UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Planificación, desarrollo y mantenimiento de un videojuego de sigilo

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Calero Dominguez, Marcos

Tutor/a: Lluch Crespo, Javier

CURSO ACADÉMICO: 2021/2022

Resumen

El principal objetivo en el que se basa este proyecto es la realización de un videojuego 2D con elementos de sigilo. Para su realización, se ha utilizado Unity, uno de los motores de videojuegos más utilizados en la actualidad. El proyecto contiene un conjunto de fases en los cuales el usuario deberá afrontar los obstáculos que se presenten, ya sea enfrentando a enemigos mediante el uso de objetos o evitándolos completamente a partir del escenario.

Palabras clave: Unity, Videojuego, 2D, Sigilo, IA, Prototipado

Resum

El principal objectiu en el qual s'ha basat aquest projecte és la realització d'un videojoc 2D amb elements de sigil. Per a la seva creació, s'ha utilitzat Unity, un dels motors de videojocs més utilitzats en l'actualitat. El projecte conté un conjunt de fases en les quals els usuaris hauran d'afrontar els obstacles que es presenten, bé a partir d'enfrontant els enemics mitjançant l'ús d'objectes o evitant-los completament a partir de l'escenari.

Paraules clau: Unity, Videojoc, 2D, Sigil, IA, Prototipat

Abstract

The main goal of this project is the development of a 2D video game with stealth elements. In order to make it, Unity has been utilized, one of the most used game engines right now. The project contains a couple of phases where the player must face different obstacles, either by defeating enemies with the use of objects or completely evading them by using the scene elements.

Key words: Unity, Video game, 2D, Stealth, AI, Prototyping

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En este primer apartado se presentará los motivos que han hecho que este proyecto haya sido realizado, que objetivos se han tenido en cuenta, la metodología empleada en la realización y planificación, y la estructura en la que se ha organizado el documento. Además, se presentará las colaboraciones presentes en el proyecto entre los distintos estudiantes.

1.1 Motivación

Desde pequeño, los videojuegos siempre han estado presentes en mí como la forma definitiva de entretenimiento. Siempre me he preguntado como era posible la creación de tanta variedad de diferentes mecánicas y géneros, y cómo estos podían combinarse entre sí como si fuera una especie de brujería. Este es uno de los motivos que me llevaron a escoger esta carrera y sobre todo este TFG, buscando un mayor entendimiento sobre su funcionamiento.

Además, mis compañeros y yo estuvimos presentes en la realización de otro proyecto mediante Unity, durante unas de las asignaturas pertenecientes a este curso. La experiencia fue recibida gratamente, permitió que obtuviera un planteamiento real sobre la realidad de desarrollar un videojuego, que factores hay que tener en cuenta y como aprender a adaptarse a resolver los problemas que surgen durante el desarrollo. Por esa razón y experiencia previa, obtuve un sentimiento de superación e ilusión de poder aplicar mis conocimientos a un proyecto más avanzado y buscar un mayor conocimiento a través de tutoriales [1].

1.2 Objetivos

El propósito principal de este TFG se basará en el desarrollo de un videojuego que contenga unas mecánicas entretenidas e interesantes que sean capaces de dar una sensación satisfactoria al jugador. Debido a ello, se ha organizado una serie de objetivos que mostrarán la envergadura del proyecto, mostrando todo el trabajo que conlleva su realización:

- **Género de plataformas y sigilo:** La principal distinción que debe presentar es que se base en una experiencia compuesta por mecánicas de videojuegos de plataformas, que a su vez, contiene elementos de sigilo cuya principal funcionalidad será dotar al jugador de diversas opciones para poder afrontar las dificultades que se le presentan en cada nivel.

- **Alta interactividad del entorno:** Cada escenario se compondrá de diversos elementos interactivos con un conjunto de acciones del jugador. Estos elementos darán una mayor jugabilidad al usuario y le permitirán sentirse más unido al mundo.
- **Enemigos reactivos a estímulos:** Los enemigos se diseñarán para que sean capaces de poder reaccionar a diversas acciones del jugador y a ciertos elementos que se encontrarán en el escenario, de esta forma el usuario puede experimentar con diversas formas de superar sus obstáculos sin combatir.
- **Posibilidad de jugar con un amigo:** El juego permitirá la posibilidad de jugar *online* con otro usuario, de forma que pueda ser experimentado solo o con la compañía de un amigo.

Adicionalmente, se ha optado por tener como objetivo seguir una metodología ágil para estructurar la organización y desarrollo del proyecto, de forma que se eviten retrasos y tengamos un ritmo de trabajo constante y eficiente.

1.3 Metodología

Durante el desarrollo de este proyecto, se ha optado por seguir una metodología ágil, siguiendo el modelo Scrum. Uno de los motivos que nos ha llevado a la aplicación de este tipo de metodología ha sido la flexibilidad y rapidez que presenta a los desarrolladores, sobre todo en ambientes colaborativos como es nuestro caso. Adicionalmente, cada vez más empresas y diferentes estudios buscan aplicar este modelo de trabajo, apoyado por diversos estudios que demuestran su mayor satisfacción conforme a metodologías más tradicionales. En la figura [1.1], podemos ver un estudio realizado por **Standish Group** [2], donde se demuestra como presentan un porcentaje de éxito muchísimo más grande.

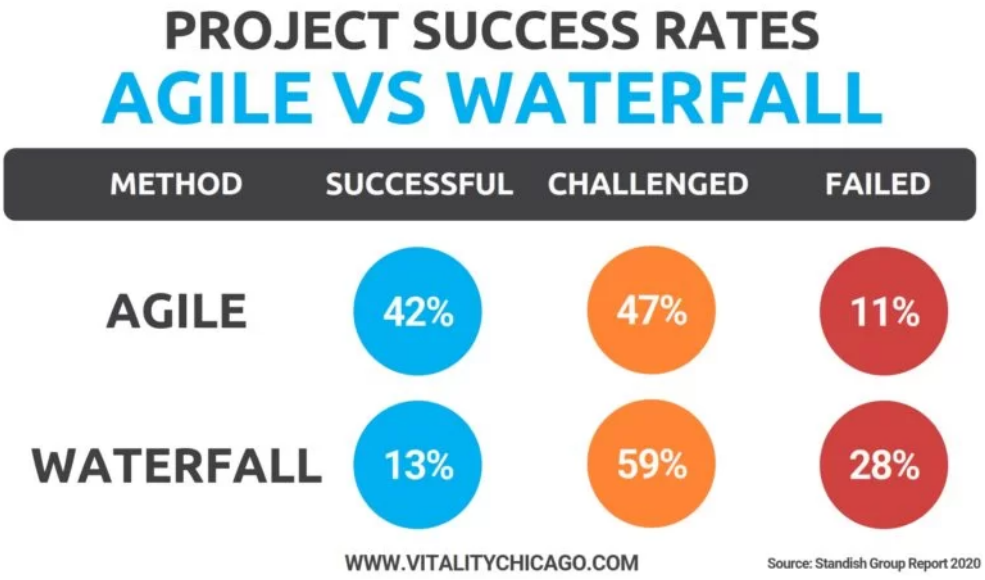

Figura 1.1: Modelo Ágil vs Cascada

Existen tres roles fundamentales en los que se compone este método de trabajo:

- **Scrum Master:** Su misión es gestionar el equipo para garantizar que se cumplen los objetivos de la metodología ágil, facilitando la comunicación entre el cliente

y el equipo de desarrollo. Debe tener un conjunto de habilidades estratégicas y comunicativas, actuando como el *entrenador* del equipo.

- **Product Owner:** Busca mantener la visión del producto final, asegurando que se cumplen los objetivos y necesidades del cliente. Su objetivo es que se maximice el valor de cada entrega. Es el principal encargado de interactuar con los *stakeholders* y manejar el **Backlog**.
- **Equipo de desarrollo:** Conjunto de trabajadores entre cinco y nueve personas con un objetivo en común. Se encargan de cumplir las tareas en el plazo establecido e interactúan con los *stakeholders* para entender sus necesidades.

Este tipo de metodología se compone de distintas fases de implementación, que se van aplicando durante cada *sprint*, donde se produce una nueva versión del producto:

- **Planificación:** Cada *sprint* resalta por tener un fin en particular, especificado en esta primera fase. Durante esta etapa, se presentarán los objetivos y desarrollo de UTs, posibles riesgos, se establecen los plazos de entrega y el conjunto de pruebas de aceptación.
- **Desarrollo:** Durante esta fase, se procede a implementar todas las UTs que se han especificado en la fase anterior y ejecutar sus correspondientes pruebas de aceptación, en el periodo establecido que se compone el *sprint*. Este proceso suele durar entre un mes o unas cuantas semanas.
- **Revisión:** Una vez acabado el *sprint*, se procede a la evaluación mediante el cliente, en el que se muestran los avances del proyecto y se hace una evaluación del proceso y como ha sido llevado. Suele realizarse mediante un estudio del *Burn Down* dirigido por el Scrum Master, donde se presenta un conjunto de gráficos que muestran cómo ha ido el avance durante el *sprint*.
- **Retroalimentación:** Se aplica lo aprendido durante la etapa de desarrollo y revisión, donde estudiamos posibles cambios en la planificación de los sprints siguientes para aplicar los nuevos conocimientos adquiridos.

1.4 Estructura de la memoria

Durante esta sección se explicarán las diferentes partes en las que se estructurará nuestra memoria, dando una breve explicación sobre cada uno de ellas:

- El primer capítulo consistirá en una **introducción**, donde se presentarán las principales motivaciones en las que se basará este proyecto, al igual que describiremos la metodología seguida durante este periodo de tiempo.
- A continuación, se presentará el **estado del arte**, donde se hablará del estado de los videojuegos, desde sus origines hasta nuestra época actual, centrándonos en aquellos con más relevancia próxima a nuestro proyecto, ya sea debido a su similitud en género o temática.
- Seguidamente, en el **análisis del problema** se hablará en términos generales del juego, donde nos centraremos en los primeros pasos seguidos en el desarrollo, describiremos como hemos aplicado la metodología utilizada y los requisitos que se han tenido en cuenta a lo largo de todo el proyecto.

- Posteriormente, se presentarán los apartados principales, el **diseño** y **desarrollo** de la solución. Estos aparados darán una explicación más extensa de todos aquellos elementos de los que se compone nuestro proyecto. Durante la sección de diseño, se explicará de una forma más general como se llevó a cabo el proceso de planificación, proponiendo propuestas para afrontar el implemento de todas estas soluciones. Por otro lado, el apartado de desarrollo se centrará en cómo se ha producido la implementación del videojuego por medio de Unity, expresando como se han llevado a cabo todos los elementos mencionados en la sección de diseño.
- En la próxima sección se encontrará el apartado de **pruebas**. Durante esta fase de desarrollo, el videojuego será ofrecido a un conjunto de compañeros para que presenten su opinión e identificar posibles mejoras o añadidos que podrían incluirse para brindar una mejor experiencia. Todos estos detalles serán explicados durante este apartado.
- Ulteriormente, se mostrarán las **conclusiones** llegadas a partir de la realización y escritura de todo el proyecto.
- Para finalizar se explicará que relación presenta con los estudios y que **trabajos futuros** se podrían realizar, donde se dará paso a detallar mejoras que podrían añadirse en caso de haber presentado un mayor periodo de tiempo.

1.5 Colaboraciones

La realización de este trabajo se ha llevado a cabo por medio de una colaboración entre dos compañeros, los cuales se han encargado de desarrollar diferentes componentes del videojuego. Seguidamente, se presentará cada uno de los compañeros presentes en el proyecto y su trabajo realizado:

- **Daniel De Castro Isasi**: Inteligencia artificial de los enemigos y jefe final, movimiento y comportamiento del protagonista, combate, iluminación y gestión de las animaciones.
- **Marcos Calero Domínguez**: Encargado principal de las interfaces e inventario, persistencia, interacción con el mapa, cuadros de texto y sonido.
- **Eva Vidal García**: Implementación y desarrollo de un videojuego de sigilo con infraestructura multijugador. Sistema online.

CAPÍTULO 2
Estado del arte

En los apartados que se muestran a continuación, se hará una retrospectiva de la historia en la que se basa nuestro proyecto, como ha ido progresando conforme han pasado los años hasta la época actual, y que alternativas existen en el mercado.

2.1 El videojuego como medio de entretenimiento

La industria de los videojuegos es una de las industrias de entretenimiento más grandes de estos últimos años, superando incluso a grandes exponentes de nuestra cultura como el cine o la lectura y generando un conjunto de beneficios enorme. Además, durante este periodo de desconexión con el mundo exterior producido a partir del confinamiento, se mostró de forma definitiva como los videojuegos tienen una importancia indudable como fuente de entretenimiento y relajación, así mismo como una vía de educación para los más jóvenes.

Su concepción se remonta a los años 50, considerado como el primer concepto de videojuego llamado **Nim**, basado en el antiguo juego matemático con el mismo nombre. Este producto se presentó en la prestigiosa feria mundial de Nueva York, la cual se realizó entre 1939-1940. Más tarde, se presentó el desarrollo de uno de los primeros computadores creado de forma exclusiva para poder ejecutar un videojuego conocido como Nimrod, debido a que fue concebido teniendo como objetivo simular **Nim** [3].

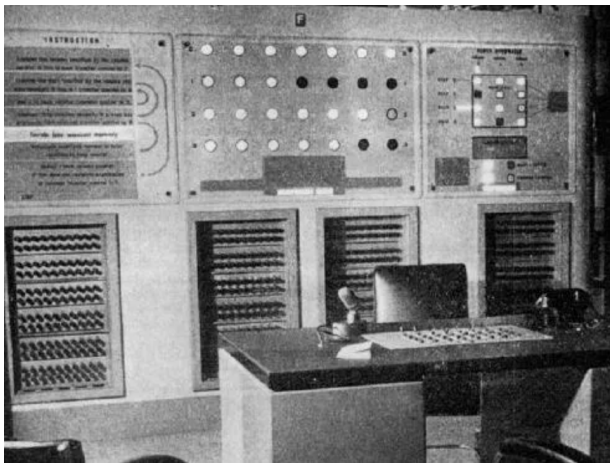

Figura 2.1: Nimrod, primer ordenador desarrollado para ejecutar un juego

No obstante, la comercialización de la primera consola conocida no se produjo hasta 1972, cuando la compañía estadounidense Philips desarrollo un primer modelo nombrado como *The brown box*, cuyo nombre acabo siendo cambiado por *Magnavox Odyssey*. Este lanzamiento produjo una cantidad de beneficios impresionante, llegando a vender un conjunto de 300.000 consolas, a pesar de su corto tiempo de vida en el mercado de solo tres años. Su lanzamiento en España no se produjo hasta dos años después, en pleno 1974, y vino acompañado por el renombre *Overkai*. Esta consola contó con un catálogo de hasta 28 juegos disponibles, incluyendo uno de los juegos que más revoluciono la industria en su momento, el famoso **Pong** basado en el deporte del tenis de mesa, que buscaba simular la experiencia de participar en una partida de este ejercicio.

Además, en 1972 se fundó la primera compañía en llevar el concepto de los videojuegos a una escala mayor, **Atari**. Centrada en el desarrollo de videojuegos *arcade*, consiguió un éxito rotundo durante un periodo de casi diez años, hasta que un conjunto de diversas compañías empezaron a desarrollar videojuegos, dando fin al monopolio establecido por Atari.

Adicionalmente, fue esta misma Atari quien comenzó a popularizar el videojuego como un pasatiempo al mismo que los deportes o la lectura, mediante el lanzamiento de la *Atari VCS* en 1977. Conocida también por el nombre de *Atari 2600*, salió al mercado con un conjunto de 10 videojuegos incluidos en la propia consola. Es de notable mención añadir que fue quien inicio el concepto de los cartuchos *ROM*, los cuales contenían videojuegos adicionales que podían ser injertados directamente en la consola desde una ranura, práctica que será apropiada para la reproducción de nuevos modelos en el futuro. A pesar de todas estas revoluciones, la consola tuvo unas ventas decepcionantes, muy por debajo de lo esperado. Sin embargo, en 1980 se produjo el lanzamiento del videojuego **Space Invader**, un fenómeno de masas ya popular en su versión *arcade*, que provoco un resurgimiento en la popularidad de la consola, incrementando sus ventas considerablemente.

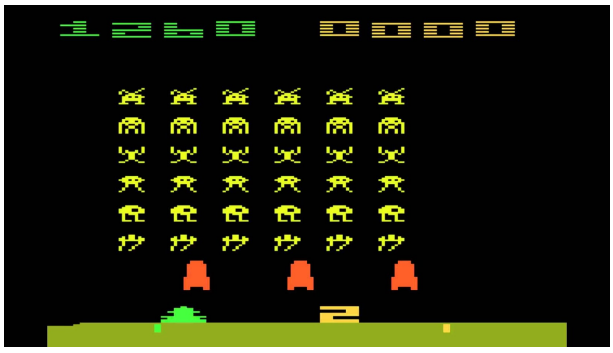

Figura 2.2: Space Invaders (1980)

2.2 Importancia de los videojuegos en el mundo contemporáneo

Durante estas últimas décadas, la popularidad de los videojuegos fue aumentando gradualmente como se muestra en el siguiente libro [4], especialmente en la época de los años 2000-2010 con la salida al mercado de la *Play Station 2*, desarrollada por **Sony Computer Entertainment**. Actualmente, es la consola más vendida de toda la historia, con un total de 155 millones de unidades vendidas, debido especialmente a su gran catálogo, con casi 3870 títulos de una gran variedad de géneros que han definido una generación ente-

ra, como **Shadow of the Colossus** con sus mecánicas revolucionarias de enfrentamiento contra enemigos o **Resident Evil 4** popularizando los videojuegos de terror y acción.

Figura 2.3: Shadow of the Colossus (2006) y Resident Evil 4 (2005)

Sin embargo, uno de los hechos que más ha mostrado la importancia que presenta esta industria en la población actual ha ocurrido a partir del periodo de pandemia y confinamiento durante 2020. Al inicio de los primeros meses de esta epidemia, la sociedad actual veía como sus capacidades sociales se limitaban. Por tanto, era imprescindible que se formaran nuevas formas de comunicación y de contacto con otros seres humanos. Debido a ello, los videojuegos, y especialmente aquellos con capacidad para jugar en línea, fueron un medio para poder interactuar con una gran variedad de personas de todo el mundo, ya sea por medio de partidas multijugador o incluso debido a las interacciones mediante plataformas como **Steam**.

Adicionalmente, a partir de un estudio cuya realización fue hecha por la propia universidad de **Oxford** [5] durante el segundo periodo de confinamiento establecido por el gobierno del Reino Unido, se observó que las experiencias competitivas y de conexión social provistas por los videojuegos tomados como muestra por el departamento de estudio contribuían al bienestar y estabilidad mental de todos aquellos usuarios que jugaban a este tipo de juegos, dotando de una mejora en la propia salud y estabilidad mental.

Dicho esto, se han realizado una gran variedad de estudios sobre cómo afectan los videojuegos a la conducta y sanidad de la población, acompañados muchos de ellos por diversas polémicas exaltando que intensifican la agresividad y las prácticas violentas entre los más jóvenes. A pesar de ello, se ha demostrado continuamente que su uso moderado provoca un aumento en ciertas capacidades humanas, como los reflejos o incluso el propio aprendizaje, como demuestra el siguiente estudio [6].

2.3 El sigilo en los videojuegos

Los videojuegos cuyo género principal es el sigilo se basan en proveer opciones al jugar para evitar a los enemigos o incluso atacarlos sin que se den cuenta. El incenti-

vo principal de la mayoría de videojuegos de este estilo es enfrentar al jugador contra enemigos muy superiores a él, por lo cual es forzado a utilizar otras herramientas. Normalmente, al usuario se le ofrece la opción de poder ocultarse en diversos elementos del escenario, agacharse, poder utilizar conjuntos de vestimentas o incluso utilizar la iluminación de los escenarios para evitar ser detectado por los enemigos. La mayoría de estas características anteriormente mencionadas requieren de un preciso y detallado diseño de niveles y dotar a los enemigos de inteligencias artificiales avanzadas para poder tener oportunidad contra las elecciones que dispone el jugador.

El primer juego centrado en estas características que aportan el elemento de sigilo fue **Shoplifting Boy**, publicado en 1979. Su enfoque principal consistía en desvalijar una tienda sin atraer la atención de los enemigos, evitando caer en las manos de la policía. Dos años más tarde, fue lanzado **Castle Wolfenstein**, cuyo objetivo consiste en atravesar un castillo poblado por grupos de vigilantes, intentando conseguir informes ocultos de los enemigos y escapar sin ser detectado. Sin embargo, uno de los videojuegos más famosos de este género fue el primer **Metal Gear**, desarrollado por la compañía **Konami** en 1987 y dirigido por Hideo Kojima, uno de los directores de videojuegos más prestigiosos de esta industria incluso actualmente.

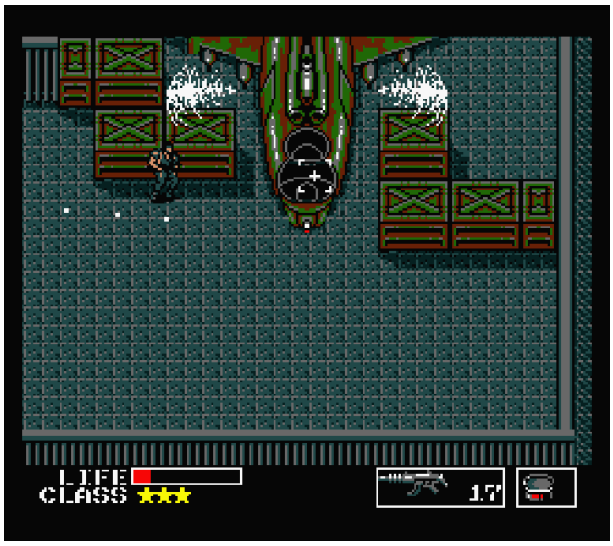

Figura 2.4: Metal Gear (1987)

Otro de los referentes importantes más actuales es la saga **Assasin's Creed**, con más de 11 lanzamientos principales desarrollados por **Ubisoft**, además de contar con diversos *spin-offs* e incluso libros y películas. Asimismo, es de notable mención la franquicia de videojuegos **Metal Gear Solid**, evolución natural de la saga mencionada anteriormente, introdujo elementos propios del cine en forma de secuencia de vídeo con un estilo muy similar al de una película e incluso aportaba tramas con contenido político y reflexivo sobre la guerra y sus consecuencias en la población y cultura.

Por supuesto, este género ha ido progresando y aumentando en importancia conforme ha pasado tiempo, incluso la mayoría de videojuegos suelen ofrecer elementos de sigilo, aunque no estén centrados en ello.

Figura 2.5: Castle Wolfenstein (1981) y Assasin's Creed 2 (2009)

2.4 Videojuegos en línea

Se nombran videojuegos en línea a todos aquellos que contienen una correlación entre dos o más jugadores al unísono por medio de una conexión a Internet o mediante el uso de una red local. En los inicios de esta industria, la principal forma de compartir una experiencia con otros jugadores se realizaba mediante la misma pantalla, ya sea en una casa o en un salón de máquinas recreativas. Sin embargo, durante el año 1973 se lanzó al mercado un producto conocido como *Empire*, consistiendo en un juego de estrategia por turnos donde se permitía la participación de hasta 8 jugadores, desarrollado para el sistema red **PLATO** [7].

Durante ese mismo año, Jim Bowery estreno **Spasim**, un juego con temática espacial permitiendo un total de hasta 32 jugadores que también funcionaba mediante PLATO. Se le considera como el primer videojuego multijugador formado en un entorno 3D, siendo el precursor de la mayoría de productos de este estilo en la actualidad.

En la época del año 1979, comenzaron a desarrollarse forma para que un conjunto de ordenadores se conectarán a partir de redes telemáticas, dando pasó a la creación de videojuegos conocidos como MUD (Multi-user dungeon). MUD fue concebido como una versión digital del popular juego de rol D&D (Dungeons and Dragons), que dieron paso a la creación de los primeros foros de comunidades *online*.

A partir del año 1990, se propagó de forma extraordinaria uno de los géneros de videojuegos en línea que más importancia han tenido durante la historia, los MMORPG's (Massively multi-player online role-playing video games). La principal idea que proporcionan este género, es la de adoptar un papel en un mundo conformado por otros jugadores como tú, siendo la evolución de los videojuegos MUD anteriormente mencionados. Algunos de los juegos más importantes lanzados durante esa década serían **Nexus: The Kingdom of the Winds** (1996), **Ultima Online** (1997), **Lineage** (1998), and **EverQuest** (1999). A pesar de ello, no fue hasta 2004 cuando se lanzó el MMORPG más conocido y famoso actualmente, **World of Warcraft**. Este fenómeno de masas consiguió contener más de 10 millones de suscripciones a su modelo pago mensual, y ha influido en la creación de múltiples videojuegos que contienen el mismo estilo como **Guild Wars 2**, **EVE Online** o **Black Desert Online**.

En estos años 2000, fue cuando se produjo uno de estilos de juegos en línea que más ha influido en estos ultimos años, conocidos como **MOBA** (Multiplayer Online Battle Arena). Sus inicios se remontan con la creación de **Defense of the Ancientes** o abreviado DOTA en 2003, basado en un modo de juego perteneciente a **Warcraft III**. A partir de ello, logró convertirse en uno de los géneros más conocidos en la actualidad, mediante productos como **League of Legends** (2010).

Figura 2.6: Guild Wars 2 (2012) y EverQuest (1999)

2.5 Motores utilizados

En la siguiente sección se mostrarán algunos de los motores presentes en desarrollo de videojuegos. Cada uno de estos motores presenta diferentes características, y a la hora de su elección, dependen muchos factores, como el equipo de desarrollo, el género de juego que estemos realizando o incluso el presupuesto.

- **Unreal Engine:** Desarrollado por **Epic Games**, es uno de los motores con más renombre en la industria actual, incluso es utilizado en otros medios, como el cine y la televisión. Su primera versión se estrenó con el lanzamiento del videojuego con el mismo nombre, **Unreal**, en 1998, un *shooter* en primera persona. Desde entonces, ha sido adaptado a una gran variedad de juegos en distintas consolas, géneros, móviles e incluso dispositivos centrados en la realidad virtual.

  Su última versión, **Unreal Engine 5**, presenta un conjunto de nuevas características a tener en cuenta para realizar mundos con una increíble fidelidad y calidad gráfica:

  - **Lumen**: Sistema dinámico de iluminación global que adapta la iluminación indirecta dependiendo de la geometría o fuente de luz, creando *efectos como sangrado de color*
  - **Nanite**: Sistema de virtualización de geometría micro-poligonal que permite la creación de elementos con gran cantidad de detalle sin perder rendimiento.
  - **Virtual Shadow Maps**: Método de estructuración de sombras centrado en proporcionar alta calidad y resolución a partir de un rendimiento estable.

- **Unity:** Uno de los motores de videojuegos más conocidos y utilizados, especialmente en proyectos *Indie*. Disponible para una gran cantidad de plataformas y entornos, ha desarrollado algunos de los videojuegos más queridos por la audiencia, como **Hollow Knight** o **Cuphead**. Lanzado en 2005, ha ido evolucionando cada año introduciendo nuevas herramientas, que van desde motores de físicas 2D/3D hasta entornos de soporte de realidad virtual y desarrollo red para videojuegos multijugador. Más adelante profundizaremos en más aspectos de Unity, al ser el motor utilizado para el desarrollo de este proyecto.

- **Motores propios:** Una gran cantidad de empresas, especialmente aquellas con un presupuesto mayor, deciden apostar por el desarrollo y uso de su propio motor, garantizando diversas ventajas. En primer lugar, se reducen el coste total de proyectos, al no ser necesario el pago de licencias a la propietaria de los motores. Además, permite ajustar el motor a las necesidades del juego, mejorando así las capacidades del propio sistema. Algunos de los motores propios más reconocidos son Frostbite de EA Digital Illusions CE, Decima de Guerrilla Games o Anvil de Ubisoft.

Adicionalmente, también existen otros diversos motores utilizados en ambientes más independientes, especializados en diversos nichos o que buscan facilitar su uso a desarrolladores con menos experiencia. Entre algunos de estos motores, podemos encontrar **RPG Maker**, especializado en el desarrollo de videojuegos del género *RPG* o incluso experiencias interactivas como el aclamado **To the Moon**, **Rem'Py**, centrado en el desarrollo de videojuegos del género *novela visual*, de código abierto y gratuito construido mediante Python, o Scratch, centrado en usuarios jóvenes con poca o nada experiencia en código, a partir de su estructura de programación por bloques.

2.6 Programas de diseño

Durante esta sección se mostrarán el conjunto de opciones que se han tenido en cuenta a la hora de diseñar los elementos artísticos presentes en el videojuego. Existe una gran multitud de opciones en el mercado, ya que cada programa ofrece sus propias características y ventajas, ya sea su precio, herramientas que presenta incluso portabilidad a diferentes formatos de imágenes. Entre ellos, se destacan los siguientes:

- **GIMP:** Es una aplicación de distribución libre focalizada en tareas de edición, creación y composición de imágenes. Presenta un conjunto de extensiones disponibles para su inserción que añade nuevas funcionalidades y opciones, ofreciendo una gran variedad de elecciones al usuario. GIMP se encuentra disponible en las principales plataformas como *Windows*, *Mac* y *Linux*, y desde su lanzamiento en 1995 ha recibido continuas actualizaciones y ofrece un gran conjunto de documentaciones en múltiples lenguajes.
- **GraphicsGale:** Editor de gráficos e imágenes píxeles desarrollado exclusivamente para *Windows* y creado por *Humanbalance Ltd*. Destaca por su facilidad de uso y grandes opciones que permiten realizar conjuntos de animaciones en múltiples archivos. Anteriormente, era de pago, pero desde 2017 fue ofrecido como una aplicación gratuita y es recomendada por multitud de usuarios.
- **Aseprite:** Herramienta centrada en el desarrollo *pixel art* para la creación de animaciones 2D. Ofrece un conjunto de herramientas enorme y exportación a diversos formatos de imágenes, además de presentar una interfaz sencilla pero muy intuitiva. Asimismo, contiene una gran comunidad de usuarios que ofrecen documentación y tutoriales sobre las diversas técnicas que se pueden aplicar para la creación de imágenes. Sin embargo, solo presenta una única versión de pago disponible.

CAPÍTULO 3
Análisis del problema

En este apartado se identificarán los diferentes aspectos necesarios para nuestro proyecto. Comenzaremos definiendo los casos de uso que nos ayudaran a desarrollar a continuación los requisitos necesarios. A continuación, expondremos los diferentes requisitos funcionales y no funcionales. Posteriormente, daremos una explicación sobre la idea en la que se basa el videojuego. Por último, expondremos el plan de trabajo, definiendo como hemos aplicado la metodología de trabajo.

3.1 Casos de uso

Se han realizado un conjunto de diversos casos de uso para definir los distintos elementos presentes en el juego, los cuales se expondrán a continuación.

La figura [3.1] muestra el conjunto de acciones que el usuario puede realizar en el menú principal. Seguidamente, se procederá a explicar en detalle las distintas opciones:

- **Nueva partida** : El usuario comienza una partida desde el principio. Pulsar esta acción, eliminará todos los datos almacenados anteriormente.
- **Continuar** : El usuario comienza desde el último nivel guardado. Si no contiene datos anteriores, se iniciará una nueva partida desde el principio.
- **Salir** : El usuario desconecta la aplicación.
- **Opciones** : El usuario entra en el apartado de opciones. Desde aquí, es la única forma para acceder al botón **Ajustar volumen**, mediante el cual se puede nivelar el volumen general de la aplicación

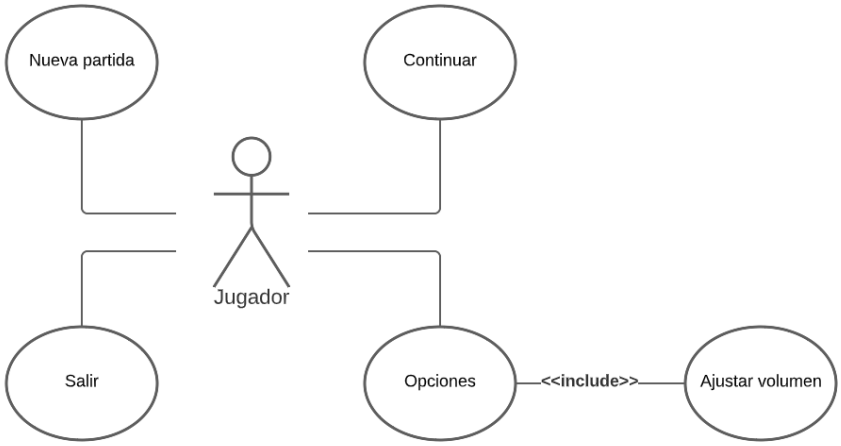

Figura 3.1: Caso de uso Menú principal

La figura [3.2] muestra el conjunto de acciones que el usuario puede realizar en el menú de pausa. Posteriormente, se procederá a explicar en detalle las distintas opciones:

- **Continuar** : El usuario reanuda la partida, cerrando el menú de pausa.
- **Opciones** : El usuario entra en el apartado de opciones. Desde aquí, es la única forma para acceder al botón **Ajustar volumen**, mediante el cual se puede nivelar el volumen general de la aplicación
- **Salir** : El usuario vuelve al menú principal

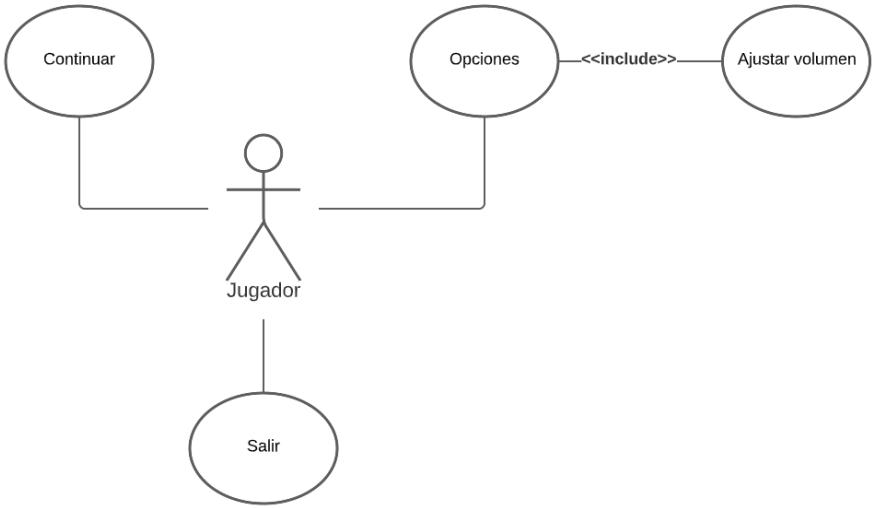

Figura 3.2: Caso de uso Menú pausa

La figura [3.3] muestra el conjunto de acciones que el protagonista puede realizar. A continuación, se procederá a explicar en detalle las distintas opciones:

- **Moverse** : El protagonista se desplaza por el nivel
- **Trepar** : Al encontrarse cerca de una escalera, el protagonista puede trepar sobre ellas
- **Embestir** : El protagonista realiza un desplazamiento veloz, embistiendo a los enemigos a su paso.
- **Agacharse** : El protagonista reduce su movimiento encogiéndose, reduciendo el rango de visión de los enemigos.
- **Recoger objeto** : Si el protagonista se encuentra cerca de un objeto, se le añadirá a su inventario.
- **Usar objeto** : Si el objeto se encuentra seleccionado en su inventario, este podrá ser utilizado. El efecto variará dependiendo de que tenga equipado.
- **Esconder** : Si el protagonista se encuentra cerca de una cortina, podrá ocultarse en ella.
- **Interactuar** : El protagonista podrá interactuar con diversos elementos del escenario que muestren un elemento visual al estar cerca.

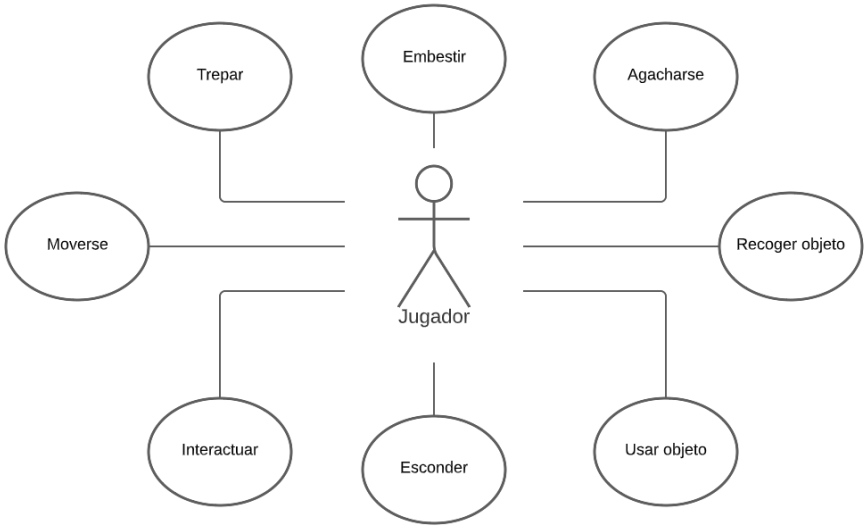

Figura 3.3: Caso de uso Protagonista

La figura [3.4] muestra el conjunto de acciones que el protagonista puede realizar. Seguidamente se procederá a explicar en detalle las distintas opciones:

- **Perseguir** : El enemigo se desplaza en dirección al protagonista
- **Morir** : Se produce cuando la vida del enemigo se reduce a cero.
- **Patrullar** : El enemigo se desplaza por el mapa.
- **Quieto** : El enemigo se queda en estado inmóvil.
- **Atacar** : El enemigo realiza un ataque si se encuentra a una distancia cercana del protagonista.
- **Investigar** : El enemigo se dirige hacia la localización donde se encuentra el objeto a investigar.

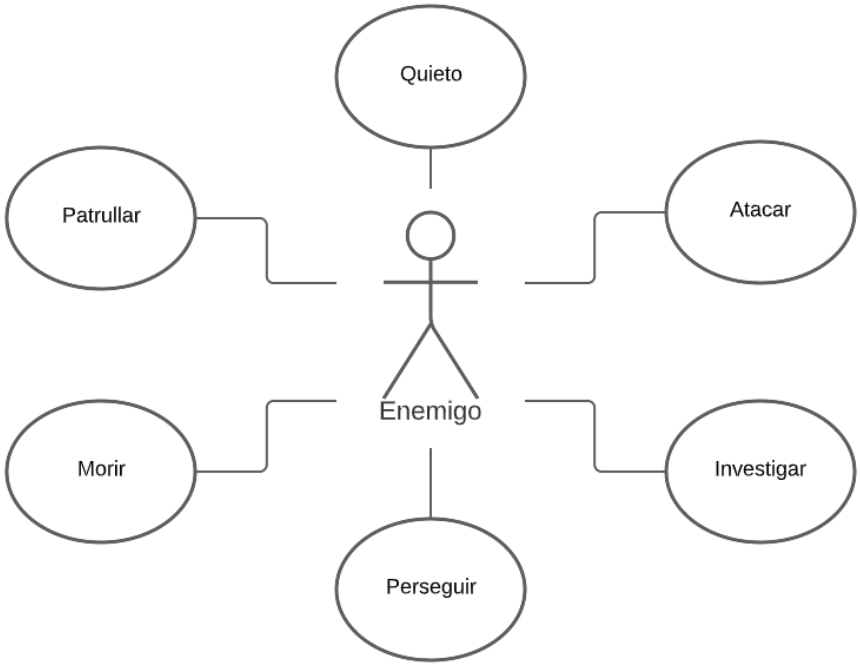

Figura 3.4: Caso de uso Enemigo

3.2 Especificación de requisitos

En esta sección, procederemos a hablar sobre los requisitos presentes en el proyecto. Cada requisito se caracterizarán por contener un identificador, un nombre y una descripción.

Requisitos funcionales

Las tablas que se presentan a continuación muestran los requisitos funcionales presentes en la aplicación

Rf1	
Nombre	Nueva Partida
Descripción	El jugador debe poder empezar una nueva partida siempre que quiera, generando nuevos datos

Rf2	
Nombre	Continuar partida
Descripción	Si el jugador ha completado un nivel, podrá empezar de nuevo desde el último nivel que haya finalizado, manteniendo su progreso

Rf3	
Nombre	Volver al menú
Descripción	El jugador podrá en cualquier momento volver al menú principal desde el menú de pausa, una vez finalizado el nivel, o al morir

Rf4	
Nombre	Salir del juego
Descripción	El jugador podrá salir de la aplicación a través del menú principal

Rf5	
Nombre	Menú de pausa
Descripción	Durante un nivel, el jugador podrá acceder al menú de pausa en cualquier instante

Rf6	
Nombre	Morir
Descripción	En el caso en el que la vida del protagonista descienda a 0, el juego pausará y se mostrará el menú de muerte

Rf7	
Nombre	Recoger objeto
Descripción	Cuando el protagonista se encuentre en el rango de un objeto, se añadirá al inventario

Rf8	
Nombre	Usar objeto
Descripción	El jugador podrá utilizar un objeto mientras esté presente en su inventario

Rf9	
Nombre	Atacar
Descripción	El jugador podrá realizar un ataque mientras tenga el objeto *Espada* equipado desde su inventario

Rf10	
Nombre	Embestida
Descripción	El jugador podrá realizar una embestida, siempre que no esté en *tiempo de enfriamiento*

Rf11	
Nombre	Abrir cofre
Descripción	Si el jugador se encuentra en una posición cercana a un cofre, podrá ser abierto

Rf12	
Nombre	Interactuar con palancas
Descripción	Si el jugador se encuentra en una posición cercana a una palanca, podrá activarla

Rf13	
Nombre	Siguiente nivel
Descripción	El sistema debe permitir que el jugador pueda acceder al siguiente nivel una vez haya completado el nivel en el que se encuentre.

Rf14	
Nombre	Guardar progreso
Descripción	Una vez un nivel haya sido completado, el sistema preservará los objetos que tenga en ese momento en el inventario para el siguiente nivel

Rf15	
Nombre	Sonidos
Descripción	El sistema reproducirá los archivos de audio asignados a cada elemento que cuente con un sonido representativo

Rf16	
Nombre	Movimiento
Descripción	El sistema debe permitir que tanto el protagonista como los enemigos puedas moverse por el escenario

Rf17	
Nombre	Mostrar rango de lanzamiento
Descripción	El sistema mostrará el rango de lanzamiento de aquellos objetos que puedan ser lanzados

Requisitos no funcionales

Las tablas que se presentan a continuación muestran los requisitos no funcionales presentes en la aplicación, basándonos en el modelo ISO/IEC 25010 [8]:

Rnf1	
Nombre	Corrección funcional
Descripción	La aplicación debería ser capaz de funcionar en cualquier ordenador

Rnf2	
Nombre	Accesibilidad
Descripción	Las mecánicas del videojuego serán sencillas, ideales para usuarios menos experimentados

Rnf3	
Nombre	Disponibilidad
Descripción	La aplicación se encontrará en funcionamiento mientras el jugador no cierre su ejecución

3.3 Explicación del videojuego

El concepto principal consiste en el usuario, tomando el control del protagonista, debe superar un conjunto de niveles, llegando hasta la sección final de cada uno de ellos. En cada nivel, aparecerán diversos enemigos, cuya principal función es detener al protagonista, buscando reducir su vida a cero. Los enemigos tendrán un rango de visión, con los cuales detectarán al protagonista y se dirigirán hacia él para atacarle. La mayoría de enemigos se encontrarán patrullando por el mapa, bloqueando el acceso del camino al jugador.

La perspectiva del videojuego se muestra como un juego de plataformas en 2D, con una visión limitada de un parte del nivel, cuya cámara va desplazándose junto con el protagonista.

El videojuego se estructura en tres niveles que irán incrementando en tamaño y dificultad. En el último nivel, el jugador deberá enfrentarse a un Jefe final en un área cerrada del mapa para poder completar el juego. Durante cada escenario, el usuario tendrá a su disposición un conjunto de objetos con los cuales podrá derrotar enemigos y resolver diversos puzzles.

Por último, existen elementos de sigilo que el jugador puede utilizar a su favor, como esconderse en cortinas, apagar antorchas o lanzar señuelos para atraer enemigos.

3.4 Plan de trabajo

En esta sección, procederemos a hablar de cómo hemos aplicado la metodología ágil Scrum para el desarrollo del proyecto, mencionada en apartados anteriores. El proceso ha sido dividido en un conjunto de tres Sprints, contando con un mes de duración para cada uno. Adicionalmente, se realizó un primer sprint de inicialización, cuyo proceso se definirá próximamente.

Para gestionar la planificación, hemos utilizado la plataforma de **Trello**. Trello es una aplicación centrada en la administración de proyectos, cuyo contenido está organizado por tableros. Cada tablero contiene un conjunto de listas, las cuales se encargan de definir el estado de las tarjetas, una funcionalidad que contiene diversos elementos como una descripción, fecha de realización, etiquetas para definirlas, entre otras opciones como se muestra en la figura [3.5].

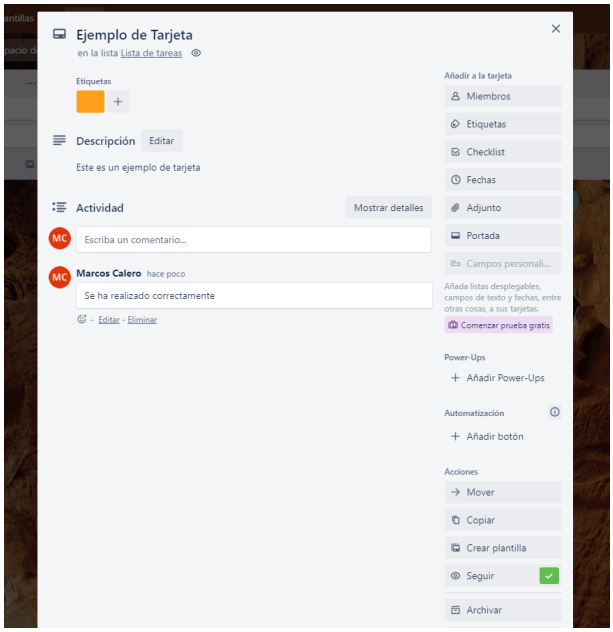

Figura 3.5: Ejemplo de tarjeta en Trello

En la figura [3.6], podemos observar como hemos llevado a cabo la distribución de las tareas mediante Trello. Cada persona es asignada a una tarea, y en ella marca con diversas etiquetas como se ha producido el desarrollo:

- **Verde**: La tarea se ha completado y ha pasado todas las pruebas
- **Amarillo**: La tarea se ha completado, habiéndose producido cambios debido a las pruebas
- **Naranja**: La tarea está en desarrollo
- **Rojo**: Se ha producido un error en las pruebas

Aquellas tareas que se han llegado a realizar o han sido descartadas se encuentran presentes en el Backlog.

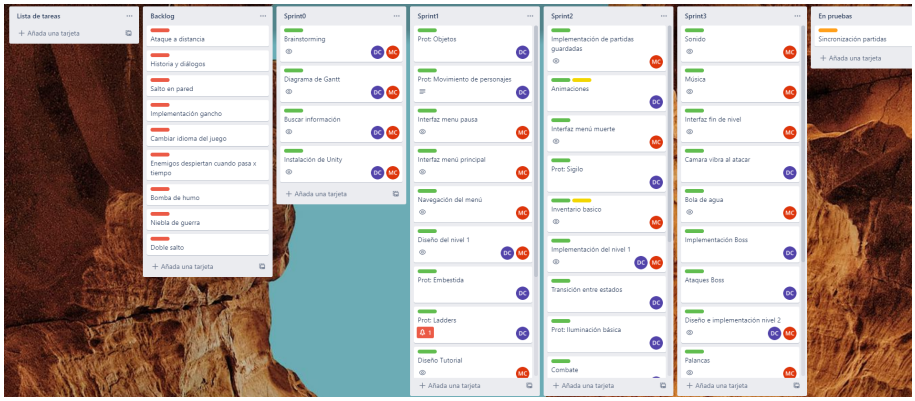

Figura 3.6: Sprints en el desarrollo del proyecto

En la figura [3.7], podemos observar cómo se ha producido la planificación establecida para la realización del proyecto, mediante la realización de un diagrama de Gantt. Cada actividad está representada conforme a los tiempos reales en las que se llevaron a cabo, incluyendo el Sprint donde se desarrollaron. Se han omitido aquellas actividades cuya realización no se produjo durante el periodo de desarrollo o fueran descartadas debido a varios factores, como es el caso de la tarea Implementación del gancho, cuya puesta en marcha fue cancelada debido al surgimiento de incoherencias con respecto al diseño definitivo de los niveles establecidos.

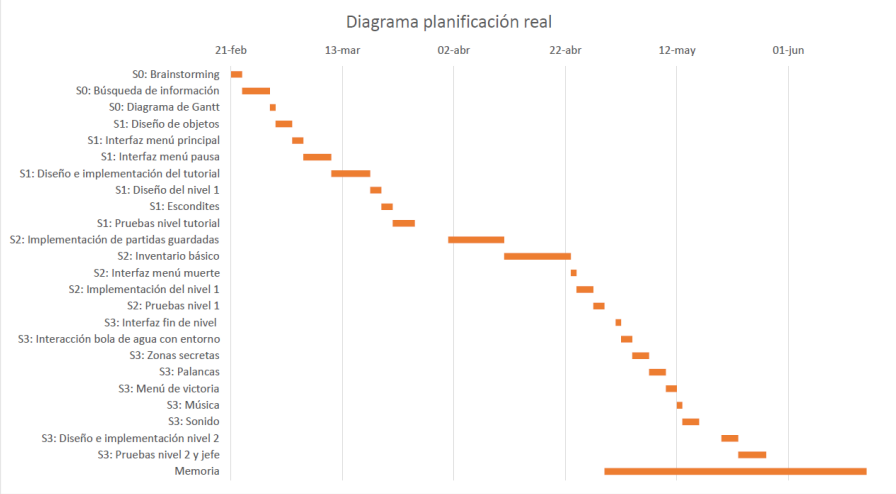

Figura 3.7: Planificación del proyecto

Sprint 0

Este sprint suplementario comprendió una duración de unas 2 semanas que se enfocaron en la realización de técnicas de planificación. Como primera actividad, se organizó un conjunto de sesiones de Brainstorming mediante las cuales se presentaron una gran variedad de ideas para el desarrollo y planificación.

A continuación, se llevó a cabo la creación de un *Diagrama de Gantt* cuyo propósito es realizar la estructura del proceso de especificación y duración del conjunto de actividades a realizar.

Por último, se organizaron el conjunto de diferentes UTs perteneciente al proyecto y se especificó como se repartirían entre cada Sprint, añadiendo sus pruebas de aceptación correspondientes.

Sprint 1

Durante este primer sprint se centró en la creación de escenas prototipo cuya finalidad buscaba experimentar con las diferentes mecánicas a desarrollar que presentaría la aplicación.

Inicialmente, se diseñaron las interfaces que debería contener el menú principal y el menú de pausa mediante el uso de bocetos y modelos de uso, concretando su navegación y que elementos tendría que presentar para más tarde ser aplicadas a partir de Unity.

Más adelante, se procedió a recolectar diversos recursos útiles para el desarrollo del videojuego, como pueden ser modelos y animaciones. Además, se dio pie a la creación de un conjunto de *sprites* necesarios para la aplicación, como los objetos, las interfaces y componentes del escenario.

Ulteriormente, tomó lugar la puesta en marcha de las interacciones básicas con el entorno, estableciendo los escondites por medio de las cortinas con el fin de ocultarse.

A continuación, se procedió con el diseño del nivel de tutorial con el fin de adoctrinar al usuario con las funcionalidades básicas, realizando modelados y esquemas a partir de herramientas de diseño. Posteriormente, se realizó la implementación de estos bocetos a partir de los recursos mencionados anteriormente y añadiendo las mecánicas básicas.

Durante el mismo periodo de tiempo, se aplicaron las primeras formas de movimiento disponibles para el protagonista, en conjunto con el desarrollo de la funcionalidad de los primeros objetos y el diseño de las máquinas de estado iniciales.

Sprint 2

El desarrollo del segundo sprint fue focalizado en la puesta en marcha a la creación del sistema de almacenamiento de partidas guardadas, con el objetivo de que el jugador pueda preservar sus avances en una sesión por medio de un archivo Json exterior al juego.

Posteriormente, se inicializó la creación del sistema de inventario con el cual el jugador podrá recolectar objetos y usarlos durante la partida. Durante este proceso, se produjeron diversos problemas que dieron paso a una ampliación del tiempo estimado para la tarea, debido a incompatibilidades con el sistema de guardado al que hemos hecho referencia anteriormente. Además, se desarrollaron con más detalle los modelos de cada objeto.

Esta etapa permitió también el esbozo e implementación del primer nivel del juego, el cual cuenta con un mapa más extenso y abundante de enemigos, ofreciendo un desafío mayor.

Adicionalmente, fueron aplicadas los comportamientos presentes en las máquinas de estado de los enemigos, dando lugar a su conjunto de comportamientos con el escenario y el jugador, incluyendo el propio combate. Por otra parte, también se introdujeron elementos de iluminación por medio de las antorchas existentes en los niveles.

Sprint 3

Al inicio de este último sprint se comenzaron a implementar las últimas mecánicas presentes en el escenario para que el jugador pudiera interactuar de una mayor forma con el entorno que presenta. En primer lugar, se diseñaron un conjunto de zonas secreteas, invisibles al jugador, que servirán como atajos. Dichas zonas deberán presentar una forma de poder ser reveladas para el usuario, por lo cual se aplicó un conjunto de palancas que pudieran comunicar con dichos pasajes. De la misma forma, estas palancas fueron aplicadas en otros puntos clave de los niveles, permitiendo la obertura de puertas bloqueadas inicialmente, de tal forma que sirvan como bloqueo al protagonista o incluso a los enemigos.

Con estos elementos en mente, se dio paso al planteamiento y desarrollo de lo que sería el último nivel presente en el videojuego, centrado en proporcionar un desafío final mediante múltiples obstáculos e interacciones, culminando en un enfrentamiento final contra un jefe en un escenario cerrado.

A continuación, se dio paso a la creación de las últimas interfaces que formarán parte del juego, proporcionando un menú de victoria al completar el nivel y uno adicional cuando el protagonista consigue derrotar al enemigo final.

A fin de suministrar una sensación de juego más agradable, se dio paso a la inserción de elementos sonoros a ciertos elementos de la aplicación, como pueden ser el protagonista y los enemigos. Además, también se proporcionó música de fondo para el menú principal y el conjunto de niveles a superar.

En la siguiente tabla [3.1] se mostrarán las tareas asociadas a cada miembro del proyecto:

	Daniel de Castro	Marcos Calero
Sprint 0		
Brainstorming	X	X
Búsqueda de información	X	X
Diagrama de Gantt	X	X
Sprint 1		
Diseño de objetos	X	
Prototipo movimiento del protagonista	X	
Prototipo de objetos	X	
Interfaz menú principal		X
Interfaz menú de pausa		X
Diseño e implementación del nivel de tutorial	X	X
Diseño de nivel 1	X	X
Diseño de máquinas de estados de enemigos	X	
Pruebas de nivel de tutorial	X	X
Sprint 2		
Implementación de partidas guardadas		X
Animaciones	X	
Prototipo de iluminación básica	X	
Prototipo elementos de sigilo	X	
Inventario		X
Interfaz menú de muerte		X
Prototipo de *Pathfinding*	X	
Implementación de máquinas de estados: Estados	X	
Implementación nivel 1	X	X
Pruebas nivel 1	X	X
Sprint 3		
Retoques a cámara	X	
Interfaz de fin de nivel		X
Implementación del jefe final	X	
Ataques jefe final	X	
Interacción bola de agua con el entorno		X
Zonas secretas		X
Palancas		X
Interfaz del menú victoria		X
Música y sonido		X
Diseño e implementación del nivel 2	X	X
Pruebas del nivel 2 y jefe final	X	X

Tabla 3.1: Reparto de tareas

CAPÍTULO 4
Diseño de la solución

Durante este capítulo, se mostrarán cada uno de los componentes que forman parte del videojuego, además de como interactúan entre ellos mismos. Debido a que este proyecto se ha realizado en conjunto, nos centraremos en aquellos pertenecientes a este trabajo. Sin embargo, se explicarán todos los elementos del juego realizados por otros compañeros de forma breve.

4.1 Tecnología utilizada: Unity

Como ha sido mencionado en apartados anteriores, escoger un motor de videojuegos para realizar un desarrollo de un juego es un parte muy importante, ya que puede condicionar las características que el propio proyecto podrá presentar, debido a las limitaciones que surgen al escoger un motor concreto. En el mercado actual existe una gran variedad de motores, por tanto, la decisión de cuál utilizar para este proyecto fue larga y tendida, debido a que se debía considerar que tipo de juego realizar, la propia experiencia conforme a conocimientos previos existentes y diversos estudios que comparan diferentes herramientas como en el siguiente estudio [9]. En vista de estos puntos mencionados, se acabó escogiendo Unity, por lo cual se explicarán a continuación los motivos de su elección.

Primeramente, durante el semestre anterior se estuvo desarrollando un proyecto con la misma base de desarrollo en equipo a partir del uso de Unity, de tal manera que se pudo obtener los conocimientos básicos y necesarios de como poder abarcar una aplicación mediante esta tecnología, aumentando así las posibilidades de éxito del producto y siendo capaces de poder trabajar a una mayor rapidez. Además, Unity utiliza como lenguaje de programación C# [10], el cual ha sido estudiado durante cada curso durante la carrera, ofreciendo un entorno familiar para su utilización.

Frente a otras opciones, el modelo principal que ofrece Unity es gratuito, disminuyendo considerablemente el coste que podría suponer en adquirir una tecnología de pago, ya que ofrece una gran cantidad de opciones de desarrollo pese a ser una versión libre. Sin embargo, si se desea obtener beneficios considerables con el proyecto, sí que se hubiera tenido que comprar la versión de pago.

Al ser una de las herramientas de más utilizadas por la comunidad de desarrollo de videojuegos, presenta un gran conjunto de tutoriales y documentación [11] esenciales para la creación de un proyecto, desde *Youtube* hasta *Stack Overflow* [12], son enormes las referencias que se pueden encontrar o incluso preguntar en lugares de contacto como foros dedicados al desarrollo de juegos.

Asimismo, Unity destaca entre los desarrolladores como la principal herramienta para la creación de proyecto de envergadura menor, como es el caso de este trabajo, ya que ofrece un sistema de exportación sencillo e intuitivo, asimismo el espacio que ocupan las versiones finales en los discos de almacenamiento de un ordenador suele ser menores que respecto a la competencia.

Por otra parte, Unity presenta un conjunto de funcionalidades que permiten desarrollar videojuegos 2D al igual que nuestro proyecto, destacando el uso de *Tilemaps* con el cual se facilita el diseño e implementación de niveles. Además, ofrece una gran variedad de *plugins* que permiten el uso de funcionalidades extra, como la introducción de un sistema de diálogos no presente en la versión base de Unity.

4.2 Tecnología utilizada: Aseprite

Entre la multitud de opciones mencionadas con anterioridad, se ha escogido Aseprite como la principal herramienta para el diseño de todos los elementos artísticos presentes en el videojuego que no hayan sido obtenidos por medio de *Unity Asset Store* [13]. La razón de esta elección se profundizará durante los párrafos posteriores.

Principalmente, Aseprite [14] es una de las herramientas más usadas para el desarrollo de videojuegos 2D. La cantidad de información presente en multitud de páginas y entornos web es enorme, además de existir una comunidad de usuarios que comparten sus creaciones de forma explicativa, para que puedan ser utilizadas como referencia.

Por último, esta herramienta había sido utilizada con anterioridad antes de la planificación de este proyecto, por lo tanto, se poseían los conocimientos básicos necesarios para su utilización. Dicho esto, Aseprite presenta una gran cantidad de herramientas similares a otros editores, por lo que su uso se siente muy familiar.

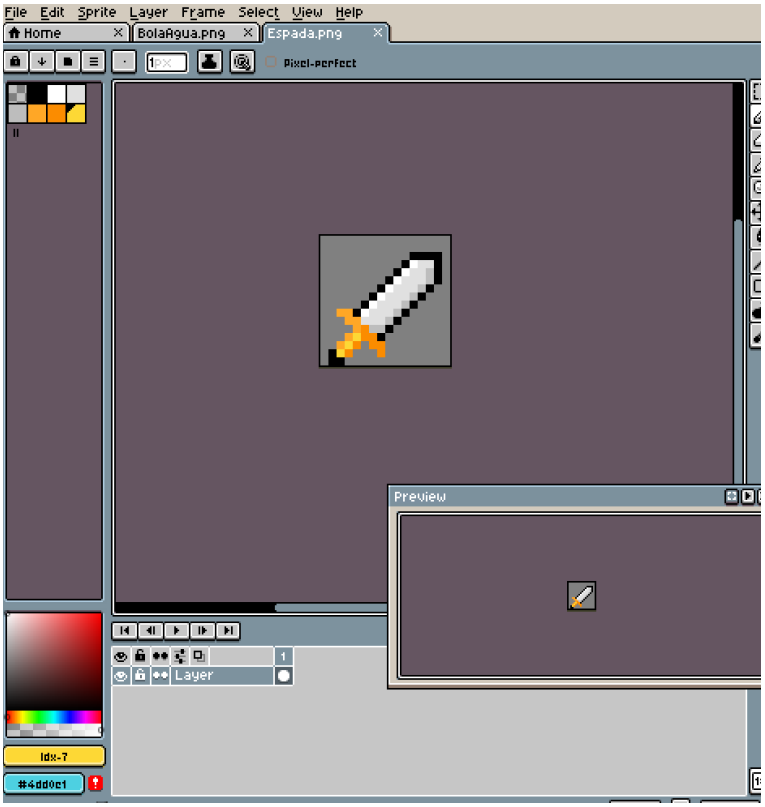

Figura 4.1: Interfaz de Aseprite

4.3 Interfaz

Las interfaces son una parte fundamental de cualquier videojuego, ya que permiten al jugador acceder a una gran cantidad de funcionalidades, ya sea adentrarse a los niveles que presenta el juego, modificar las opciones, pausar la partida o incluso salir del propio sistema. En este proyecto, se ha buscado exponer al jugador un conjunto de interfaces sencillas que permitan ofrecer las opciones básicas que presentan cada videojuego.

Al dar inicio a la aplicación, el jugador será presentado con el menú principal. A continuación, en la figura [4.2] se mostrarán las distintas transiciones que deberán estar presentes:

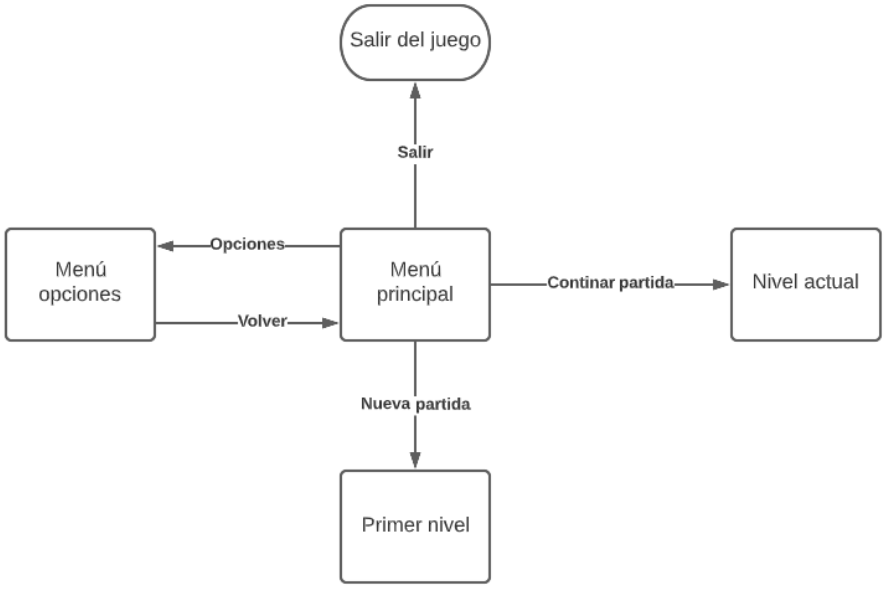

Figura 4.2: Navegación del menú principal

Menú principal	
Descripción	Interfaz mostrada al comenzar el videojuego
Transiciones	
Primer nivel	Pulsar el botón Nueva partida
Nivel actual	Pulsar el botón Continuar partida
Menú opciones	Pulsar el botón Opciones
Menú principal	Pulsar el botón Volver
Salir del juego	Pulsar el botón Salir

Tabla 4.1

Durante el transcurso de los niveles, el jugador podrá accionar una tecla para poder acceder al menú de pausa. A continuación, en la figura [4.3] se mostrarán las distintas transiciones que deberán estar presentes:

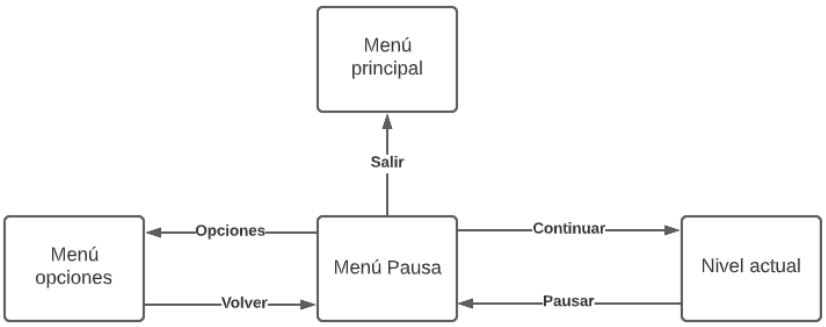

Figura 4.3: Navegación del menú de pausa

Menú pausa	
Descripción	Interfaz mostrada al pausar el videojuego
Transiciones	
Nivel actual	Pulsar el botón Continuar
Menú pausa	Accionar la tecla Escape
Menú opciones	Pulsar el botón Opciones
Menú pausa	Pulsar el botón Volver
Menú principal	Pulsar el botón Salir

Tabla 4.2

Al completar un nivel, se mostrará al jugador el menú de victoria. A continuación, en la figura [4.4] se mostrarán las distintas transiciones que deberán estar presentes:

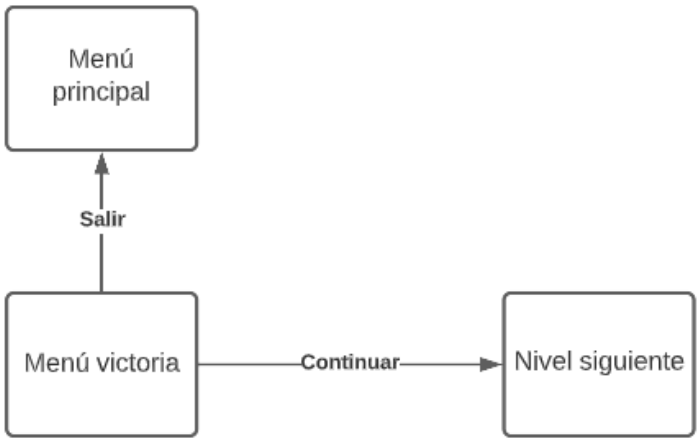

Figura 4.4: Navegación del menú de victoria

Menú victoria	
Descripción	Interfaz mostrada al completar un nivel
Transiciones	
Nivel siguiente	Pulsar el botón Continuar
Menú principal	Pulsar el botón Salir

Tabla 4.3

Si la vida del jugador se reduce a cero, se mostrará el menú de muerte. A continuación, en la figura [4.5] se mostrarán las distintas transiciones que deberán estar presentes:

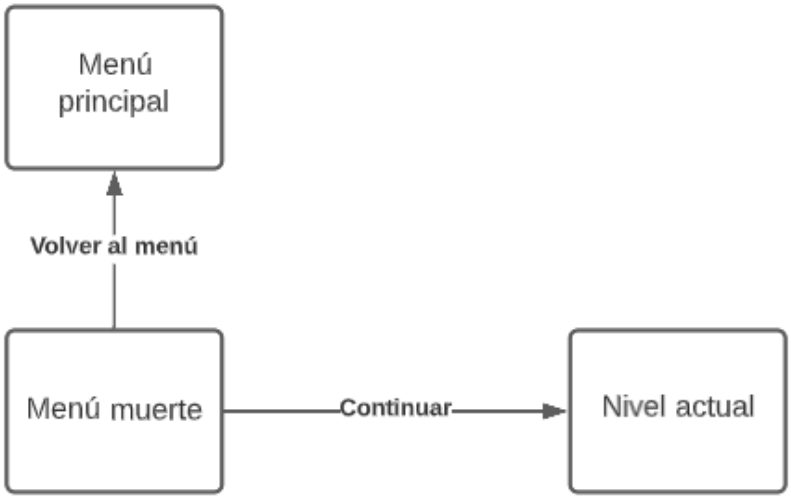

Figura 4.5: Navegación del menú de muerte

Menú muerte	
Descripción	Interfaz mostrada al reducir la vida del jugador a cero
Transiciones	
Nivel actual	Pulsar el botón Continuar
Menú principal	Pulsar el botón Volver al menú

Tabla 4.4

En ciertas secciones de la partida, aparecerán diálogos que detallarán explicaciones sobre los objetos, actuando como tutoriales. Deberán ser representadas mediante un cuadro de texto, localizado en la parte superior de la pantalla.

Cada objeto que el jugador obtenga se mostrará mediante una interfaz que representará el conjunto de huecos disponibles en el inventario, llegando a un máximo de 6 huecos en total.

4.4 Niveles

Cada uno de los diversos escenarios de los que se compone en proyecto han sido desarrollados mediante el sistema de cuadrículas perteneciente a Unity, el cual ha aliviado

la carga a la hora de editar y actualizar los niveles conforme se ha ido desarrollando el videojuego.

Este sistema se basa en el uso de una interfaz gráfica de edición, mediante la cual se puede elegir la posición de los *tiles* a lo largo de la escena. Cada uno de estos elementos, se pueden almacenar en diversas capas, las cuales permiten que pueda haber varios tipos de *tiles* en la misma posición, una opción que permite el desarrollo de niveles más diversos y complejos.

En cuanto al diseño, se ha optado por utilizar una estética que dotará a los escenarios de un ambiente de penumbra y oscuridad, la cual fortalece ciertos elementos jugables como la iluminación y los elementos de sigilo. Para llevar esto a cabo, se utilizó un conjunto de paquetes de *assets* disponibles desde la propia tienda de Unity, *Unity Asset Store*. Sin embargo, se han diseñado algunos elementos propios, como es el caso de las puertas y las palancas. La ambientación principal se centra en mostrar el interior de una oscura fortaleza, de ambiente decrépito y abandonada, la cual está poblada por diversas oleadas de enemigos viviendo allí desde hace años y que no dudaran en atacar a quien perturbe su estancia en aquel funesto lugar.

Durante el desarrollo del conjunto de niveles, decidimos dotar al usuario de la mayor cantidad de opciones posible para que él pudiera afrontar los retos que el videojuego le ofrece a su propio ritmo y estilo, resultando en una mayor satisfacción. De esta forma, el usuario presenta varias formas con las cuales podrá sobrepasar a sus enemigos, ya sea derrotándolos en combate con un mayor riesgo de poder ser eliminado, o distrayéndolos y evitando un enfrentamiento directo, resultando ser una opción más segura, pero a la vez más costosa de llevar a cabo.

4.5 Protagonista

El diseño del protagonista tuvo como intención principal ofrecer un aspecto acorde con la ambientación de los niveles, una apariencia de un explorador que pueda protegerse de todas aquellas dificultades que se presenten. Debido a esto, se buscó un modelo disponible en *Unity Asset Store* que contará con un conjunto de *sprites* que ofrezcan todas las mecánicas necesarias, aliviando la carga de diseño y así poder centrarse de forma más significativa en la implementación de las animaciones del personaje.

En cuanto al diseño jugable del personaje, se buscará ofrecer un gran conjunto de posibilidades que permitan al usuario afrontar los obstáculos como prefiera, combatiendo con los enemigos presentes en el videojuego o incluso poder evitarlos mediante diferentes tácticas. En la sección de casos de uso presentes en el apartado anterior, se muestran todas las acciones que el jugador podrá realizar. Algunas acciones estarán limitadas a ciertas condiciones, como ejemplo, solo se podrá escalar una plataforma si se encuentra cerca de un lugar que permita esta acción.

4.6 Enemigos y jefe

El inconveniente principal al que el protagonista deberá enfrentar será a los enemigos presentes en los escenarios, los cuales entorpecerán su avance a través de los niveles. Se buscaba ofrecer una estética consistente y semejante a la de los escenarios, por tanto, se ha escogido enemigos presentes en ambientes de fantasía medieval, como por ejemplo esqueletos vivientes.

Para afrontar la lógica que presentaran los diversos enemigos, se ha decidido utilizar una máquina de estados, diferenciado ente la de enemigos principales y el jefe final. Estas máquinas deberán cubrir todos los posibles comportamientos que contendrán, los cuales variarán respeto a diferentes posibilidades, ya sea el propio entorno o los propios actos de jugador. Seguidamente, se presentarán un conjunto de máquinas de estados diseñadas y se explicará a fondo cada uno de los estados que presentan.

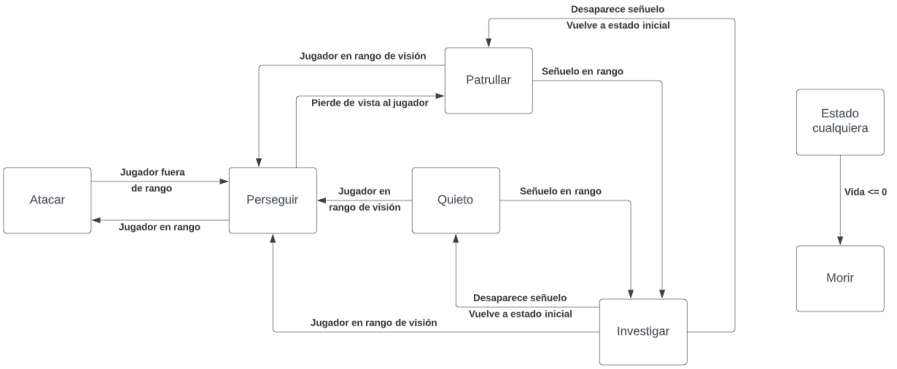

Figura 4.6: Máquina de estados de enemigos comunes

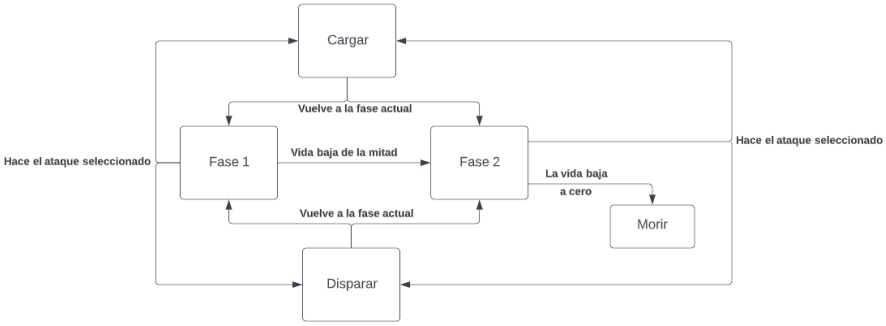

Figura 4.7: Máquina de estados del jefe final

Estado: Patrullar	
Descripción	El agente recorre el espacio que tenga disponible a una velocidad constante. La longitud de la patrulla puede variar, ya que depende del terreno que tenga disponible, variando de forma dinámica según la posición del agente en el mapa
Transiciones	
Señuelo	El agente detecta un señuelo lanzado por el jugador
Perseguir	El jugador entra en el campo de visión del agente
Morir	La vida del agente baja a cero

Tabla 4.5

Estado: Quieto	
Descripción	El agente permanece en su posición esperando estímulos externos
Transiciones	
Señuelo	El agente detecta un señuelo lanzado por el jugador
Perseguir	El jugador entra en el campo de visión del agente
Morir	La vida del agente baja a cero

Tabla 4.6

Estado: Perseguir	
Descripción	El agente sigue al jugador mediante un algoritmo de *pathfinding*
Transiciones	
Patrullar	Pierde de vista al jugador
Atacar	Alcanza al jugador, quedándose a rango de ataque
Morir	La vida del agente baja a cero

Tabla 4.7

Estado: Atacar	
Descripción	El agente se mantiene a rango de ataque del jugador y realiza la acción
Transiciones	
Perseguir	El jugador se aleja del agente y, por tanto, sale del rango de ataque del mismo
Morir	La vida del agente baja a cero

Tabla 4.8

Estado: Señuelo	
Descripción	Mientras que el jugador no está en rango de visión, el agente detecta un señuelo y se mueve hacia él
Transiciones	
Perseguir	El jugador entra en el rango de visión
Patrullar	El señuelo desaparece y el jugador no está en rango de visión
Morir	La vida del agente baja a cero

Tabla 4.9

Estado: Fase 1	
Descripción	El jefe tiene más de la mitad de la vida. Sus ataques son más flojos y fáciles de esquivar y te persigue de forma continua
Transiciones	
Fase 2	El jefe pierde la mitad de la vida
Cargar	Ataque elegido de forma aleatoria
Disparar	Ataque elegido de forma aleatoria

Tabla 4.10

Estado: Fase 2	
Descripción	El jefe ya ha perdido la mitad de la vida. Sus ataques se vuelven más fuertes y difíciles de esquivar. Sigue persiguiendo al jugador de forma continuada
Transiciones	
Cargar	Ataque elegido de forma aleatoria
Disparar	Ataque elegido de forma aleatoria
Morir	La vida del jefe baja a cero

Tabla 4.11

Estado: Cargar	
Descripción	Realiza una embestida a gran velocidad hacia la posición actual del jugador y continua hasta chocar con una pared
Transiciones	
Fase 1	El ataque acaba y el jefe todavía tiene más de la mitad de la vida
Fase 2	el ataque acaba y el jefe tiene menos de la mitad de la vida

Tabla 4.12

Estado: Disparar	
Descripción	Dispara una cantidad variable bolas de energía dependiendo de la fase en la que estuviese el jefe. En caso de ser la fase 2 además podrá realizar un nuevo ataque consistente en dejar caer piedras desde el techo
Transiciones	
Fase 1	El ataque acaba y el jefe todavía tiene más de la mitad de la vida
Fase 2	el ataque acaba y el jefe tiene menos de la mitad de la vida

Tabla 4.13

Estado: Morir	
Descripción	El agente muere

Tabla 4.14

4.7 Objetos

Muchos de los obstáculos que presentará el juego requerirán el uso de diversos objetos para que el jugador pueda progresar. Estos objetos podrán ser accedidos a partir de la interfaz que contendrá el inventario, como se ha explicado en su apartado anterior.

La mayoría serán obtenidos por medio de diferentes cofres repartidos por los escenarios, aunque los enemigos tendrán una posibilidad de dejar caer algunos de estos objetos al ser derrotados por el jugador. Durante los niveles presentes, el jugador podrá recolectar los objetos que se muestran a continuación:

- **Pociones de vida:** Objeto utilizable que permite al jugador regenerar una cantidad de vida variable, conforme el color de la poción.
- **Espada:** Artilugio necesario para poder realizar ataques con los que dañar a los enemigos.
- **Señuelo:** Objeto que puede ser equipado desde la interfaz del inventario, mostrando su trayectoria hasta donde se lanzará. Una vez arrojado se mantendrá en el suelo durante un corto periodo de tiempo, atrayendo a aquellos enemigos que se encuentren a un rango máximo y sirviendo como distracción para poder avanzar y evitar un enfrentamiento directo.
- **Bola de agua:** Permite apagar antorchas existentes en los niveles, permitiendo una mayor interacción y eliminando fuentes de luz que reducen el rango de visión de los enemigos.
- **Bola de fuego:** Del mismo modo que el objeto anterior, su principal utilidad recae en la interacción con las antorchas, permitiendo ser encendidas.

4.8 Almacenamiento

Para que el usuario no deba empezar una misma partida siempre que se inicie la aplicación, se ha planteado la introducción de un sistema que permita recolectar su transcurso durante una partida para que pueda ser accedida en el futuro.

Dicho esto, se pensaron que elementos deberán ser almacenados y de que forma realizarlo, a lo cual se concluyó la utilización de un archivo exterior al propio juego que contendrá toda esta información.

Este archivo deberá contener:

- **Salud actual del protagonista:** Al iniciar un nivel o sección nueva, la cantidad de vida presente en ese momento deberá ser preservada.
- **Cantidad de objetos en el inventario:** Todos aquellos objetos que haya recogido durante un nivel y su cantidad deberán encontrarse recogidos en sus datos permanentes.
- **Nivel en el que se encuentre:** Sí un jugador ha superado un nivel, a no ser que empiece una nueva partida no deberá volver a tener que repetirlo.
- **Objeto seleccionado:** Deberá preservarse el objeto que el jugador haya seleccionado con anterioridad.

4.9 Interacción con el escenario

Como se ha mencionado en secciones anteriores, el jugador será capaz de poder interactuar con elementos presentes en el escenario, pudiendo utilizar un objeto para apagar o encender antorchas, según encuentre conveniente. Sin embargo, también dispondrá de otro conjunto de opciones.

En cada nivel se encontrarán un conjunto de cofres, los cuales el usuario será capaz de abrirlos para poder obtener objetos, muchos de los cuales serán necesarios para poder proseguir. Asimismo, se presentarán diversos obstáculos en los que el jugador no podrá avanzar directamente, y tendrá que encontrar una manera alternativa para poder continuar, bien sea encontrando una forma para poder atravesar el inconveniente en cuestión o interactuando con los objetos que tenga a su disposición.

4.10 Misceláneo

Además de aquellos aparatos ya mencionados, es fundamental hacer énfasis en ciertos elementos cuya importancia es esencial a la hora de desarrollar y detallar un videojuego. A continuación, se presentan estos elementos:

- **Cámara:** Durante el transcurso del juego, la cámara seguirá al personaje principal a partir de un cierto retardo, ofreciendo una sensación de movimiento al margen del usuario. Asimismo, cada vez que el jugador ataque o reciba daño se producirá una vibración, dotando de un *feedback* claro de que ha producido alguna de las posibilidades mencionadas.
- **Sonido:** Durante todo el juego habrá música de fondo. Adicionalmente, tanto el protagonista como los enemigos presentan varios sonidos para ofrecer una mayor sensación a todas las acciones que presentan, ya sea por medio de caminar, saltar o incluso recibir daño, entre muchas otras.

CAPÍTULO 5
Desarrollo de la solución

El contenido que presenta este apartado indagará en cómo se ha realizado el desarrollo del proyecto. Cada sección presente dará una explicación sobre los componentes realizados, profundizando en cómo se ha llevado a cabo a partir de las herramientas disponibles y los elementos de programación escogidos.

Durante este periodo, el videojuego fue progresando por medio de la elaboración de prototipos, que incluían las diversas funcionalidades trabajadas para no entorpecer el desarrollo completo. No obstante, este punto no se tendrá en cuenta durante la elaboración de la memoria, ya que se busca centrarse en la versión final de cada una de las partes, explicando el proceso que se ha llevado a cabo para su realización.

5.1 Inventario

Durante la realización de este proyecto, se llevaron a cabo diversas implementaciones con respecto al sistema de almacenamiento de objetos, como en el siguiente modelo [15]. Sin embargo, las primeras versiones fueron descartadas debido a incompatibilidades con el sistema de persistencia. Teniendo en cuenta estas razones anteriores, su desarrollo concluyo en la versión final basada en este proyecto[16], cuya explicación se dará a continuación.

El componente principal se trata de la clase **Inventory**. Esta clase contiene dos *arrays* separados, encargados de manejar el objeto presente en cada uno de los huecos del inventario y la cantidad de estos respectivamente. Además, se ha introducido un tercer array cuya funcionalidad consiste en almacenar los *GameObject* que manejan un componente, imagen y texto con el objetivo de representar un espacio en el inventario. Con el objetivo de simplificar la estructura, cada ranura del inventario está relacionada con un tipo de objeto, de tal forma que siempre que el jugador obtenga un objeto, se colocará en la misma posición.

El usuario puede interactuar con cada hueco del inventario mediante las primeras seis teclas numéricas, indicándole en que hueco se encuentra durante ese momento cambiando su color. Si el usuario está seleccionando un hueco donde se encuentre un objeto, este podrá ser utilizado al realizar la acción correspondiente, variando respecto a cada tipo de objeto.

Respecto a las pociones, estas tendrán una cantidad específica que se mostrará por medio de un número en cada posición de la interfaz donde se encuentre. Para lograr esta funcionalidad, se ha optado por añadir un componente de texto que será modificado dependiendo de la cantidad de pociones actual, modificándose por un valor de *string* vació mientras solo contenga una unidad.

Cuando el jugador entra en contacto con el *Rigibody* contenido en cada objeto, este será añadido a su ranura correspondiente del inventario, modificando su cantidad existente. En caso de que ya se encuentre presente, se aumentará el número de veces que se encuentra almacenado en el array, aumentando su valor en el componente de texto que forma parte de la interfaz visual.

Por último, durante cada escena deberá estar presente el objeto **ItemAssets**. Este objeto presenta una clase con nombre homónimo, cuya instancia permite que se puedan generar todos los objetos disponibles dentro del juego, para que pueda ser utilizado por medio de diferentes clases. Para su realización, se ha optado por aplicar el patrón Singleton [17], de modo que solo existe una única instancia durante todas las escenas.

En la siguiente figura [5.1] podemos observar cómo se muestran cada uno de los seis huecos que forman parte de la interfaz del inventario. Para poder seleccionar un objeto se utilizarán las primeras seis teclas numéricas, permitiendo navegar al usuario entre cada posición del inventario, cambiando su componente imagen por un color rojo al ser elegido y revirtiendo su color al utilizar otra tecla. Es de notable importancia añadir que la interfaz permanecerá en todo momento oculta hasta que se pulse alguna de los números correspondiente, habilitando su componente visual por un breve instante de tiempo, en el cual volverá a ocultarse de nuevo.

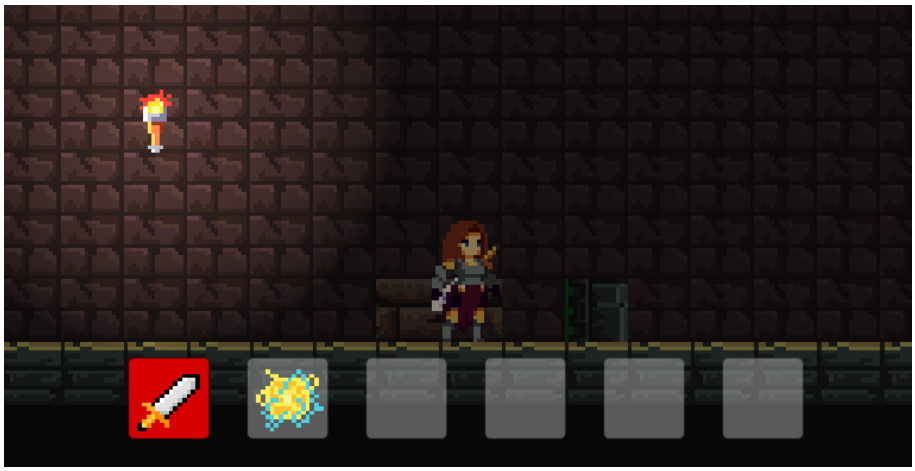

Figura 5.1: Interfaz del inventario

5.2 Salud

Para representar la cantidad de vida que el jugador tiene disponible durante el transcurso de la partida, se ha optado por diseñar una interfaz cuya localización se encontrará en la parte superior derecha. Consistirá en un *GameObject* formado por un conjunto de tres imágenes que contiene la representación de la salud como un *sprite* con diseño de corazón. Cada uno de estos tres *sprites* se situarán dentro de un *array*, equivaliendo a una cantidad de dos puntos de vida por cada corazón, cuya forma variará dependiendo de la vida del usuario, cambiando su componente imagen por el correspondiente:

- **Corazón completo:** Corresponde a dos puntos de vida.
- **Medio corazón:** Corresponde a un punto de vida.
- **Corazón vació:** Corresponde a cero puntos de vida.

Figura 5.2: Interfaz de salud

5.3 Almacenamiento

Durante el transcurso de una partida, el jugador genera un conjunto de información que deberá ser guardada para futuras partidas mediante el uso del *script* **GameData** basado en la siguiente estructura [18], entre las cuales se incluyen:

- Salud de protagonista
- Objetos presentes en el inventario
- Objetos selecionado
- Nivel donde se encuentra

Este conjunto de datos que el jugador genera durante una sesión de juego deberán ser recolectados por medio de un documento. Para la realización de este proyecto, se ha decidido almacenar este conjunto de información por medio de un archivo en formato *JSON*, el cual será accedido para su lectura y escritura. *JSON* es un formato focalizado en el intercambio de información centrado en ofrecer una estructura sencilla e independiente del lenguaje de programación, mediante el uso de una estructura universal. Está estructurado mediante:

- Una colección de pares de nombre/valor.
- Una lista ordenada de valores

Para evitar que el jugador pueda acceder a este registro y sea capaz de modificarlo a su antojo, se introdujo un método de cifrado mediante el uso de un algoritmo *XOR*. Este algoritmo consiste en aplicar a cada letra correspondiente en el archivo JSON una operación *XOR* mediante otro carácter, de tal forma que si queremos descifrar este conjunto de información para que pueda ser utilizado por Unity, solo deberemos realizar de nuevo la

operación con el mismo carácter. Su funcionamiento se aplicará a partir de un *string* que proporcionara las letras con las cuales se realizarán las transformaciones.

El elemento esencial que administra esta recogida de datos se trata de la clase **DataPersistenceManager**. Su principal finalidad consiste en ofrecer una instancia que contendrá un conjunto de métodos con el objetivo de poder crear una nueva partida, eliminando todos los datos anteriores, guardar o cargar una sesión. Estas dos últimas acciones requerirán del uso de un método encargado de llamar a todas las clases que contengan la interfaz **IDataPersistence**, compuesta por dos métodos cuya función es implementar que datos deberán ser almacenados o cargados respectivamente para cada uno de los *scripts* que la incluyan. Para conseguir llamar a todos los archivos de código que utilicen esta interfaz se hará uso de un método *FindObjectsOfType*, mediante el cual se almacenarán en una lista para posteriormente hacer uso de un bucle recorriendo cada elemento y ejecutando la función de la interfaz correspondiente.

Con el objetivo de mantener un registro de que cofres han sido recolectados, se ha llevado a cabo mediante el uso de tuplas, las cuales contiene un identificador único generado automáticamente para cada cofre, y un valor confirmado si su contenido ha sido extraído o no. Este tipo de datos provocaba ciertos problemas respecto a la sintaxis de *JSON*, por lo cual se tuvo que aplicar un conversor de tupla a tipo diccionario implementado mediante la clase **SerializableDictionary**.

5.4 Interfaces

Cada una de las interfaces que forman parten del proyecto deberán ser realizar mediante el objeto *Canvas* de Unity. *Canvas* es la herramienta que engloba todos los elementos presentes en la *UI*, los cuales deben cumplir dos requisitos esenciales:

- Permanecer como sus hijos en la jerarquía de *GameObjects*.
- Encontrarse dentro del límite visual formado por el *Canvas*.

Este conjunto de objetos sustituye el componente *Transform* y *Mesh Render*, presente en la mayoría de *GameObjects*, por un **ReactTransform** y **Canvas Render** respectivamente. Este primer elemento es el encargado de gestionar su posición, escala que presentará y donde quedará anclado respecto a su objeto padre.

Todas las interfaces contendrán un conjunto de textos encargados de guiar el usuario, definiendo cada opción disponible. Unity incorpora un *plugin* llamado **TextMeshPro**, que sustituye a su formato de representación textual original. La utilización de TextMeshPro es muy recomendada por muchos usuarios [19], ya que ofrece la posibilidad de aplicar efectos a textos, un conjunto mayor de posibles fuentes y una calidad de representación visual considerablemente superior.

Con el objetivo de gestionar la interacción del usuario con las interfaces se introduce el elemento Button. Sus funcionalidades se especifican mediante el método *OnClick* presente en un *script* dentro del propio objeto y modificable directamente sin la necesidad de aplicar código, permitiendo gestionar su navegación de forma sencilla e interactiva.

En la figura [5.3] se puede apreciar cómo se ha implementado el menú principal. Este menú se compone de un conjunto de cuatro botones encargados de aplicar distintas funcionalidades. Al pulsar el botón **NUEVA PARTIDA**, el jugador comienza desde el primer nivel, eliminando sus progresos anteriormente almacenados. Sí por el contrario, seleccionamos **CONTINUAR** accederemos al último nivel en el que hayamos dejado la

partida. Para acceder a la configuración, se deberá seleccionar el botón **OPCIONES**, que se encargará de ocultar el componente visual de este menú para habilitar el menú de opciones. Por último, si el usuario decide abandonar la sesión de juego, deberá pulsar el botón **SALIR**, cerrando la aplicación.

Figura 5.3: Navegación del menú principal

Si el jugador decide pausar el juego, deberá pulsar la tecla de escape, mediante la cual se mostrará el menú de pausa y la aplicación congelará el movimiento de todos los entes existentes en la escena, como se puede observar en la figura [5.4]. Al presionar el botón **CONTINUAR** producirá que se oculte la interfaz, reanudando el juego de nuevo. Los botones **OPCIONES** y **SALIR** ofrecerán la misma funcionalidad que en el menú anteriormente mencionado.

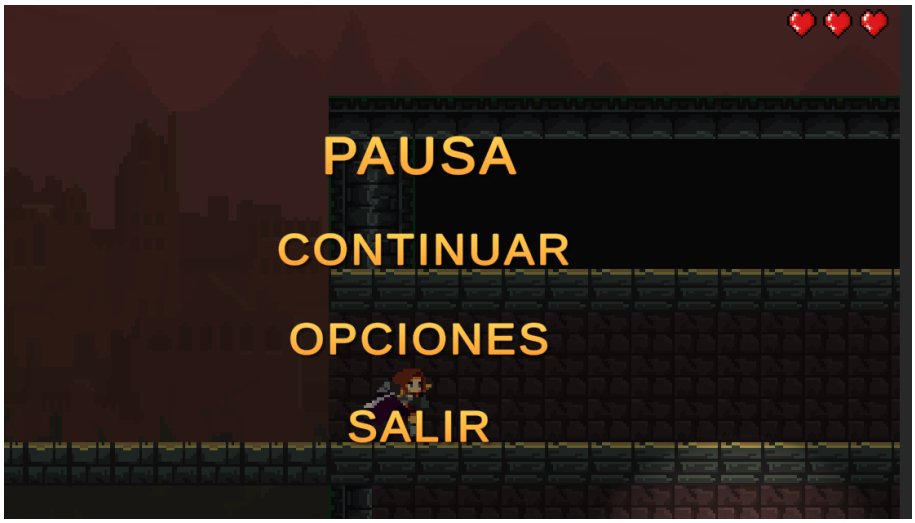

Figura 5.4: Navegación del menú de pausa

Cuando la salud del protagonista consigue reducirse hasta su mínimo, se mostrará el menú de muerte, deteniendo completamente el juego. Desde aquí, se le ofrecerá al usuario la posibilidad de reanudar el nivel, volviendo al punto inicial de la escena. Sin embargo, también es posible volver al menú principal desde otro botón.

Con el fin de completar un nivel, el jugador deberá atravesar un *Box Collider* situado en la última parte de la escena. Al entrar en contacto, se exhibirá una interfaz desde la

cual, se podrá acceder al siguiente escenario o retornar al menú principal, almacenando en ambos casos los datos de la partida.

Por otro lado, cuando el usuario complete el nivel final derrotando al jefe de la zona, aparecerá una interfaz mostrando un mensaje de victoria, en conjunto con una opción que retornará al jugador al menú principal.

Para ocultar el tiempo que lleva cargar el siguiente escenario, se ha desarrollado una pantalla de carga que oculta la escena hasta que se produce el cambio al nivel posterior, como se puede divisar en la figura [5.5].

Figura 5.5: Interfaz de la pantalla de carga

Con el propósito de que cada uno de los botones funcione correctamente, la escena debe contener un objeto **Event System**, creado automáticamente al introducir un *Canvas*.

5.5 Diálogos

A la hora de representar textos, Unity no posee de una herramienta capaz de poder mostrar párrafos continuos que sigan una historia o conversación. Debido a ello, existen una multitud de *plugins* exteriores que permiten la implementación de estas características, y para este proyecto se eligió la realización mediante *Ink*.

Ink se define como un lenguaje de código centrado en ofrecer capacidades narrativas a videojuegos. Mediante su editor oficial, **Inky**, se pueden crear y modificar archivos que contienen textos sofisticados, llegando a poder desarrollar decisiones para el jugador y aplicar efectos textuales. A causa de su interfaz sencilla e intuitiva, Ink es un lenguaje sencillo y simple de aprender, ya que no requiere gran cantidad de conocimientos en programación.

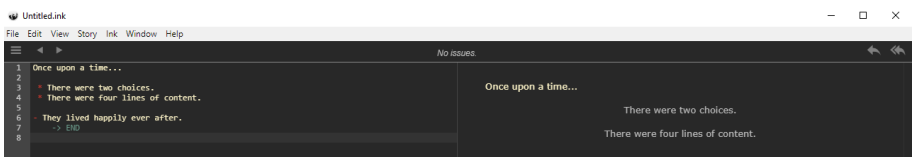

Figura 5.6: Interfaz del editor Inky

El principal motivo de su elección para esta aplicación fue debido a que ofrece una integración sencilla mediante Unity, proporcionando un conjunto de herramientas que

alivian la carga de trabajo, permitiendo recompilar archivos directamente o ver una representación del contenido.

El texto se exhibirá a partir de una interfaz sencilla en la parte superior de la pantalla. Se compondrá de un Panel que actuará como contenedor del diálogo y un objeto *TextMeshPro* que se modificará mostrando el escrito correspondiente.

A la hora de administrar que fragmento de texto se muestra en cada momento, se ha desarrollado un gestor encargado de administrar que contenido se expone al jugador. El gestor se encarga de presentar cada línea contenida en un fichero Inky, dando al usuario la posibilidad de avanzar el diálogo mediante la barra espaciadora. Mientras el jugador permanezca en una conversación, ni él ni los enemigos podrán moverse o realizar una acción.

Figura 5.7: Ejemplo de diálogo

5.6 Niveles

El contenido que presentará este apartado consistirá en explicar cómo se han desarrollado el conjunto de niveles que presenta la aplicación, dando paso a explicar que elementos lo componen.

Para poder generar un nivel dentro de Unity, necesitamos un entorno que sirva como lugar donde construir los componentes que presentará el escenario, para ello se debe utilizar un elemento conocido como *Scene* o escena. Cada uno de los diferentes niveles estarán dentro de una escena diferente, las cuales se organizarán por medio de índices a través de la opción *Build Settings*. De esta forma, podemos acceder a cualquier nivel a partir de su índice correspondiente.

En conjunto, existen un total de tres niveles que el jugador deberá superar a lo largo de una sesión, empezando por el primero y llegando al último para completar el juego. El objetivo es lograr llegar al punto más alejado a la derecha del mapa, sobrepasando a todos los enemigos presentes en las distintas áreas. Cada escena irá incrementando la dificultad, ofreciendo un desafío mayor con múltiples enemigos en las partes finales, en las cuales el usuario tendrá que aplicar sus máximas habilidades.

Con el objetivo de diseñar cada escenario, se ha dado uso del componente *Grid*, permitiendo unir los distintos *tiles* que presentan los *sprites* de cada bloque del juego. Las celdas pertenecientes a la *Grid* contendrán un tamaño de 16 x 16 píxeles de ancho y alto, ya que al ser un juego con estética 2D no presenta el eje Z de coordenadas. Cada conjunto de diferentes *tiles* se ha agrupado por medio de diferentes *gameObjects* con el componente *TilemapRender*. Su principal utilidad consiste en la opción *SortingLayer*, permitiendo ordenar que elementos se sobrepondrán visualmente entre otros por medio de la capa elegida. Adicionalmente, aquellos *tiles* que no deban ser atravesados por el jugador contendrán un componente *TilemapCollider 2D* y las plataformas incluirán el elemento *Plataform Effector 2D* añadiendo un efecto de fricción a sus esquinas.

5.7 Escenario e interacciones

El usuario no solo deberá enfrentarse a los enemigos presentes en el videojuego para poder avanzar, sino que se presentarán un conjunto de obstáculos e interacciones en las cuales el protagonista podrá obtener recursos para avanzar o conseguir objetos.

Cada escenario se encontrará poblado por un conjunto de cofres del tesoro, los cuales contendrán una recompensa, ya sea necesaria para continuar o totalmente opcional. Cuando el jugador se encuentre cerca de alguno de estos arcones, se mostrará un icono de interrogación indicando que puede interactuar con él. Si el usuario entonces pulsa la tecla E, se iniciará un diálogo explicando que se encontraba dentro del baúl, invocando el objeto correspondiente en el nivel. Una vez abierto, el protagonista ya no podrá interactuar con el cofre.

Durante su camino, el usuario localizará un conjunto de palancas, con la finalidad de abrir puertas y entradas secretas. Su funcionamiento será similar a los cofres anteriormente mencionados, apareciendo un icono de interrogación al encontrarse a sus proximidades. Mediante la tecla de interacción, el protagonista activará o desactivará la palanca, dependiendo del estado en el que se encuentre.

Las puertas mencionadas en el párrafo posterior presentarán dos estados:

- **Abierto:** El jugador y los enemigos podrán avanzar por la puerta.

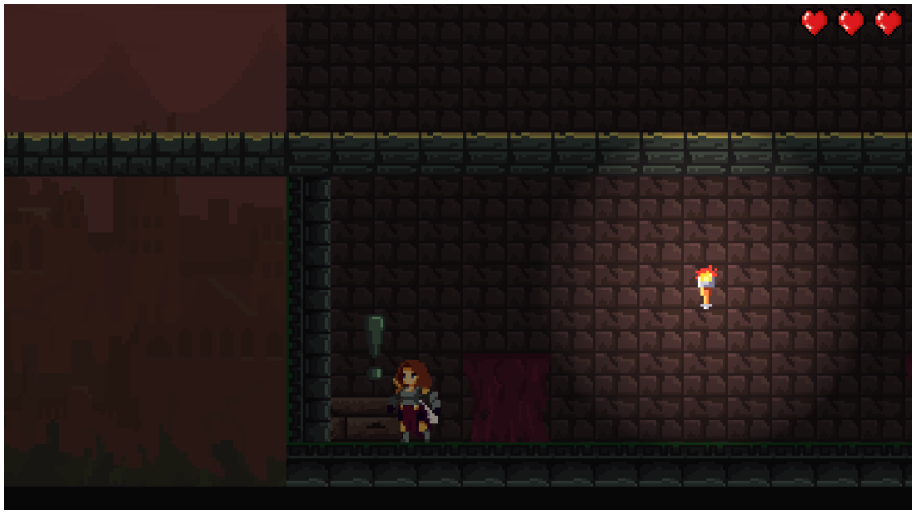

Figura 5.8: Jugador interactuando con un cofre

- **Cerrado:** Se impedirá que el jugador y los enemigos puedan cruzar la puerta.

Los escenarios se encontrarán iluminados mediante antorchas, aumentando el rango de visión para localizar al usuario que presentan los enemigos. Para decrementar esta ventaja, el protagonista podrá utilizar el objeto *Bola de agua*, el cual al lanzarlo cerca de una antorcha, se activará el *collider*, sustituyendo el sprite por un modelo apagado y eliminando el rango de luz producido por ellas. Por contra, mediante el objeto *Bola de fuego* podrá ser lanzado para encender antorchas, necesario para completar una sección en el último nivel del juego.

Presentes en la parte final de la aplicación, existirán una sección donde el protagonista deberá saltar por un conjunto de huecos, donde si cae por alguno de estos lugares, se activará la animación de muerte, pausando el juego y mostrando el menú de muerte.

5.8 Sonido

Para ampliar la experiencia de juego, se decidió aplicar una variedad de sonidos para el conjunto de elementos en la aplicación. La mayoría de estos archivos de sonido se han obtenido por medio de una página web de recursos gratuitos para el desarrollo de videojuegos, donde cualquier usuario puede subir su contenido para compartirlo con otros desarrolladores, como en la siguiente página [20].

Las listas presentes a continuación muestran el conjunto de ficheros de audio que deberán tener cada elemento para el proyecto:

- **Protagonista:**
  - **Pisadas:** Al caminar, se deberá reproducir un sonido.
  - **Ataque:** Realizar un golpe con la Espada incluirá reproducirá un archivo de audio.
  - **Daño:** Recibir un golpe provocará que el protagonista realice un sonido.
  - **Salto:** Al saltar, se deberá incluir un archivo de audio.
  - **Embestida:** Realizar esta acción provocará que se reproduzca un sonido.
  - **Objetos:** Al ser usado un objeto, se deberá reproducir su propio archivo de audio.

    - **Interacciones:** Cuando el protagonista interactué con el escenario, estos elementos producirán un sonido.

- **Enemigos:**

    - **Pisadas:** Al caminar, se deberá reproducir un sonido.
    - **Ataque:** Realizar un ataque incluirá reproducirá un archivo de audio.
    - **Daño:** Recibir un golpe provocará que el enemigo realice un sonido.
    - **Muerte:** Cuando su salud se reduzca a cero, cada enemigo deberá reproducir su propio archivo de audio.

- **Música:**

    - **Menú principal:** Durante el menú principal, se escuchará continuamente una pista de audio.
    - **Niveles:** Durante los escenarios deberá escucharse una música de fondo.

Para aplicar cada uno de estos archivos de audio, se ha dado uso de una clase que actúa como gestor administrando cada uno de los sonidos. Dentro de esta clase, se encontrarán tres arrays compuesto de archivos *Sound*, donde se incluirá cada uno de los sonidos dependiendo de si pertenecen a las pisadas, a la música o a efectos del juego.

Cada elemento Sound presentará un conjunto de atributos que se explicarán a continuación:

- **Clip:** Contiene el archivo de audio correspondiente
- **Name:** Nombre que lo identifica
- **Volume:** Intensidad del sonido
- **Pitch:** Intensidad del tono
- **Loop:** Al estar activo, el archivo se reproducirá continuamente

Al iniciar la aplicación, en el método *Awake* se añadirá un componente *AudioSource* para cada elemento presente en los *arrays*, necesario para administrar sonidos en Unity. Para ejecutar un archivo de audio específico, existirá un método que reproducirá un elemento perteneciente a cada tipo de *array*, el cual deberá ser incluido dentro de la función donde se requiera. Un ejemplo de esto podrá ser cuando el jugador recibe un golpe por parte del enemigo, entonces deberá llamarse a este método con el nombre del sonido elegido.

A la hora de controlar el nivel de sonido de cada elemento presente en el videojuego, se ha incluido un componente *Slider* presente en el menú de opciones, tanto para el menú principal como para el menú de pausa. Todos los tres diferentes tipos de *arrays* presentarán un *Slider* que se encargara de regular su variable *Volume*. Los valores disponibles se encontrarán en un intervalo conteniendo los valores entre 0 y 1, y podrá ser modificado mediante el uso del ratón a partir de una función *Listener*, que se encargará de llamar al método que producirá la nueva asignación de los valores cuando detecte un cambio en el *Slider*. Al iniciar la aplicación, los *Sliders* contendrán un valor predefinido de 0,5.

5.9 Protagonista

El elemento inicial de la mayoría de videojuegos es el protagonista, ya que es el medio principal mediante el cual el jugador interactúa con el escenario y el entorno. A continuación, se presentarán las partes en las cuales está formado este protagonista.

En primer lugar, se compondrá de un objeto vacío en Unity a partir del cual se diferenciaran cada uno de los componentes que lo forman, cada uno con sus propios elementos y funcionalidades. Dicho esto, el protagonista cuenta con un conjunto de *sprites* que muestran la forma visual del personaje, además de la utilización del componente **Animator** y de una máquina de estados de tipo **Animator controller**, que administra las animaciones dependiendo de las condiciones establecidas.

El protagonista contiene dos funcionalidades principales, la cuales se expondrán en los sub apartados siguientes:

5.9.1. Movimiento

Consiste en el conjunto de *scripts* que permiten al usuario realizar diversas acciones relacionadas con el movimiento. El jugador podrá desplazarse en la dirección escogida siempre y cuando no colisione con un objeto que no se pueda atravesar. Además, el protagonista podrá realizar la acción de salto, aumentando su posición vertical, permitiéndole llegar a plataformas elevadas o atravesar huecos. Para reducir su presencia ante los enemigos, el personaje podrá agacharse para reducir su *collider* y su velocidad de desplazamiento, además de evitar que pueda realizar otras acciones como atacar o saltar. Sí se encuentra en frente de una escalera, podrá utilizarlas como mecanismo para subir por ellas, haciendo que solo pueda desplazarse de forma vertical hasta finalizar el recorrido. Como adición, el usuario podrá embestir a los enemigos, dirigiéndose en una dirección en un corto instante de tiempo y golpeando a enemigos presentes en su travesía.

5.9.2. Combate

Las mecánicas que contienen todos elementos que proporcionará el combate se encuentran dentro del *script* **PlayerState**, incluido dentro del propio objeto **Player**. Durante un enfrentamiento contra un enemigo, el jugador podrá utilizar el objeto Espada con el cual rebajar la salud si realiza un ataque que conecte con el *collider* del rival y su objeto interno **AttackPosition**, que contiene el rango de ataque del protagonista. De la misma forma, si el enemigo consigue conectar un ataque dentro del área del jugador, este verá su salud reducida en uno. Cada una de estas interacciones cuentan con su animación correspondiente, las cuales en algunos casos se ha aplicado un retraso al inicio para que puedan encajar correctamente.

5.10 Enemigos

El principal obstáculo a superar presente en la mayoría de videojuegos se representa mediante enemigos. Para la realización de este proyecto, se han incluido un conjunto de enemigos cuyo comportamiento irá variando dependiendo de una máquina de estados compuesta por la clase **AiStateMachine**, exceptuando el jefe final que presentará unos estados exclusivos añadidos a este *script*. Cada estado irá cambiando a partir de las acciones del jugador y del entorno donde se encuentre, mostrando las animaciones correspondientes a cada acción.

5.10.1. Enemigos comunes

- **Quieto:** El enemigo actuará inmóvil, activado al perder de vista al jugador o al señuelo.
- **Patrullar:** Recorrerá una zona limitada a partir de la posición donde se localice, cambiando de dirección al encontrarse con una pared o hueco.
- **Perseguir:** El enemigo localiza al jugador y se dirige hacia él.
- **Atacar:** El enemigo realiza un ataque al jugador, dañándolo si consigue conectar.
- **Señuelo:** Al localizar un señuelo, el agente se mueve hacia él.
- **Morir:** El enemigo deja de moverse y desaparece a partir de un instante de tiempo.

5.10.2. Jefe final

- **Fase 1:** Estado inicial del jefe al entrar a la zona de combate, donde realizará una serie de ataques concretos.
- **Fase 2:** Al reducirse su vida, cambiará del estado Fase 1 a este, aumentando su número de ataques y agresividad.
- **Cargar:** El jefe embiste al jugador, dañándolo si hace colisión con él.
- **Disparar:** El jefe realiza un ataque a distancia, aumentando el número de ellos dependiendo del estado Fase en el que se encuentre.

CAPÍTULO 6
Pruebas

Las siguientes secciones mostrarán como una vez se finalizó el desarrollo del videojuego, se decidió contactar con un grupo de personas para que pudieran probar el correcto funcionamiento de todos los componentes presentes y dar un conjunto de ideas sobre posibles mejoras o añadidos. Además, se presentarán los enlaces a un video mostrando una partida completa y a el mismo juego para poder ser descargado.

6.1 Beta

El término *versión beta* se refiere al estado de un producto o software en una versión preliminar. Dicho de otra forma, su objetivo es poder presentar mejoras o detectar errores previos a su lanzamiento al público general.

Para lograrlo, se hace uso de un grupo de personas conocidas como *Beta tester*, encargas de explorar estos productos en busca de posibles fallos o problemas de diseño para su posterior corrección.

A la hora de aplicarlo a este proyecto, se ha reunido a un grupo de personas confiables que han probado la primera versión del videojuego, cuyos comentarios se mostrarán en los siguientes párrafos.

Durante una sección presente en el nivel final, un jugador tuvo problemas con la animación del protagonista, mostrándose como si estuviera presente en el aire aun estando en el suelo; sin embargo, se debía a que el *collider* del escenario estaba mal especificado, en la versión final fue corregido.

Al derrotar al jefe final, se encontró que el menú de pausa podía ser accedido, reanudando la partida pero continuando mostrando el menú de victoria. Se solucionó evitando que puedas acceder al menú de pausa cuando ya se encuentra un menú en pantalla.

Mientras se encontraba agachado, se descubrió que se podía realizar un ataque si el objeto Espada estaba equipado, no mostrando la animación pero realizando el audio y pudiendo dañar a enemigos. Para corregirlo, se ha evitado que el jugador pueda atacar al encontrarse en esa posición.

Ocultándose en una sección, se descubrió que si anteriormente te encontrabas agachado, al salir del estado oculto la animación presente era incorrecta, mostrando un movimiento normal al estar agachado. En una última versión el jugador permanece en el mismo estado al salir del escondite, evitando que la animación sea errónea.

Por último, existía la posibilidad de que el inventario no se actualizará correctamente, mostrando los objetos en una ranura incorrecta. Sin embargo, especificando la posición concreta de cada hueco no volvió a surgir este problema.

6.2 Feedback

Además de todas las pruebas realizadas, ciertos usuarios nos comentaron un conjunto de posibles mejoras aplicables a la aplicación a partir de sus gustos y opiniones, de las cuales se destacan las siguientes que fueron añadidas como parte del desarrollo.

La primera versión ofrecía unas interfaces que muchos usuarios no encontraban del todo entendibles, con colores muy apagados y sin tener una representación visual correcta de que botón está siendo seleccionado en cada momento. Con el objetivo de satisfacer estas necesidades, se optó por añadir colores más vivos a cada menú, aumentando el tamaño de todos los textos y añadiendo un efecto cuando el jugador sitúa el ratón encima de un botón, mostrando un color verdoso con el objetivo de indicar que acción elegirá.

Al obtener objetos, muchos usuarios comentaron que les parecía muy incómodo tener que recordar en que posición exacta se situaba cada objeto, causando confusión a la hora de tomar decisiones rápidas al enfrentarse a enemigos. Dicho esto, se procedió a fijar en que ranura se encontraba cada objeto, de tal forma que por ejemplo al recolectar una poción siempre se encontrará en el mismo hueco, haciendo que el usuario no deba memorizar en qué lugar será almacenado.

Por último, se obtuvieron comentarios de ciertos jugadores que notaban que el videojuego en sus primeras versiones estaba falto de ambiente, ya que no contenía ninguna música de fondo durante su transcurso, ofreciendo una experiencia más aburrida y falta de personalidad. Por esta razón, se dio paso a implementar una música que acompañará a los usuarios durante su aventura.

6.3 Enlaces

En esta sección se compartirán los enlaces de interés a disposición de los usuarios.

- **Video partida completa:** En la dirección siguiente contendrá un video mostrando las diferentes mecánicas de las que se compone el proyecto, superando cada uno de los niveles hasta lograr finalizarlo en su completitud. https://media.upv.es/#/portal/video/6973dcd0-fa19-11ec-a73b-db9432f33999
- **Proyecto:** En el enlace mostrado a continuación se podrá realizar la descarga de la versión final del proyecto, permitiendo su completa jugabilidad. https://drive.google.com/file/d/1-c4B643QZfcawnWyOfQDVdC5RTRCMZGp/view?usp=sharing

CAPÍTULO 7
Conclusiones

Haber conseguido realizar este proyecto, ha logrado aumentar las capacidades y experiencia en el ámbito de desarrollo de videojuegos, específicamente en el uso y manejo de la herramienta Unity, que han servido como un incremento de habilidad y capacidades respecto a lo aprendido durante este curso. Además, se ha obtenido un mayor entendimiento respecto a cómo organizar un proyecto mediante la aplicación de una metodología ágil, uno de los enfoques de desarrollo más importas y usados en gran variedad de entornos que permiten una mayor preparación de cara al futuro entorno laboral.

En referente a los objetivos propuestos al principio del desarrollo, la mayoría han sido completados sin ningún problema. El resultado final ha culminado en la creación de un videojuego de plataformas con elementos de sigilo completamente divertido y entretenido, el cual era el objetivo principal propuesto, permitiendo al usuario poder realizar saltos entre los distintos espacios alcanzables y esconderse de la presencia de los enemigos evitando ser detectado y pudiendo lograr evadir enfrentamientos difíciles. Igualmente, los enemigos presentes en las diferentes secciones reaccionan a la presencia del jugador y a diversos objetos, por no hablar de como las interacciones existentes proporcionan una experiencia más completa, dotando al jugador de diversos objetivos a cumplir y caminos que escoger.

A pesar de ello, se han producido ciertos problemas en el proceso de desarrollo que han entorpecido el tiempo de producción, como por ejemplo la implementación del inventario y el sistema de almacenamiento, que tuvieron que ser planteados de nuevo, ya que provocaron incompatibilidades entre ellos. Asimismo, tampoco se ha conseguido obtener un modo cooperativo funcional que cooperar con otro jugador debido falta de tiempo, aunque está desarrollándose en un TFG distinto a este y podrá ser presentado en un futuro.

CAPÍTULO 8
Relación con los estudios

Es de notable mención como el desarrollo de este proyecto está estrechamente ligado a la asignatura de **Desarrollo de Videojuegos 3D**, en la cual se participó en un producto de envergadura similar, trabajando en una aplicación con un conjunto de estudiantes mediante el uso de Unity y proporcionando actualizaciones de cada una de las versiones desarrolladas a partir de una fecha concreta. Pese a ello, también ha tenido una gran importancia algunas asignaturas que han proporcionado la adquisición de los conocimientos básicos necesarios para afrontar este desafío, como **Programación** e **Introducción a la programación**.

Adicionalmente, las habilidades adquiridas a partir de asignaturas como **Proceso de software**, **Proyecto de ingeniería de software** y **Gestión de proyectos** han dado una idea clara y concisa de como poder afrontar y estructurar trabajos en conjunto. Dicho esto, es destacable resaltar como **Análisis y especificación de requisitos** e **Ingeniería del software** han permitido la creación de los diagramas de descripción utilizados y aplicados para describir los conceptos presentes. Por último, cabe remarcar como **Interfaces persona computador** ha influido notablemente en la creación y diseño de las interfaces, conceptos estudiados y adquiridos durante su participación.

CAPÍTULO 9
Trabajos futuros

Durante esta sección se explicarán que ciertas mejoras o añadidos podrían haberse implementado si se hubiera dispuesto de un intervalo de tiempo mayor para su desarrollo. A continuación, se propondrán las mejoras seleccionadas para una futura actualización:

- En lo que duración respecta, el proyecto presenta una cantidad de niveles que podrían haberse expandido de presentar más tiempo, provocando que el videojuego pueda sentirse algo corto. Debido a ello, se podrían realizar una mayor cantidad de niveles aparte de los existentes, o incluso expandir aquellos presentes, añadiendo más caminos y obstáculos.
- Durante el proceso de planificación, se presentó la idea de dotar al jugador de un ataque a distancia complementario que permitiera causar daño a los enemigos, pero no llego a aplicarse debido a problemas con la implementación. Por lo tanto, sería posible expandir el repertorio del jugador con diversas armas y objetos que afecten a las mecánicas y entornos.
- Como se ha mencionado, la intención era proporcionar un sistema multijugador donde dos usuarios puedan superar juntos la misma partida de forma *online*, pero en la versión final no existe este modo. En consecuencia, se buscaría ofrecer este servicio para aumentar el grado de satisfacción entre jugadores y proponiendo una nueva forma de jugar, aumentando las veces que el título puede ser jugado.
- La presencia de puntos de guardado donde el jugador pueda reaparecer en caso de muerte o al guardar la partida es una característica que contienen la mayoría de videojuegos de este estilo. De este modo, el jugador podría experimentar más frecuentemente ciertas zonas y desafíos, sin tener que volver a repetir el nivel desde el principio.

CAPÍTULO 10
Glosario

- **2D:** Estilo de juego que no contiene geometría tridimensional.
- **AER:** Análisis y especificación de requisitos.
- **Arcade:** Tipo de videojuego creado para ser jugado en salones recreativos.
- **Array:** Objeto similar a una lista cuyo objetivo es almacenar variables de un mismo tipo.
- **Asset:** Tipo de archivo presente en el proyecto de Unity y que puede ser utilizado.
- **Awake:** Método presente en un script llamado siempre primero una única vez cuando se carga la escena.
- **Backlog:** Lista que contiene el conjunto de actividades a desarrollar para un proyecto.
- **Box Collider:** Componente que proporciona colisiones en forma de cubo al objeto elegido.
- **Burndown:** Representación gráfica del trabajo restante.
- **Brainstroming:** Método grupal cuyo objetivo es generar ideas a partir de un ambiente creativo.
- **Cámara:** Mecanismo que permite mostrar la vista de un videojuego.
- **Consola:** Sistema electrónico encargado de ejecutar videojuegos contenidos en un dispositivo de almacenamiento.
- **Collider:** Componente que define la forma de un objeto para que le afecten colisiones físicas.
- **Escena:** Contenedor de todos los escenarios y elementos que presenta un juego, como si fueran niveles o secciones.
- **Feedback:** Evaluación por parte de individuos que permite obtener las fortalezas y debilidades de algo que esté siendo evaluado.
- **Game Object:** Objetos presentes dentro de una escena en Unity.
- **Indie:** Aquellos juegos creados por empresas independientes, normalmente mediante un presupuesto y equipos de desarrollo menores.

- **Juego de plataformas:** Género de videojuegos característico por tener que recorrer una serie de plataformas o superficies para superar los niveles.
- **Máquina de estados:** Modelo de comportamiento donde existen estados que representan las entradas y salidas dependientes entre ellas a partir de ciertas condiciones.
- **Mecánicas:** Conjunto de reglas que constituyen un videojuego.
- **Metodología ágil:** Enfoque software centrado en el desarrollo iterativo e incremental, donde las necesidades van evolucionando conforme el tiempo.
- **Novela visual:** Género de videojuegos centrado en ofrecer una experiencia narrativa acompañada por imágenes y personajes.
- **Online:** Termino referido a como algo se encuentra conectado en Internet.
- **Pixel art:** Estilo artístico creado a partir la construcción de imágenes mediante píxeles.
- **Plataforma:** Sistema operativo encargado de ejecutar aplicaciones compatibles con él.
- **Plugin:** Programa informático cuyo objetivo es añadir funcionalidades adicionales.
- **Rigibody:** Componente de Unity que dota a los objetos de físicas.
- **ROM:** Sistema de almacenamiento de información centrado en la lectura de datos.
- **RPG:** Género de videojuegos donde el jugador asume el papel de un personaje dentro de un universo o mundo.
- **Stakeholders:** Individuos u organizaciones que se sienten afectados respecto a las acciones de una empresa.
- **Script:** Fragmento de código encargado de añadir o realizar funciones.
- **Shooter:** Género de videojuegos centrado en controlar a un personaje equipado con una especie de arma a distancia.
- **Sprint:** Intervalos de tiempo cortos entre una y cuatro semanas donde se avanza en el desarrollo ágil.
- **Sprite:** Imagen en formato 2D cuya funcionalidad es mostrar el diseño de un objeto en escena presente en Unity. Puede ser utilizado en múltiples aspectos visuales, como los personajes o el escenario.
- **String:** Conjunto de caracteres usados para representar textos.
- **Tilemap:** Componente que permite dibujar celdas dentro de Unity.
- **Tile:** Celda que contiene un elemento visual.
- **Transform:** Componente encargado de manejar la posición de un objeto en la escena.
- **UI:** La interfaz con la que interactúa el usuario.
- **UT:** Tarea a realizar durante el desarrollo ágil.
- **XOR:** Operador lógico de tipo disyunción entre dos operandos, aplicando la disyunción exclusiva.

Bibliografía

[1] **Will Goldstone**. *Unity Game Development Essentials*. Packt Publishing; N.º 1 edition, 2009

[2] **Standish Group**. *Why Agile is Better than Waterfall*. [En línea] https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/.

[3] **Tristan Donovan**. *Replay: The History of Video Games*. Yellow Ant; Illustrated edition, 2010.

[4] **Riad Chikhani** *The History Of Gaming: An Evolving Community*. [TechCruch+] [En línea] https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/.

[5] **University of Oxford**. *Groundbreaking new study says time spent playing video games can be good your well*. Entrevista a: Professor Przybylski. [En línea] https://www.ox.ac.uk/news/2020-11-16-groundbreaking-new-study-says-time-spent-playing-video-games-can-be-good-your-well.

[6] **Amy B. Jordan Y Daniel Romer**. *Media and the Well-Being of Children and Adolescents*. OUP USA; N.º 1 edición, 2014.

[7] **Jason Rutter, Jo Bryce**. *Understanding Digital Games*. SAGE Publications Ltd; N.º 1 edition, 2006.

[8] **Portal ISO 25000**. [En línea] https://iso25000.com/index.php/normas-iso-25000/iso-25010.

[9] **Antonín Šmíd**. Comparison of Unity and Unreal Engine. https://core.ac.uk/download/pdf/84832291.pdf.

[10] **C#**. Microsoft. [En línea] https://docs.microsoft.com/es-es/dotnet/csharp/.

[11] **Unity Documentation**. Unity. [En línea] https://docs.unity3d.com/2021.2/Documentation/Manual/index.html.

[12] **Stack Overflow**. [En línea] https://es.stackoverflow.com.

[13] **Unity Asset Store**. Unity. [En línea] https://assetstore.unity.com/?gclid=CjwKCAjwquWVBhBrEiwAt1KmwuublLSALWo-KSP88ml3h_BhiB3XjW0Vo7ZI26n-3qZ4MtjBspqRwxoCh40QAvD_BwE&gclsrc=aw.ds.

[14] **Aseprite**. Igara Studio S.A. [En línea] https://www.aseprite.org.

[15] **Jayanam**. Youtube. [En línea] https://www.youtube.com/watch?v=Hj7AZkyojdo.

[16] **BlackthornProd**. Github. [En línea] https://github.com/BlackthornProd/Inventory-Tutorial-Series.

[17] **Alexander Shvets**. *Sumérgete en los patrones de diseño*.

[18] **TreverMock**. Github. [En línea] https://github.com/trevermock.

[19] **Brackeys**. Github. [En línea] https://github.com/Brackeys.

[20] **OpenGameArt**. [En línea] https://opengameart.org.

APÉNDICE A
Controles

La siguiente tabla se encargará de mostrar al lector los controles principales que presenta la aplicación:

	Teclas
Movimiento	A / D
Salto	Espacio
Agacharse	C
Embestida	Mayús
Escaleras de mano	W / S
Bajar escaleras o plataformas	Mantener S
Interactuar // Esconderse	E
Usar objeto	Click izquierdo
Inventario	1-6
Menú de pausa	ESC
Mirar hacia abajo	S estando quieto

Tabla A.1: Controles

APÉNDICE B
Objetivos de Desarrollo Sostenible

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

Objetivos de Desarrollo Sostenible	**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

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

Durante las primeras fases del diseño y desarrollo de este proyecto se pretendían conseguir unos objetivos distintos a aquellos propuestos en estos objetivos de desarrollo sostenible. Es por ello que el juego relacionado no estará estrechamente relacionado con ninguno de los mismos, aunque si puede ayudar al cumplimiento de varios de ellos, tal y como podemos ver en varios de los apartados de la propia memoria y que se explicará a continuación.

El **ODS** con el que más está relacionado sería el de **Salud y bienestar**. En todo momento, uno de los objetivos principales a la hora de diseñar cualquier juego debe ser conseguir que aquella persona que lo vaya a jugar se divierta, lo cual afecta de forma directa a su bienestar. Además, como hemos visto en el apartado de la **Importancia de los videojuegos en el mundo contemporáneo**, existen numerosos estudios que relacionan de forma directa los videojuegos con el bienestar de los jugadores al cumplir ciertas condiciones, sobre todo en épocas en las que existen complicaciones para las conexiones sociales como ha sido el confinamiento. Es por ello que la existencia de juegos con los que poder desconectar durante unas horas o relacionarte con otras personas puede afectar de forma muy positiva en la salud mental de los jugadores. Además, aunque no sería en el caso de nuestro juego, con el paso del tiempo los videojuegos se están empleando cada vez más en procesos de rehabilitación, así como educación en clases de distintas edades, siendo capaces de ayudar con diferentes competencias tan importantes como lo puede ser la creatividad, así como enseñar de forma lúdica algunas asignaturas como puede ser historia, dando a los alumnos una motivación mayor a la que presentarían en muchos otros casos.

Otro de los objetivos de desarrollo sostenible con el que se podría enlazar el desarrollo del videojuego, aunque en menor medida, sería el de **Trabajo decente y crecimiento económico**. Esto es debido a que, como se ha explicado en la memoria del trabajo, la industria de los videojuegos es actualmente el sector de entretenimiento que más dinero genera, por encima de otros sectores como podría ser el cine. Además, este sector está todavía consolidándose, creciendo más cada año. Esto además afectaría a nuestro proyecto en el caso de continuar con su desarrollo en un futuro, participando en este sector en crecimiento.

Por último, otro de los objetivos que se podría relacionar con nuestro proyecto sería el ODS **Industria, innovación e infraestructuras**. Este objetivo estaría estrechamente relacionado con el anterior en cuanto al crecimiento económico que ha caracterizado a esta industria desde los comienzos a la actualidad. Además, podemos ver que es posiblemente una de las industrias en las que más importancia se le ha dado a la innovación debido, entre otras cosas, a los grandes saltos que se han producido en la tecnología en las últimas décadas. Esto es fácilmente distinguible al ver los grandes cambios que se han llevado a cabo en este sector desde sus orígenes.