofre o se coge el alma de un enemigo derrotado.

Esta interfaz pertenece al objeto RewardSystem, que está controlado por el script RewardManager.cs.

Las recompensas se otorgan siguiendo estas reglas:

- Los cofres contienen cartas de magia y la derrota de enemigos otorga cartas de combate.
- El contenido de los cofres puede ser aleatorio o determinado por el juego.
- La recompensa de los enemigos es aleatoria salvo que el jugador tenga pocos puntos de vida. En este caso la recompensa será una poción de salud.

4.3.10. Persistencia

El juego guarda de forma automática el estado de la partida. Esto permite continuar el juego en el último nivel que se alcanzó la última partida que se jugó. Si se selecciona la opción de Nueva Partida en el menú principal la partida guardada queda eliminada.

La clase PersistentData.cs contiene los datos que se guardan. Estos datos son:

- Nivel alcanzado.
- Puntos de vida del jugador.
- Pasos ejecutados.
- Cartas de combate.
- Cartas de magia.

Los datos se guardan en el archivo game-data.json. La clase estática GameUtilities.cs contiene los métodos para crear, leer y escribir este archivo a partir de los datos de PersistenData. El archivo está en la carpeta Assets/StreamingAssets. Como esta carpeta se copia tal cual en la build, se asegura que el archivo estará presente.

La clase estática GameData.cs es la que se encarga de actualizar los datos de juego en tiempo de ejecución. Los momentos relacionados con la persistencia de datos en los que interviene son estos:

- Cuando se selecciona Nueva Partida en el menú principal, GameData resetea los valores del juego, se los pasa a un objeto PersistentData, y se escriben en el archivo game-data.json.

- Cuando se selecciona Continuar en el menú principal, GameData obtiene los valores de juego a partir de un objeto PersistentData, que a su vez los lee del archivo game-data.json.
- Cuando se supera un nivel, GameManager pasa los datos actuales de GameData a un objeto PersistenData, que los escribe en el archivo game-data.json.
- Cuando se recarga un nivel, un objeto PersistenData lee los datos del archivo game-data.json, y se los pasa a GameData.

4.4. Iluminación

4.4.1. Herramientas de iluminación

Para el diseño de iluminación del juego se ha utilizado el paquete Universal Render Pipeline (URP) desarrollado por la propia Unity.

Una vez instalados los componentes del paquete, se crea un asset llamado UniversalRenderPipelineAsset, que crea una instancia de flujo de render y controla los parámetros asociados al proceso de render.

En las opciones de proyecto (Project Settings), dentro del apartado de gráficos (Graphics), se asigna el UniversalRenderPipelineAsset como el objeto que va a encargarse del flujo de render.

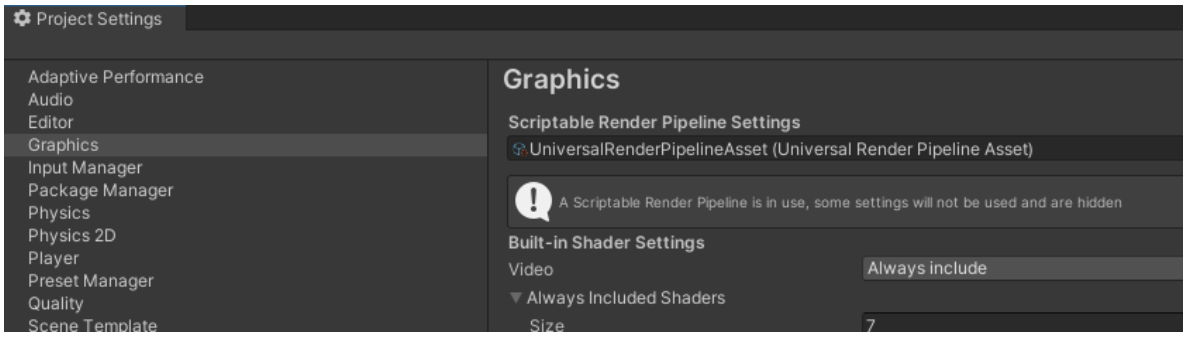

Figura 27: Configuración de URP.

Y por último, se crea un renderizador 2D, que se ha llamado 2D Renderer Data. Este renderizador se carga en la lista de renderizadores del UniversalRenderPipelineAsset.

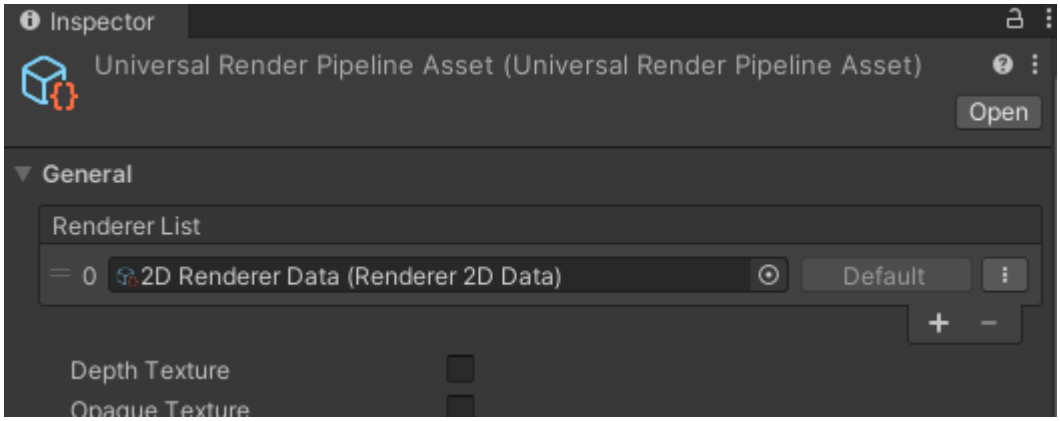

Figura 28: Carga de un renderer 2D en el asset URP.

4.4.2. Diseño de iluminación

Para diseñar la iluminación de los niveles de juego se ha creado el prefab LevelLight, que contiene dos tipos de luces:

- GlobalLight: es una luz uniforme y plana que se aplica sobre los elementos de interfaz de usuario (botones, textos, cartas, etc.).
- DungeonLight: es una luz de tipo freeform que permite dibujar el perímetro de la zona que se desea iluminar. En este caso se ha utilizado para iluminar la mazmorra de una forma muy tenue, para que quede muy oscura. La luz la van a proporcionar en gran medida los personajes y objetos.

Figura 29: Nivel solo con la iluminación GlobalLight y DungeonLight .

Para darle al entorno una estética más elaborada y atractiva se han utilizado antorchas, que tienen una luz anaranjada de tipo puntual.

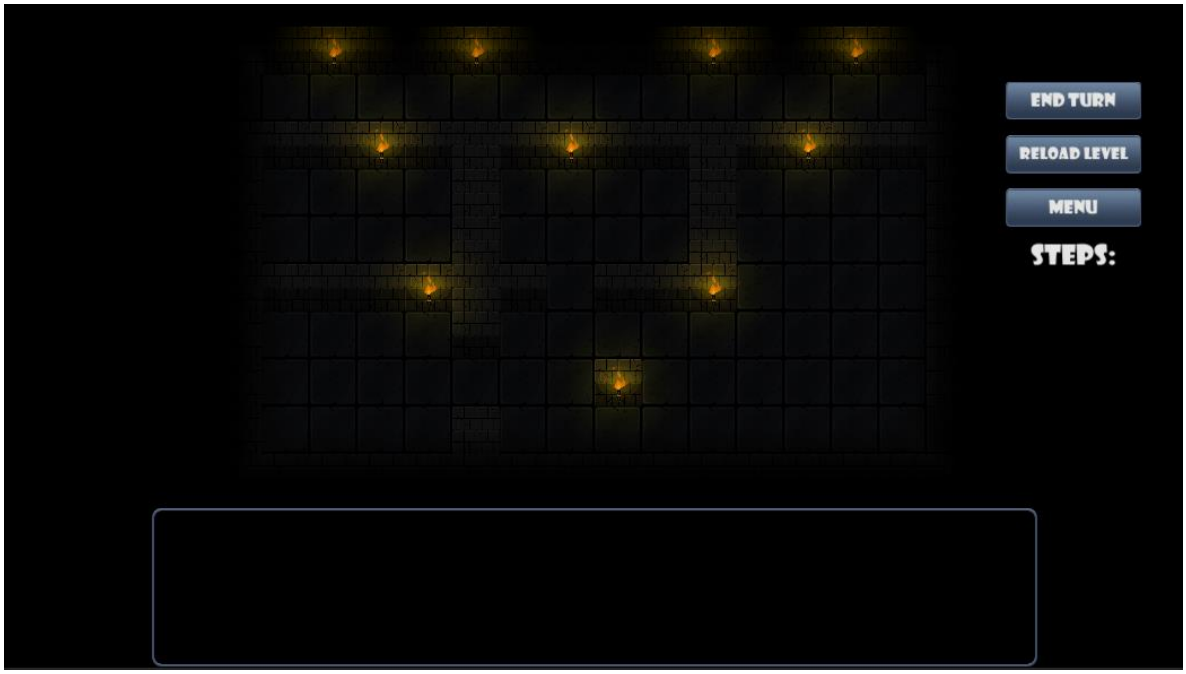

Figura 30: Nivel solo con la iluminación de las antorchas añadida.

Todos los personajes (player y NPCs) y objetos (puertas, pinchos, palancas, cofres, etc.) tienen una luz blanca de tipo puntual que tiene dos funciones: añadir contraste a la escena y llamar la atención del jugador sobre el objeto.

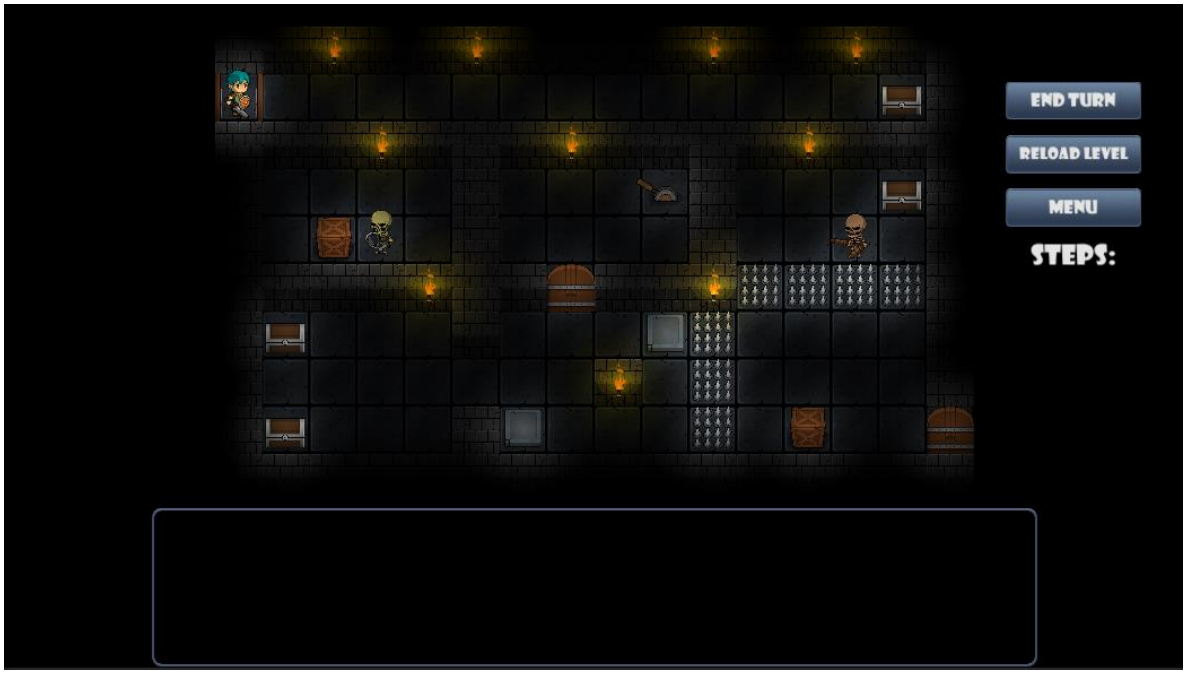

Figura 31: Nivel con la iluminación completa.

4.5. Sonido

4.5.1. Herramientas de sonido

Los assets de sonido se han obtenido a través de la plataforma Soundly (https://getsoundly.com/).

Soundly se presenta como una aplicación de escritorio conectada a una enorme librería de efectos sonoros en la nube. Cuenta con un potente buscador, y de una manera muy sencilla se descarga el archivo deseado simplemente arrastrándolo a una carpeta del ordenador. Es compatible con las DAWs (Digital Audio Workstation) más populares del mercado, de modo que, también permite abrir directamente el archivo seleccionado en la DAW, insertándolo en la pista seleccionada y en la posición donde se encuentre el cursor.

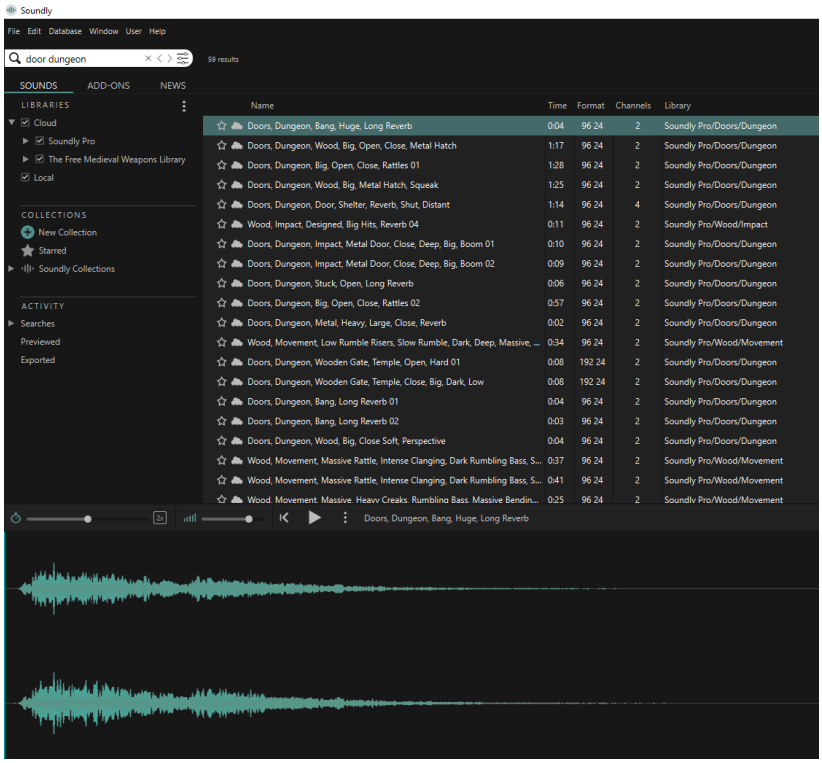

Figura 32: Parte de la interfaz de usuario de Sounly.

De Soundly no solo se han extraído los efectos sonoros, sino también los archivos para crear el sonido de ambiente y los loops musicales.

Para la edición de audio se ha utilizado Reaper, como ya se comentó anteriormente.

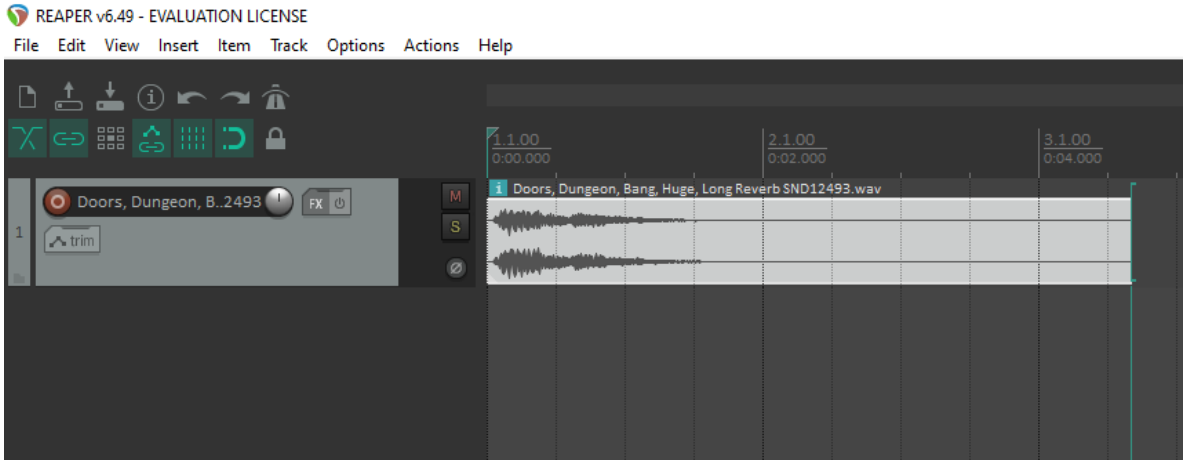

Figura 33: Parte de la interfaz de usuario de Reaper.

4.5.2. Diseño sonoro

El audio lo gestiona el prefab AudioManager, mediante el script AudioController.cs. AudioManager contiene un componente AudioSource para cada clip de sonido. Por su parte, AudioController tiene un método público llamado PlayAudio que reproduce el AudioSource requerido en cada momento.

La mezcla de audio tiene tres grupos:

- FX: controla el nivel de los efectos sonoros.
- Ambience: controla el nivel del sonido de ambiente. Se ha creado un ambiente sonoro que evoca un lugar solitario y siniestro como puedan ser una cueva o unas catacumbas. Este ambiente se reproduce en bucle en los niveles de juego.
- Music: controla el nivel de los fragmentos musicales. Hay dos bucles musicales, uno para los menús y otro para el combate.

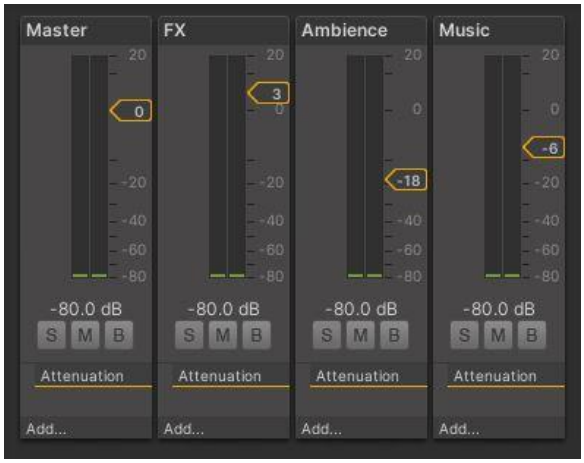

Figura 34: Mezclador de audio de Unity.

4.6. Diseño de niveles

4.6.1. Criterios de diseño

Los niveles se diseñan siguiendo los siguientes criterios:

- Dificultad creciente.
- Cada nivel se resuelve de una única forma. Es decir, hay que llevar a cabo una serie de acciones concretas, aunque se puedan resolver en distinto orden.

Podría ocurrir que haya varias formas de solucionar un nivel. En este caso el número de formas diferentes es muy reducido.

- En cada nivel se encuentran los recursos necesarios para resolverlo. No es necesario tener guardados hechizos de otros niveles.
- El número de enemigos varía entre 1 y 3.
- En aquellos niveles en los que haya más de 3 enemigos, estos pueden neutralizarse de alguna forma especial diferente del combate (trampa de pinchos, bola de fuego, transformar en estatua, etc.).

4.6.2. Niveles

A continuación se muestran los diferentes niveles que se han implementado.

Figura 35: Nivel 1.

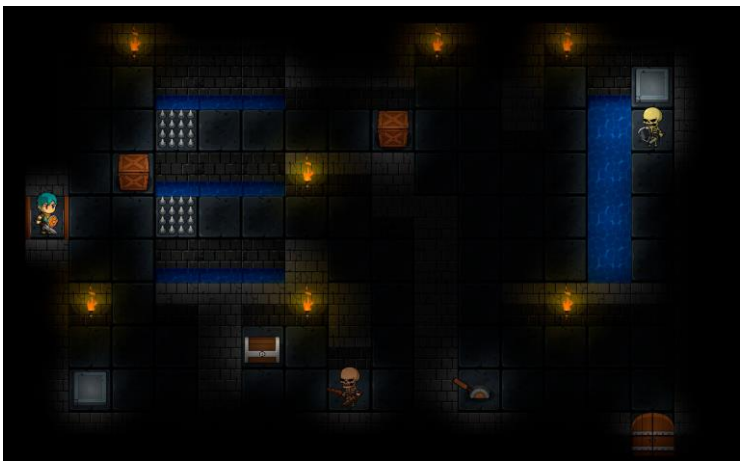

Figura 36: Nivel 2.

Figura 37: Nivel 3.

Figura 38: Nivel 4.

Figura 39: Nivel 5.

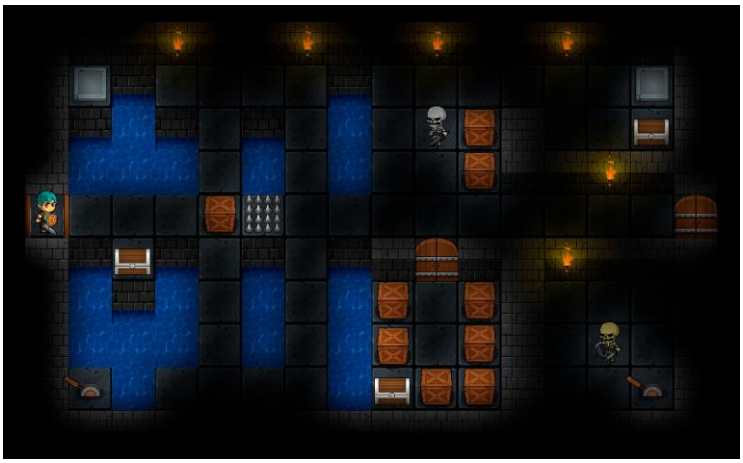

Figura 40: Nivel 6.

Figura 41: Nivel 7.

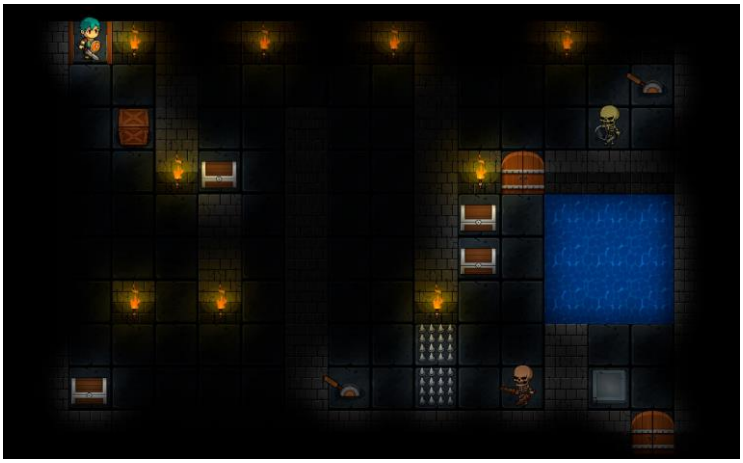

Figura 42: Nivel 8.

Figura 43: Nivel 9.

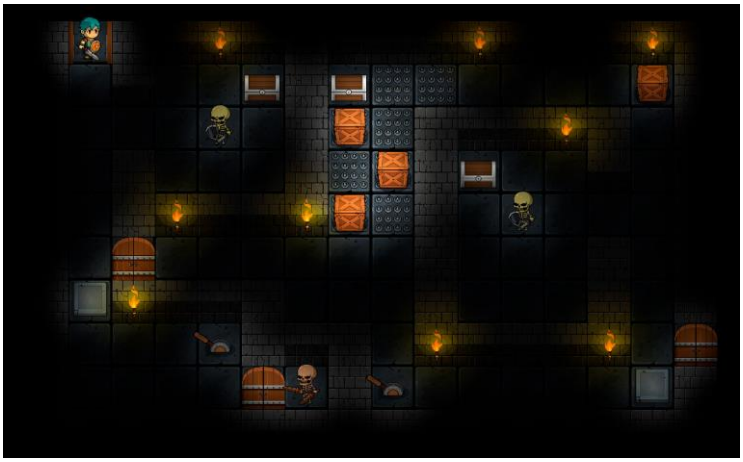

Figura 44: Nivel 10.

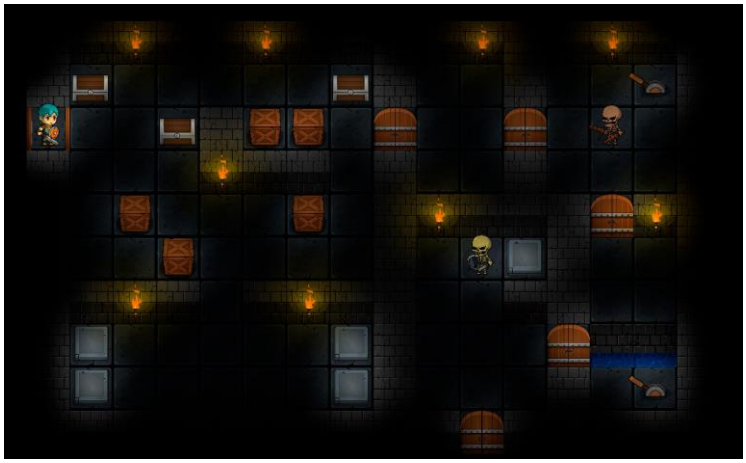

Figura 45: Nivel 11.

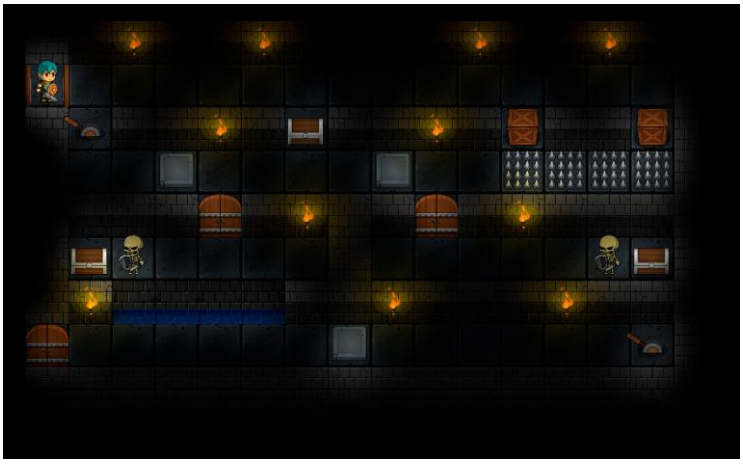

Figura 46: Nivel 12.

4.7. Balanceado

El jugador solo puede perder la partida en combate. Sin embargo, para balancear el juego se toma la siguiente decisión: el verdadero desafío del juego lo proporcionan las mecánicas de puzle, mientras que el combate se presenta como un complemento que no debiera ser difícil de superar.

De este modo, se balancea el juego asumiendo que los combates van a ser fáciles de ganar, lo que no implica que el jugador no pueda perder. Esto sucederá en el caso en el que un jugador juegue constantemente al ataque, y no se preocupe en ningún momento de defender o curar salud.

El balanceado se ha hecho mediante el método de prueba y error. Teniendo en cuenta los atributos de los enemigos (mostrados anteriormente en el apartado 4.3.5.1. Tipos de Enemigos), observamos lo siguiente:

- El enemigo de tipo guardián tiene una defensa máxima de 4. Si obtiene este valor, el jugador necesita un arma superior a la espada sencilla que lleva siempre equipada (daño aleatorio de 2 a 4).
  Por el contrario, este enemigo tiene un ataque que puede ser cubierto con el escudo básico de madera.
- El enemigo de tipo depredador tiene un ataque máximo de 6. Tiene un 50% de probabilidades de obtener 4,5 ó 6. En este caso el jugador necesita una defensa superior al escudo de madera que lleva siempre equipado (daño aleatorio de 1 a 3).
  Por el contrario, este enemigo tiene una defensa (1 - 2) que puede ser fácilmente superada con la espada sencilla. También tiene menos vida que los otros enemigos, para reducir la dificultad del combate.
- El enemigo de tipo patrullero tiene el mismo poder de ataque (2 - 4) y defensa (1 - 3) que el jugador cuando utiliza la espada sencilla y el escudo de madera.

Para decidir el valor de los ítems de combate se toman como referencia de los ítems permanentes.

En el caso de las armas, partiendo de la espada sencilla con daño aleatorio de 2 a 4, se implementan las siguientes:

- Cuchillo de ladrón: tiene un daño fijo de 3. Esto hará dudar al jugador entre utilizar este arma y la espada sencilla.
- Espadas gemelas: tiene un daño aleatorio de 1 a 8. Esto hace que pueda ser un arma muy buena o muy mala con respecto a la espada sencilla.
- Hacha grande: tiene un daño fijo de 5. Esta es la mejor arma del juego, y la que puede romper la defensa de 4 de un guardián.

En el caso de las defensas se procede de la misma forma, partiendo del escudo de madera con defensa aleatoria de 1 a 3, se implementan las siguientes:

- Casco de soldado: tiene una defensa fija de 2. Esto hará dudar al jugador entre utilizar este escudo o el de madera.
- Escudo de acero: tiene una defensa fija de 1 a 6. Esto hace que sea mejor que el escudo de madera, pero podría dar los mismos resultados. Podría contener el ataque máximo (6) de un enemigo de tipo depredador.
- Armadura de caballero: tiene una defensa fija de 5. Asegura una muy buena defensa, pero aun así no puede contener el ataque máximo (6) de un enemigo de tipo depredador.

Todas las recompensas de combate aparecen de forma aleatoria cuando se derrota a un enemigo, excepto las pociones de salud. En el caso de las pociones se siguen estas normas:

- Si el jugador termina un combate con un 25% de la vida máxima (16), o menos, obtiene una poción grande que restaura toda la salud.
- Si el jugador termina un combate con un valor comprendido entre el 25% y 50% de la vida máxima, obtiene una poción pequeña que restaura la mitad de la salud máxima (8).

Los enemigos tienen una media de 6 puntos de vida. Esto se decide después de hacer muchas pruebas, con el objetivo de que los combates se resuelvan con una media de 3 ó 4 lances (turnos para cada oponente).

5. Implementación

5.1. Instrucciones de ejecución

El juego está contenido en el archivo TheDungeonRules.zip.

Figura 47: Archivo TheDungeonRules.zip.

Para ejecutar el juego primero se deben descomprimir los archivos.

Para descomprimir los archivos se selecciona el archivo y se abre el menú contextual pulsando el botón derecho del ratón.

A continuación se pulsa la opción de Extraer todo.

Figura 48: Menú contextual del archivo.

En el diálogo que aparece se puede seleccionar la ubicación donde se desean copiar los archivos descomprimidos.

Después se ejecuta la descompresión pulsando Extraer.

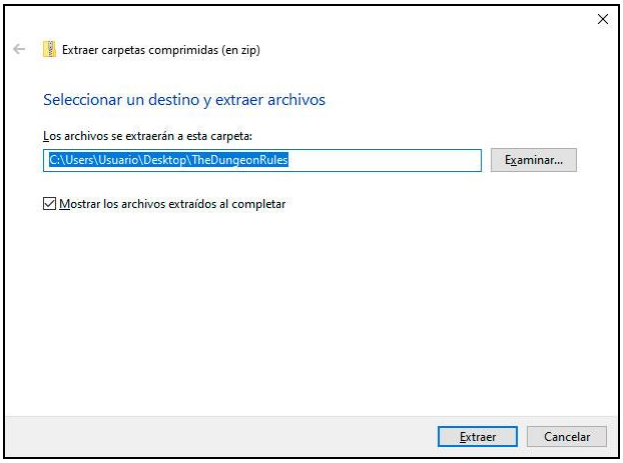

Figura 49: Diálogo para extraer los archivos.

Por último, se accede a la carpeta donde se han copiado los archivos descomprimidos, y se ejecuta el archivo TheDungeonRules.exe haciendo doble click con el botón izquierdo del ratón.

Figura 50: Archivos descomprimidos.

6. Demostración

6.1. Prototipos

6.1.1. Assets provisionales

Las primeras pruebas del juego se llevan a cabo con assets provisionales.
Estos assets se han obtenido en la página web Kenney (https://www.kenney.nl/). La licencia es de libre uso, tanto para proyectos personales como comerciales.

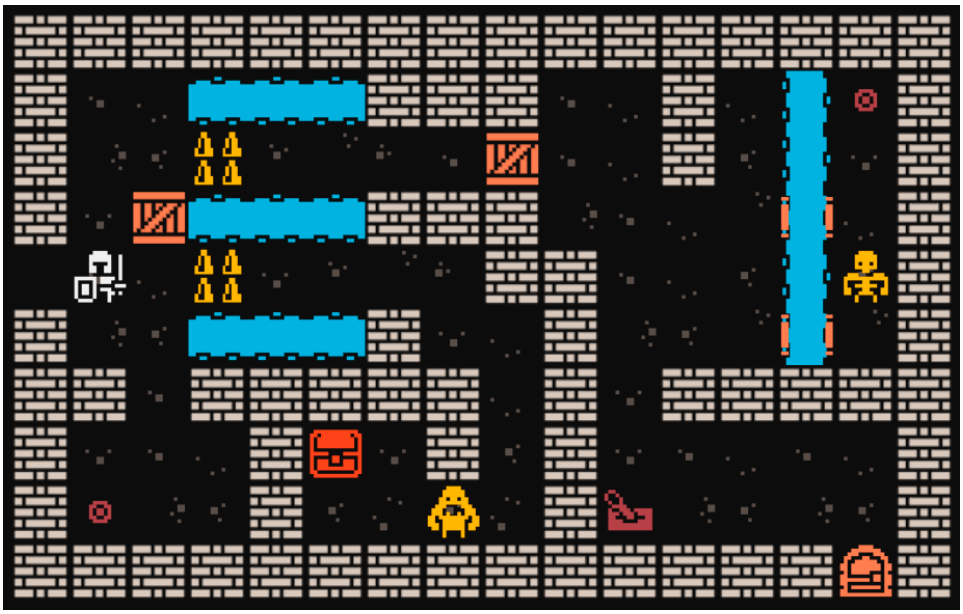

Figura 51: Prototipo con assets provisionales.

Se llegó a valorar que estos assets conformaran los gráficos definitivos del juego, pero finalmente se optó por utilizar una estética cartoon, para conseguir un aspecto más amigable, con la intención de atraer a un mayor número de jugadores.

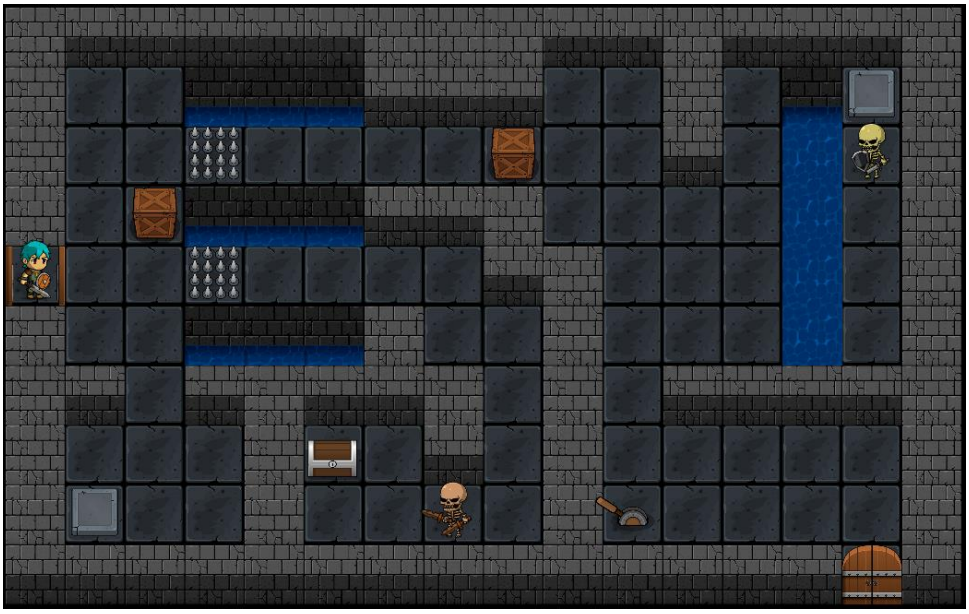

Figura 52: Assets definitivos con estética cartoon.

Para conseguir esta estética se compraron licencias en la web Unity Asset Store (https://assetstore.unity.com/). Los assets adquiridos son los siguientes:

- Undead heroes: character editor. Autor: Hippo https://assetstore.unity.com/packages/2d/characters/undead-heroes-character-editor-100192
- Fantasy heroes: character editor. Autor: Hippo https://assetstore.unity.com/packages/2d/characters/fantasy-heroes-character-editor-90592
- The dungeon – top down tileset. Autor: Zuhria Alfitra. https://assetstore.unity.com/packages/2d/environments/the-dungeon-top-down-tileset-112675

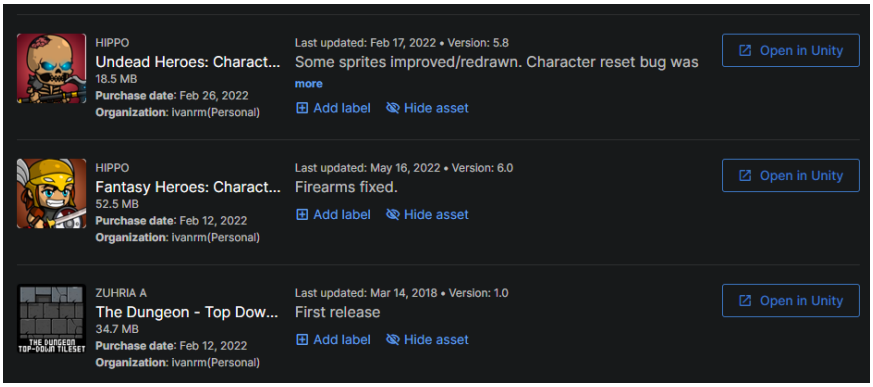

Figura 53: Assets adquiridos en la Unity Asset Store.

6.1.2. Bocetos de los niveles

El diseño de cada uno de los niveles empieza sobre papel y lápiz. Para facilitar el trabajo se imprimieron plantillas con una cuadrícula de 16x10 casillas, que coincide con las dimensiones de la mazmorra, según la unidad de medida de Unity.

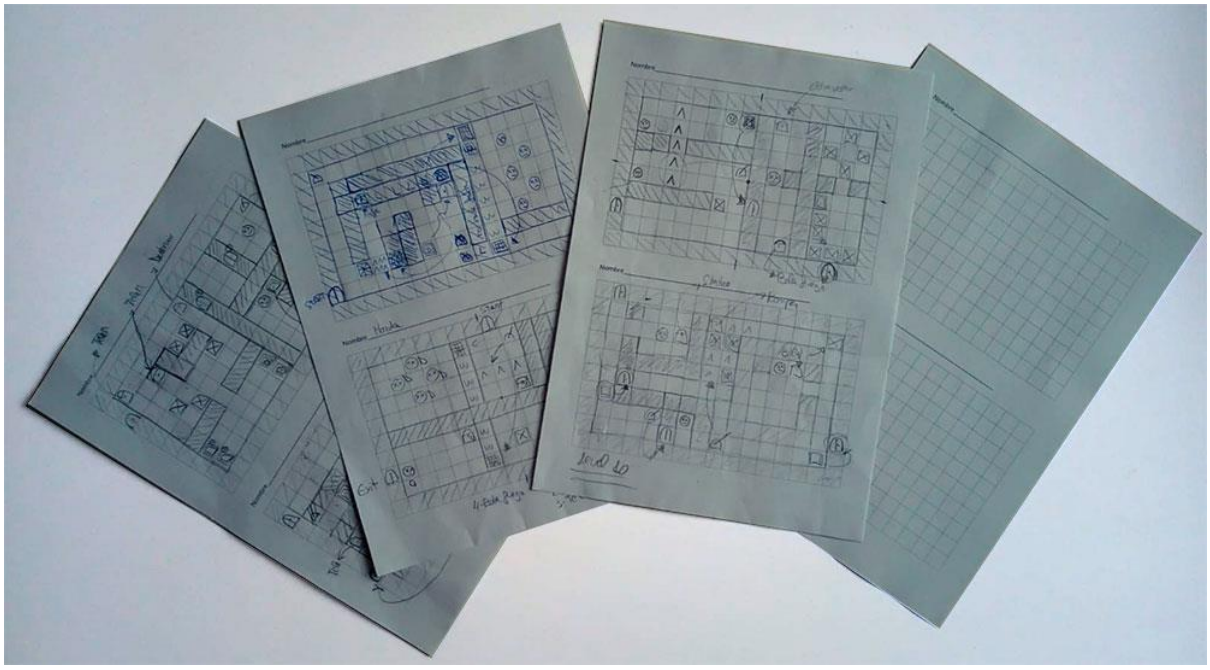

Figura 54: Bocetos de algunos niveles y plantilla.

6.2. Guía de usuario

El menú principal del juego presenta las siguientes opciones:

- Continue: pulse esta opción para jugar desde el nivel en el que se terminó la última partida.
- New game: pulse esta opción para iniciar una nueva partida.
  Esta acción borrará la partida guardada.
- Tutorial: pulse esta opción para acceder al tutorial del juego.
  El tutorial consta de 9 niveles en los que se explican las mecánicas básicas del juego.
  Cuando se superan todos los niveles del tutorial se regresa al menú principal.
- Settings: pulse esta opción para configurar algunos parámetros de visualización.
- Exit: pulse esta opción para cerrar la aplicación.

Se recomienda encarecidamente hacer el tutorial antes de iniciar una nueva partida por primera vez.

Este juego se controla con teclado y ratón. Los controles son los siguientes:

Acción	**Control**
Movimiento	WASD o cursores
Interactuar	Moverse hacia el objeto con WASD o cursores
Utilizar carta	Botón izquierdo del ratón

Tabla 9: Controles del juego.

Durante la partida se muestran tres botones en la parte superior derecha de la pantalla. Sus funciones son las siguientes:

- End turn: pulse esta opción para pasar el turno sin realizar ninguna acción.
  Esta opción no está disponible durante el transcurso de un combate.
- Reload level: pulse esta opción para volver a empezar el nivel actual.
  Utilice esta opción cuando llegue a una situación de bloqueo y el nivel no tenga solución.
- Menu: pulse esta opción para volver al menú principal.

Los parámetros de visualización que se pueden configurar en la pantalla de settings son los siguientes:

- Full screen: marque esta opción para visualizar el juego a pantalla completa.
  Desmarque esta opción para visualizar el juego con marco de ventana.
- Video resolution: seleccione en este desplegable la resolución de vídeo a la que se desea mostrar el juego.
  Se recomienda utilizar la resolución de 1920x1080 siempre que sea posible, porque el juego se ha diseñado para visualizarse con esta resolución.
- Camera vibration: la cámara vibra durante los combates cuando un personaje ejecuta un ataque.
  Marque esta opción para activar esta vibración.
  Desmarque esta opción para desactivar esta vibración.

7. Conclusiones y líneas de futuro

7.1. Conclusiones

Este videojuego es el fruto de más de un año de trabajo, desde que se empezaron a hacer los primeros diseños, pruebas y prototipos. Pero ha sido en los últimos cuatro meses cuando se ha materializado en una aplicación real, y con un nivel de acabado que permite ser jugado y disfrutado.

A lo largo de este tiempo de desarrollo se han aprendido muchas lecciones, y en muchos ámbitos diferentes.
Desde el punto de vista técnico, se ha profundizado mucho en los sistemas de pathfinding 2D y en la iluminación 2D.
En el área de programación, se han desarrollado scripts relativamente complejos, por ejemplo, para la gestión de cartas, para el sistema de combate o para determinados hechizos, como la destrucción de paredes, que se ha hecho manipulando directamente los tiles del mapa de nivel.
Otra lección aprendida muy importante ha sido la necesidad de iterar repetidas veces sobre el diseño del juego, hasta conseguir que todo encaje perfectamente. La falta de experiencia en este sentido, y la impaciencia por empezar a programar, hizo que se tuviera que tirar mucho trabajo hecho, y rehacerlo en varias ocasiones.

Aunque se rebajaron las especificaciones iniciales, por ejemplo, se pretendían incluir varios idiomas, o se pretendían crear más cartas y niveles, se considera que los objetivos se han cumplido.
El objetivo principal siempre fue crear una base sólida que permitiera escalar el juego con facilidad. La aplicación tiene actualmente un diseño que permite implementar, de forma muy sencilla, nuevos niveles y cartas.

La metodología de trabajo basada en la consecución de hitos, que académicamente se corresponden con las entregas evaluables (PEC), ha funcionado correctamente. Salvo la rebaja de especificaciones comentada anteriormente, debido a una estimación incorrecta del tiempo que requerían algunas tareas, la planificación inicial se siguió de forma bastante precisa.

Caben destacar también los comentarios y ánimos recibidos por parte del tutor del proyecto.

7.2. Líneas de futuro

El juego tiene mucho margen de mejora y muchas posibilidades de ampliación, tanto en contenidos como en jugabilidad.

Además de diseñar nuevos niveles, algunas propuestas de ampliación son las siguientes:

- Desarrollar una narrativa.
- Añadir más idiomas.
- Añadir más animaciones para las diferentes acciones.
- Crear diferentes “skins” para el player: con el objetivo de atraer a todo tipo de público, se diseñarían personajes con diferente género y diferentes rasgos físicos asociados a diferentes culturas (rasgos caucásicos, sudamericanos, afroamericanos, asiáticos, etc.).
- Añadir nuevas mecánicas de exploración: nuevas trampas, zonas oscuras que deben ser iluminadas, portales de teletransporte, etc.
- Añadir más tipos de armas y más tipos de hechizos.
- Añadir más tipos de enemigos.
- Incrementar la complejidad del combate: añadir “puntos de acción”, de modo que cada carta tenga un coste en puntos de acción, pudiéndose utilizar varias cartas en un turno. Añadir otros tipos de cartas que permitan hacer combos o jugadas más elaboradas.
- Incluir un “leaderboard” de pasos y de tiempo, para poder hacer “speedruns”.

Algunas características que serían para un plazo mucho más largo, y que encajarían mejor con una segunda parte del juego (The Dungeon Rules 2), son las siguientes:

- Gráficos 3D de tipo low-poly, con perspectiva isométrica.
- Incluir mecánicas de género metroidvania: las mazmorras estarían interconectadas. El player podría regresar a una mazmorra ya superada. Sería necesario este backtracking para progresar en el juego.
- Crear un editor de niveles, y una plataforma que permita a los jugadores publicar y compartir sus niveles con la comunidad, similar a la de Portal 2.

8. Bibliografía

Marhulets, W. (2020). Gamedev: 10 steps to making your first game succesful. Unfold Publishing.

UWM website, Thinking outside the box with sokoban and Baba is you, https://sites.uwm.edu/digital-cultures-collaboratory/2021/04/26/thinking-outside-the-box-with-sokoban-and-baba-is-you/#, consultado 28/02/2022.

RogueBasing website, http://roguebasin.com/, consultado 01/03/2022.

Steam website, https://store.steampowered.com/, consultado 25/03/2022.

SteamDB website, https://steamdb.info/, consultado 25/03/2022.

Metacritic website, https://www.metacritic.com/, consultado 28/03/2022.

Unity website: 2D Documentation, https://docs.unity3d.com/es/530/Manual/UnityManual.html, consultado 21/02/2022.

Aron Granberg website, A* Pathfinding Project, https://arongranberg.com/astar/, consultado 21/03/2022.

YOUTUBE website: 2DLights in Unity!, https://www.youtube.com/watch?v=nkgGyO9VG54 , consultado 16/05/2022.

YOUTUBE website: Luces 2D en Unity, https://www.youtube.com/watch?v=UYRfkrCmreI , consultado 16/05/2022.

YOUTUBE website: 2D Pathfinding, https://www.youtube.com/watch?v=jvtFUfJ6CP8 , consultado 21/03/2022.