Wirlous' Little Adventure

Autor: Carlos Fajardo Sánchez
Tutor: Helio Tejedor Navarro
Profesor: Joan Arnedo Moreno

Máster Universitario en Diseño y Programación de Videojuegos
Programación avanzada

Junio 2022

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Wirlous´ Little Adventure*
Nombre del autor:	*Carlos Fajardo Sánchez*
Nombre del colaborador/a docente:	*Helio Tejedor Navarro*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	*06/2022*
Titulación o programa:	*Plan de estudios*
Área del Trabajo Final:	*Programación avanzada*
Idioma del trabajo:	*Español*
Palabras clave	*Videogame, random generation, top-down perspective, roguelike*

Resumen del Trabajo (máximo 250 palabras): Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo

En este proyecto se ha creado un juego roguelike 2D de acción y aventura. Los dos aspectos principales que se quieren explorar en este trabajo son la creación de niveles procedimentales y crear un juego que invite a la rejugabilidad.

A lo largo del documento se explicarán la definición e implementación del juego, cuáles has sido las decisiones tomadas y por qué. El objetivo final es conseguir un juego que funcione en PC y se pueda ejecutar en una página web. Además, se quiere subir el juego a una plataforma distribuidora de videojuegos.

El objetivo personal con este trabajo es aprender a desarrollar la idea de un videojuego partiendo desde el principio y adaptándose a los imprevistos que puedan surgir a lo largo del desarrollo.

Abstract (in English, 250 words or less):

The focus of this project is to develop a roguelike action-adventure 2D game. There are two main aspects that want to be explored in this work. To create a set of procedural levels, and to create a game with a focus on replayability.

This document will explain the definition and development of the game. It will cover the decision-making process to reach the final stage of the game development. The final goal is to create a game that works on PC and can be run on a web browser. And can be uploaded on a videogame distribution platform.

My personal goal with this work is to learn how to develop a game from the ground up. Starting from an initial idea and adapting the development cycle to any eventualities that may occur.

Cita

"Never pay more than 20 bucks for a computer game"

- Guybrush Threepwood (Monkey Island series)

Abstract

The focus of this project is to develop a roguelike action-adventure 2D game. There are two main aspects that want to be explored in this work. To create a set of procedural levels, and to create a game with a focus on replayability.

This document will explain the definition and development of the game. It will cover the decision-making process to reach the final stage of the game development. The final goal is to create a game that works on PC and can be run on a web browser. And can be uploaded on a videogame distribution platform.

My personal goal with this work is to learn how to develop a game from the ground up. Starting from an initial idea and adapting the development cycle to any eventualities that may occur.

Resumen

En este proyecto se ha creado un juego roguelike 2D de acción y aventura. Los dos aspectos principales que se quieren explorar en este trabajo son la creación de niveles procedimentales y crear un juego que invite a la rejugabilidad.

A lo largo del documento se explicarán la definición e implementación del juego, cuáles has sido las decisiones tomadas y por qué. El objetivo final es conseguir un juego que funcione en PC y se pueda ejecutar en una página web. Además, se quiere subir el juego a una plataforma distribuidora de videojuegos.

El objetivo personal con este trabajo es aprender a desarrollar la idea de un videojuego partiendo desde el principio y adaptándose a los imprevistos que puedan surgir a lo largo del desarrollo.

Palabras clave

Videogame, random generation, top-down perspective, roguelike

Videojuego, generación aleatoria, perspectiva top-down, roguelike

Índice

Figuras y tablas

Índice de figuras

1. Introducción

1.1. Introducción

En este documento se expondrá el trabajo realizado para la creación de un videojuego de carácter no comercial. Se explorará el diseño de un juego de estilo 2D con una bucle de juego corto e interesante que se preste a la rejugabilidad.

En primer lugar, se hará una breve descripción del juego que se va a desarrollar. Se explicará cuáles son los pasos para pasar de la definición inicial hasta tener un videojuego funcional, el cual se alojará en una plataforma de distribución de videojuegos.

Para poder alcanzar los objetivos en un tiempo y forma adecuada, se ha realizado una planificación de las tareas necesarias para conseguir el juego. También se ha hecho una estimación del presupuesto para desarrollar el juego en el plazo de tiempo establecido. Aunque es cierto que este presupuesto no corresponde con un gasto real al no tratarse de un juego comercial, si cumple una función orientativa de lo que supondría desarrollar un juego de características similares.

1.2. Descripción

Wirlous' Little Adventure está enfocado como un juego roguelike de corta duración. El objetivo es que el jugador quiera volver a jugar una y otra vez para mejorar su puntuación y tiempo. Las partidas tienen una duración corta de entre 5 y 10 minutos y la dificultad del juego es baja para incentivar al jugador a volver al juego.

Este juego toma como referencias a otros roguelikes como The Binding of Isaac: Rebirth (Nicalis Inc., 2014) y Enter the Gungeon (Dodge Roll, 2016) para los aspectos del bucle del juego y la rejugabilidad. Aunque varias de las mecánicas de combate del juego se asemejarían más a juegos como The Legend of Zelda: A Link to the Past (Nintendo EAD, 1991).

1.2.1. Mecánicas

Las mecánicas del juego son las siguientes:

- **Combate**. El jugador tiene que luchar con enemigos a lo largo de la mazmorra para poder avanzar. Hay dos estilos de combate:
    - **Cuerpo a cuerpo**. Estilo espada o hacha. Las armas pueden variar en velocidad de uso, alcance y daño.
    - **Armas a distancia**. Estilo arco o magia. Las armas pueden variar en munición/maná, alcance, área de efecto, velocidad del proyectil y daño.
- **Inventario**. El jugador tiene que gestionar un pequeño inventario.
    - **Armas**: El jugador puede llevar dos armas consigo en todo momento, una de tipo cuerpo a cuerpo y otra arma a distancia.
    - **Munición**: Para armas a distancia.
    - **Pociones**: Para recuperar vida y maná.
- **Navegación**. La mazmorra está dividida en varios niveles. Para avanzar de nivel el jugador debe encontrar las escaleras hacia el nivel inferior. La dificultad de la navegación puede incrementarse usando sistemas de llave/cerrojo para poder avanzar.

Los objetivos del juego es conseguir una puntuación máxima y llegar al final de la mazmorra en el menor tiempo posible. En la interfaz del juego, el jugador puede ver su puntuación actual debido al combate y el tiempo de la partida. Una vez finalizada, se calculará la puntuación final en función del combate y del tiempo de la partida, recompensando el menor daño recibido y la mayor velocidad a la hora de superar la mazmorra.

1.3. Objetivos generales

1.3.1. Objetivos principales

Objetivos del juego:

- Rejugabilidad mediante la generación de niveles aleatorios y variedad de armas.
- Enfocado a jugadores casuales con poco tiempo para jugar.

Objetivos para el jugador:

- Conseguir una puntuación alta.
- Completar la mazmorra en el menor tiempo posible.

Objetivos personales con el juego:

- Realizar un juego 2D sencillo y completo.
- Aprender a generar niveles de forma procedimental.
- Conseguir un bucle de juego interesante y variado para partidas de corta duración.

1.3.2. Objetivos secundarios

Objetivos adicionales que enriquecen el proyecto. Estos objetivos se intentarán incluir en el producto final en función del progreso:

- Generar niveles con una distribución orgánica.
- Crear pequeños retos en forma de puzles al jugador para poder avanzar por los niveles.
- Crear variedad en la IA para los enemigos.
- Profundizar en los efectos visuales y auditivos del juego.

1.4. Metodología y proceso de trabajo

Para realizar el juego se ha usado el motor de desarrollo Unity (Unity Technologies, 2005). Los assets usados en la parte gráfica son de carácter gratuito. Para realizar modificaciones a estos assets, o crear algunos nuevos, se ha usado Aseprite (Igara Studio S.A., 2001) y Photoshop (Adobe Inc., 1990) para la parte gráfica.

Los primeros prototipos que se han realizado usando un enfoque de "grey box" para enforcarse en el desarrollo de las mecánicas primero y dejar para las etapas más avanzadas la parte grafica del juego.

Para guardar un histórico del desarrollo del juego se ha creado un repositorio GIT (Torvalds, 2005), el cual se ha alojado en GitHub (Microsoft, 2008) de forma abierta para que cualquier persona tenga acceso al código fuente y los assets usados.

El producto final y todos los prototipos con cierta entidad se han subido a itch.io (Leaf Corcoran, 2013) para poder distribuir el ejecutable (PC) y jugar online en la propia página compilando el juego para web.

1.5. Planificación

La estimación inicial de este trabajo es de 200 horas de trabajo, equivalente a los 12 ECTS del Trabajo Fin de Master. Dichas horas se han repartido entre las distintas tareas:

- Definición del trabajo.
- Prototipado de las mecánicas.
- Implementación del juego.
- Pulido del juego.
- Documentación y defensa del proyecto.

Los hitos del trabajo coinciden con las fechas de entrega de este proyecto:

- Primera semana de marzo. Definición del proyecto a realizar.
- Primera semana de abril. Prototipado de las mecánicas y creación de un prototipo jugable del juego.
- Primera semana de mayo. Desarrollar los niveles, ajustar del bucle de juego y pulir los aspectos técnicos, gráficos y sonoros del juego. Conseguir un Producto Mínimo Viable (MVP).
- Primera semana de junio. Preparación de la documentación, videos y presentación del proyecto.
- Tercera semana de junio. Presentación del proyecto al tribunal de la Universitat Oberta de Catalunya (UOC).

En el siguiente diagrama de Gantt se puede ver una estimación inicial de las tareas necesarias desde la definición del juego hasta conseguir el MVP. En este diagrama se ha excluido la planificación relativa a la presentación de esta memoria.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Definición del proyecto	6 días	mar 01/03/22	dom 06/03/22	
2		Crear repositorio Git	0 días	dom 06/03/22	dom 06/03/22	1
3		Control del personaje	4 días	lun 07/03/22	jue 10/03/22	2
4		IA enemigos	4 días	vie 11/03/22	lun 14/03/22	3
5		**Armas**	**4 días**	**mar 15/03/22**	**vie 18/03/22**	
6		Arma cuerpo a cuerpo	2 días	mar 15/03/22	mié 16/03/22	4
7		Arma de alcance	2 días	jue 17/03/22	vie 18/03/22	6
8		**Sistemas**	**4 días**	**sáb 19/03/22**	**mar 22/03/22**	
9		Sistema de vida	2 días	sáb 19/03/22	dom 20/03/22	5
10		Sistema de municion / man	2 días	lun 21/03/22	mar 22/03/22	9
11		**Inventario**	**3 días**	**mié 23/03/22**	**vie 25/03/22**	
12		Pociones vida	1 día	mié 23/03/22	mié 23/03/22	8
13		Pociones mana	1 día	jue 24/03/22	jue 24/03/22	12
14		Municion	1 día	vie 25/03/22	vie 25/03/22	13
15		Interfaz	1 día	sáb 26/03/22	sáb 26/03/22	11
16		Menu	1 día	dom 27/03/22	dom 27/03/22	15
17		Sonido base	2 días	lun 28/03/22	mar 29/03/22	16
18		Primer prototipo	0 días	mié 30/03/22	mié 30/03/22	1;2;3;4;5;11;15;16;17
19		Generacion de nivels	8 días	mié 30/03/22	mié 06/04/22	18
20		**Graficos**	**7 días**	**jue 07/04/22**	**mié 13/04/22**	
21		Sistema de tiles	3 días	jue 07/04/22	sáb 09/04/22	19
22		Animaciones personajes	4 días	dom 10/04/22	mié 13/04/22	21
23		**Sonido**	**6 días**	**jue 14/04/22**	**mar 19/04/22**	
24		Efectos	3 días	jue 14/04/22	sáb 16/04/22	22
25		Musica	3 días	dom 17/04/22	mar 19/04/22	24
26		Testeo	11 días	mié 20/04/22	sáb 30/04/22	25
27		Pulir	7 días	dom 01/05/22	sáb 07/05/22	26
28		Version jugable	0 días	dom 08/05/22	dom 08/05/22	18;19;20;23;26;27

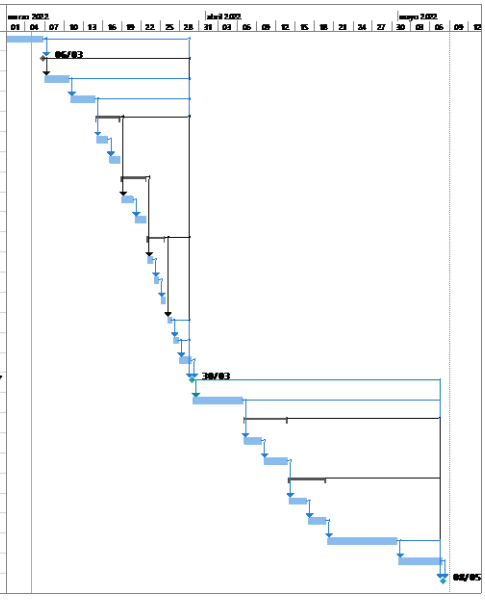

Figura 1. Diagrama de Gantt

Se han ordenado las tareas en un orden cronográfico y se han planificado teniendo en cuenta que una única persona se encargará de la realización de todas ellas. Las estimaciones se han dado en días,

teniendo en cuenta una dedicación media diaria de unas 2-3 horas. Aunque las tareas se han planificado con cierto margen en caso de surgir algún imprevisto en el desarrollo.

1.6. Presupuesto

La estimación de los costes del proyecto está divida en los siguientes apartados:

- Equipo humano.
- Hardware.
- Software.
- Assets.
- Marketing.

El primer apartado, **equipo humano**, es relativamente fácil de calcular. El proyecto tiene una duración estimada de 200 horas de trabajo. Esto equivale un contrato de duración de 2 meses y medio, el cual supone entre 2.500 y 4.000 euros en función del salario y los gastos administrativos.

En la categoría de **hardware** entra todo lo relacionado con el equipo de trabajo. Esto incluye el ordenador de trabajo, pantalla y similares. Como la plataforma de desarrollo en la que se enfoca este trabajo es PC y web, no es necesario incluir hardware de desarrollo de consolas u ordenadores con Mac o Linux. Partiendo de que ya se dispone de este equipo antes de empezar el proyecto, se pueden incluir unos gastos por uso de 100 euros por la duración total de proyecto.

Los gastos relacionados con el **software** son todos aquellos relacionados con las licencias necesarias para desarrollar el videojuego. Las licencias que se tienen que comprar para este trabajo son las de Aseprite (17€) y Photoshop (licencia 3 meses: 60€). El resto de las aplicaciones que se usaran tienen una licencia gratuita o similar, por lo que no se contarán en este trabajo.

A continuación, hay que estimar el gasto por los **assets** usados en el videojuego. La propuesta inicial cuenta que todos los assets que se usarán en el videojuego serán gratuitos. Sin embargo, se reservará un presupuesto de 200 euros, por si se quisiera recurrir al uso de assets de pago.

Por último, no hay que olvidar el **marketing**, ya que es uno de los aspectos más importantes si se quiere lanzar un juego comercial. Ya que el juego desarrollado en este proyecto no es de carácter lucrativo, se destinará un presupuesto menor de alrededor de 200 euros. Este presupuesto se empleará en una campaña de publicidad digital en redes sociales para promocionar el juego y alcanzar un mayor público.

Según estas primeras estimaciones el presupuesto aproximado sería de entre 3000 y 5000 euros, incluyendo un amplio margen de maniobra para afrontar cualquier gasto imprevisto.

2. Estado del arte

Para poder poner este trabajo en contexto es necesario analizar algunos de los juegos sobre los que se toma inspiración y entender cuál son los rasgos principales que definen al género roguelike. Los principales juegos que se han tomado como referencia del roguelike modernos son:

- The Binding of Isaac (Nicalis Inc., 2014).
- Enter the Gungeon (Dodge Roll, 2016).
- Spelunky (Mossmouth, 2008) / Spelunky 2 (Mossmouth, 2020).
- FTL: Faster Than Light (Subset Games, 2012).
- Dead Cells (Motion Twin, 2018).
- Downwell (Fumoto, 2015).
- Hades (Supergiant Games, 2020).

2.1. Definición del género roguelike

La definición del género roguelike es ampliamente debatida en el mundo de los videojuegos. Existe una definición formal denominada "Interpretación de Berlín" (RogueBasin, 2008) que lista los aspectos principales y secundarios que debe de tener un juego roguelike.

El juego desarrollado incluye alguno de los aspectos más importantes como la muerte permanente, la generación aleatoria de niveles y la gestión de recursos.

2.1.1. Muerte permanente

La muerte permanente (*permadeath* en inglés) es uno de los aspectos más importantes que definen a una juego roguelike. Con la muerte del jugador se tiene que empezar la partida desde el principio, perdiendo todo el progreso.

Algunos roguelikes modernos permiten un progreso permanente entre partidas, pudiendo desbloquear distintos elementos del juego:

- Elementos visuales (Figura 2).
- Avanzar a niveles avanzados (Figura 3).
- Variedad de juego (Figura 4).
- Mejoras al personaje (Figura 5).

Figura 2. Paletas de color en Downwell

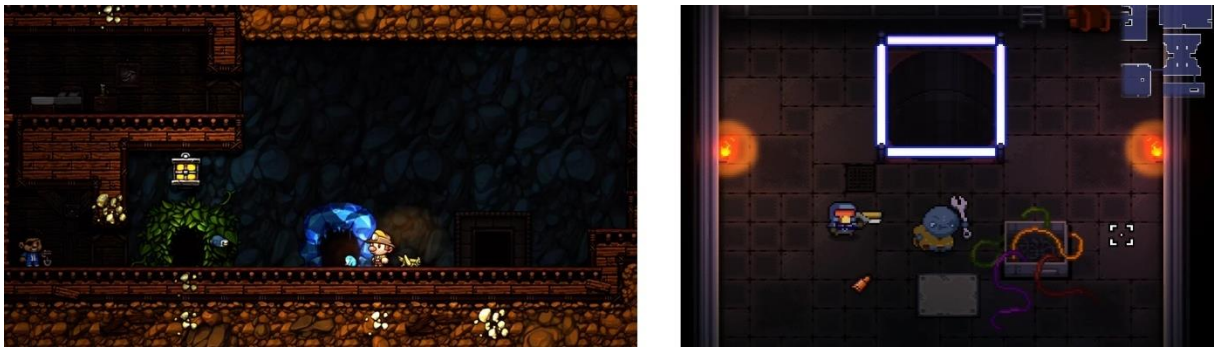

Figura 3. Atajos en Spelunky (izquierda) y Enter The Gungeon (derecha)

Figura 4. Objetos en The Binding of Isaac (izquierda) y naves en FTL: Faster Than Light (derecha)

Figura 5. Mejoras en Dead Cells (izquierda) y Hades (derecha)

En el juego desarrollado se ha implementado una *permadeath* clásica haciendo que todo el progreso se resetee con la muerte del jugador. Lo único que se guarda entre partidas es la puntuación y el tiempo de la partida para marcar los objetivos a superar en el siguiente intento.

2.1.2. Generación de niveles aleatorios

Otro de los aspectos principales de los roguelikes es la aleatoriedad de los niveles. Para ellos al comienzo de una partida los niveles se crean de forma aleatoria. Esto también afecta a los enemigos y objetos que pueblan los niveles.

Dependiendo del tipo de juego, se pueden implementar la generación aleatoria de niveles de muchas formas distintas. Algunos ejemplos de niveles que se pueden conseguir son:

- Serie de habitaciones en rejilla (Figura 6).
- Habitaciones interconectadas (Figura 7).
- Pantalla única (Figura 8).
- Red de encuentros (Figura 9).

Figura 6. Mapa de The Binding of Isaac

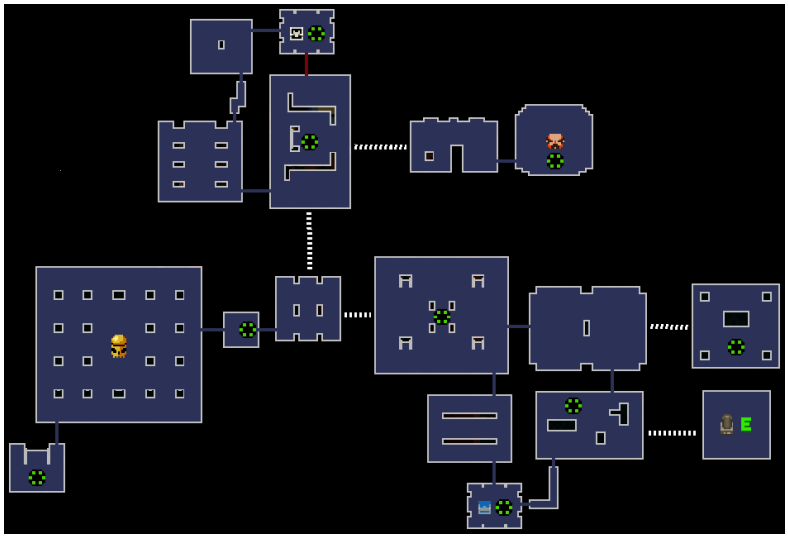

Figura 7. Mapa de Enter the Gungeon

Figura 8. Mapas de Dead Cells (izquierda) y Spelunky (derecha)

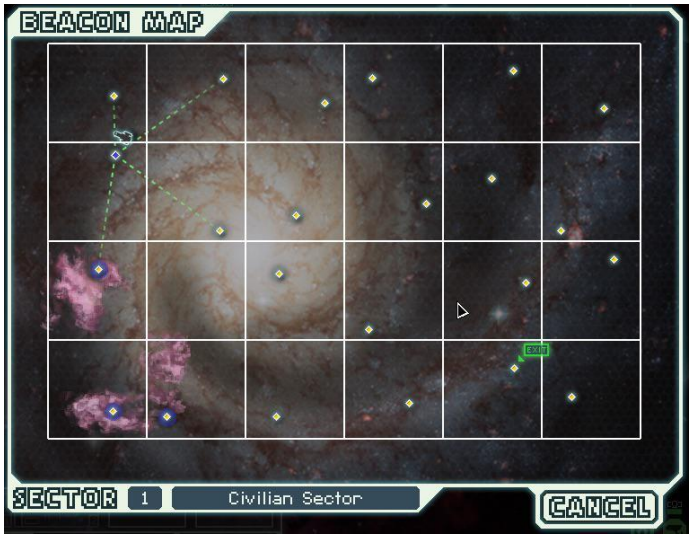

Figura 9. Mapa FTL: Faster Than Light

En el juego que se ha desarrollado se ha decidido tomar un enfoque similar a The Binding of Isaac utilizando una serie de habitaciones predefinidas y colocadas en una rejilla. Esto permite crear mapas aleatorios con gran facilidad partiendo de un número de elementos predefinidos reducido.

2.1.3. Gestión de recursos

El jugador tiene un número reducido de recursos que debe gestionar en cada partida. Estos elementos pueden incluir vida, armas, consumibles o mejoras. El jugador tiene que elegir cuando es el mejor momento para usar estos recursos. También se puede dar la ocasión en el que el juego haga elegir al jugador entre varios recursos y tendrá que elegir cual es el que más le conviene.

La gestión de recursos, juntos con la *permadeath*, hacen que aumente la importancia de cada partida y sean diferentes entre ellas, ya que no se sabe con qué recursos se contará en la siguiente partida.

2.2. Público objetivo

Al desarrollar el juego se ha tenido en mente a un tipo de persona que juega a videojuegos en cortos periodos de tiempo y que busca un juego centrado en las mecánicas y no tanto en la historia.

Este tipo de jugador se alinea muy bien con las mecánicas que ofrecen los roguelikes. Por un lado, estos juegos tienen un enfoque central en las partidas y la repetición. El jugador tiene un tiempo limitado y busca jugar al juego de principio a fin una serie de veces, teniendo una experiencia nueva cada vez.

3. Propuesta

La creación de este juego tiene como objetivo el profundizar en los distintos aspectos de trabajo que engloba al desarrollo de videojuegos. A lo largo del master se ha estudiado múltiples ramas de conocimiento, desde el diseño de los aspectos del juego, hasta la implementación de sistemas interconectados para crear las mecánicas, que a su vez dan pie a dinámicas de juego.

El desarrollo del juego se completará con la publicación de este en una plataforma de venta y distribución como es itch.io, la cual permite crear una página para el juego y distribuirlo de forma sencilla, así como integrar una versión para web en la propia página.

Para conseguir una mayor difusión del juego y conseguir *feedback* variado el juego se impulsará a través de dos plataformas de redes sociales: Twitter y Discord. Se centrará el esfuerzo en mostrar el juego en los ámbitos de desarrolladores de juego que quieran tener acceso al material usado para crear el juego: el código fuente, los assets del juego y la documentación adicional.

3.1. Definición de objetivos

A continuación, se van a detallar algunos objetivos que se quieren alcanzar con este proyecto. Estos objetivos incluyen elementos de diseño del juego y también metas personales relacionadas con la implementación y organización de los componentes creados.

3.1.1. Código fuente

La mayor parte del código utilizado en el proyecto se ha creado desde cero. Se ha intentado crear una estructura clara y simple utilizando varios componentes en función de la funcionalidad que necesiten cubrir. Los componentes se comunican entre ellos usando referencias directas, las cuales se pueden conseguir de varias formas directas: Referencia pública, búsqueda dentro de objeto base, colisión y usando referencias globales. Esta última forma se utiliza para acceder a componentes que solo van a estar instanciados una única vez, como los componentes de control del juego o los del jugador.

Todo el código y assets del juego están colgados en un repositorio en GitHub, el cual será público una vez el juego se termine. La idea es poder usar este repositorio como referencias de los distintos puntos del diseño e implementación del juego y poder trazar el camino que se ha seguido a la hora de implementar las distintas funcionalidades y como se han conectado entre sí.

3.1.2. Niveles

Los niveles del juego son generados aleatoriamente al comienzo de cada partida. Para poder construir los niveles se han creado una serie de habitaciones rectangulares predefinidas, las cuales se tienen marcadas algunos puntos relevantes como los *spawn points* de los enemigos, armas y consumibles. Una vez se hayan colocado el numero de habitaciones deseado en el niveles es necesario marcar cuales son las conexiones entre habitaciones y cuales son paredes por las que no se puede avanzar. Por último, se marca la habitación inicial donde se instanciará al jugador, la habitación final con la salida del nivel y se puebla el nivel con los enemigos y demás objetos.

Una partida está compuesta de varios niveles, los cuales se van aumentando en tamaño y dificulta de los enemigos. El jugador tiene varios objetivos en los niveles:

- Tesoros: Tiene que conseguir la mayor cantidad de tesoro posible antes de descender al siguiente nivel.
- Combate: El jugadores tendrá que vencer a los enemigos de cada habitación para poder avanzar a la siguiente. Con cada golpe recibido su puntuación final se reducirá.
- Tiempo: Cuanto menor sea el tiempo en completar la partida, mayor será la puntuación final.

3.1.3. Armas

Unos de los objetivos personales del juego es crear una serie de armas de combate variadas de forma sencilla. Para ello se ha usado una estructura que define los distintos tipos de armas (espadas, arcos y magia). Esta definición incluye información como el daño, el alcance y las características del proyectil.

Para crear las distintas armas se pueden generar datos aleatorios para cada una de las propiedades usando límites máximos y mínimos, relaciones entre ellas, o se puede usar una lista predefinida con varias armas que estén balanceadas y elegir de forma aleatoria una de la lista. Para este juego se ha decidido tomar este segundo enfoque ya que se quería aprender a crear estas listas predefinidas usando JSON.

Como acabamos de comentar, la ventaja de este segundo método es que se pueden balancear las estadísticas de las armas de forma directa sin tener que crear una lógica compleja que genere valores aleatorios que este balanceados. Otra ventaja de este punto es que, gracias a la estructura organizada y estándar del JSON, es que es muy fácil poder importar estos datos y modificarlos de múltiples formas, ya sea en otros softwares, o de forma programática con scripts.

3.1.4. Control del personaje y feedback

Para realizar el control del personaje se ha querido usar el *Input System* de Unity que permite, de forma muy sencilla, asociar las acciones del juego con entradas del usuario. Esto ha permitido que se puedan usar tanto un teclado como un mando para poder interactuar con el juego. A lo largo del prototipado del juego se ha hecho foco para que la jugabilidad con mando sea buena e intuitiva.

Aparte de la interacción, también se ha experimentado con otros aspectos de control del personaje, como el movimiento y la interacción con otros elementos del juego, como las armas o los enemigos. La idea es dar el mayor feedback posible al jugador y transmitir la información relevante de las distintas acciones que realiza.

3.1.5. Rejugabilidad

Para centrarse en la rejugabilidad del juego se ha implementado un sistema que registra las mejores partidas y las más recientes. De esta forma se puede ofrecer al jugador las estadísticas y que sepa cuál es el objetivo que tiene que superar, y como ha sido su progreso en los últimos intentos.

Para realizar el seguimiento de toda esta información se utilizará un fichero JSON al cual se le irán añadiendo las puntuaciones de cada partida. Toda esta información se procesa al final de la partida para mostrar los datos de la partida actual y hacer la comparativa con las anteriores.

4. Diseño

En este apartado se detallarán las herramientas y recursos utilizados en el desarrollo del juego, así como cuáles han sido las decisiones tomadas para el diseño de este.

4.1. Herramientas

El juego se ha desarrollado usando las siguientes herramientas:

- **Control de versiones**: Se ha utilizado GIT usando el cliente de Windows de SourceTree y Git Bash. El código se ha almacenado en un repositorio público en GitHub.
- **Motor de juegos**: Se ha utilizado Unity 2D versión 2019.
- **Edición de scripts**: Se ha usado Visual Code y Visual Studio 2019 para editar los scripts y hacer debugging del código.
- **Edición de imagen**: Se ha usado Aseprite para crear y editar los sprites del juego.
- **Plataforma distribuidora**: Cada una de las versiones parciales del juego se han subido a Itch.io.

4.2. Recursos empleados

Los recursos que se han usado en el juego son de uso libre o de creación propia. A continuación, se hace un repaso de todos:

- Sprites del jugador. Super Retro World – RPG asset character pack (Gif).
- Sprites de los enemigos. Super Retro World – RPG asset Action Pack (Gif).
- Sprite armas: Kenney Scribble Dungeons (Kenney).
- Sprite pickups: Los pickups, flechas y proyectiles son de creación propia.
- Sprites de la mazmorra: Creación propia.
- Tipografía: Kenney Font Package (Kenney).

Las animaciones del juego se han creado usando la herramienta de Unity de animaciones.

Para realizar el seguimiento del jugador con la cámara se ha utilizado el paquete de Unity Cinemachine, que proporciona diferentes tipos de cámaras virtuales para controlar la Main Camera de Unity.

El texto de la UI utiliza el paquete de Unity TexMeshPro, que permite tener mayor control sobre texto que se quiere mostrar.

Para poder realizar el control del juego mediante teclado o mando se ha utilizado el Input System de Unity que permite de forma sencilla asociar las entradas del jugador a distintas acciones dentro del juego, así como activar y desactivar ciertas acciones mediante código.

4.3. Arquitectura del juego

A continuación, se explicarán los principales elementos desarrollados para el juego, como se han desarrollado y su función dentro del juego.

4.3.1. Animaciones

Se han creados animaciones para distintos elementos del juegos:

Pickups

Las armas, pociones y munición son todas instancias de un objeto común (pickup). Para distinguirlos internamente tienen una variable indicando que tipo son y un valor numérico. Según el tipo del pickup el animator elije la animación correspondiente al objeto adecuado. stun

Enemigos

Para las animaciones de los enemigos se creó un animator base (Figura 10) que implementa la maquina de estados para hacer la transición entre los distintitos estados. Estas animaciones incluyen el movimiento en cuatro direcciones (Movement) (Figura 11), el recibir daño (Hit), y el morir (Die). Cada enemigos después implementa cada una de las animaciones y utiliza el animator base.

Dentro de las animaciones de Hit y Die se utilizan eventos para modificar el comportamientos de los enemigos. En el caso de Hit, el enemigo será aturdido y no podrá moverme durante la duración del aturdimiento. En caso Die, se destruirá el enemigo una vez termine la animación de muerte.

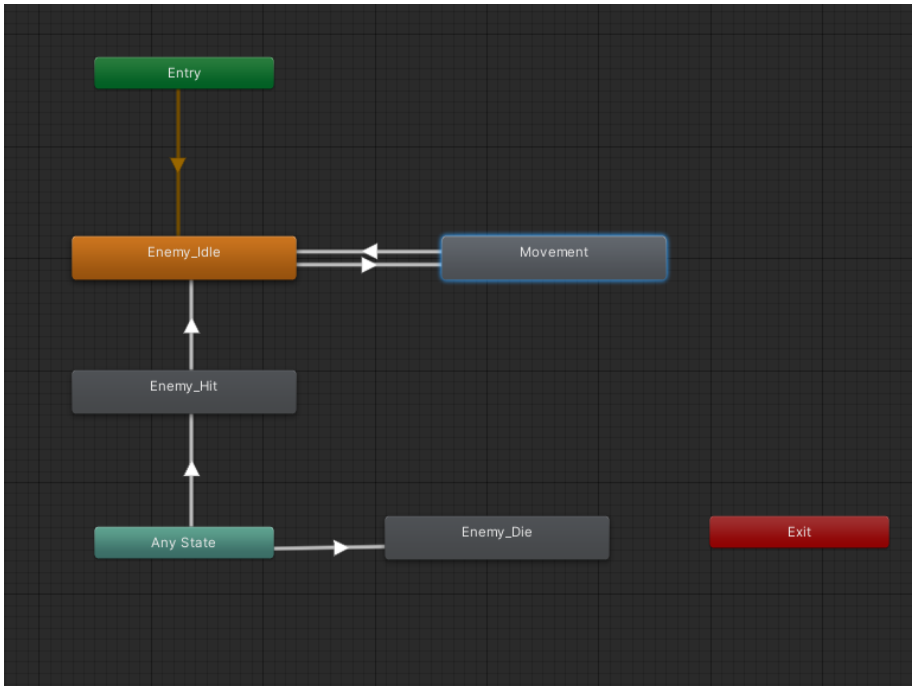

Figura 10. Animator base de los enemigos

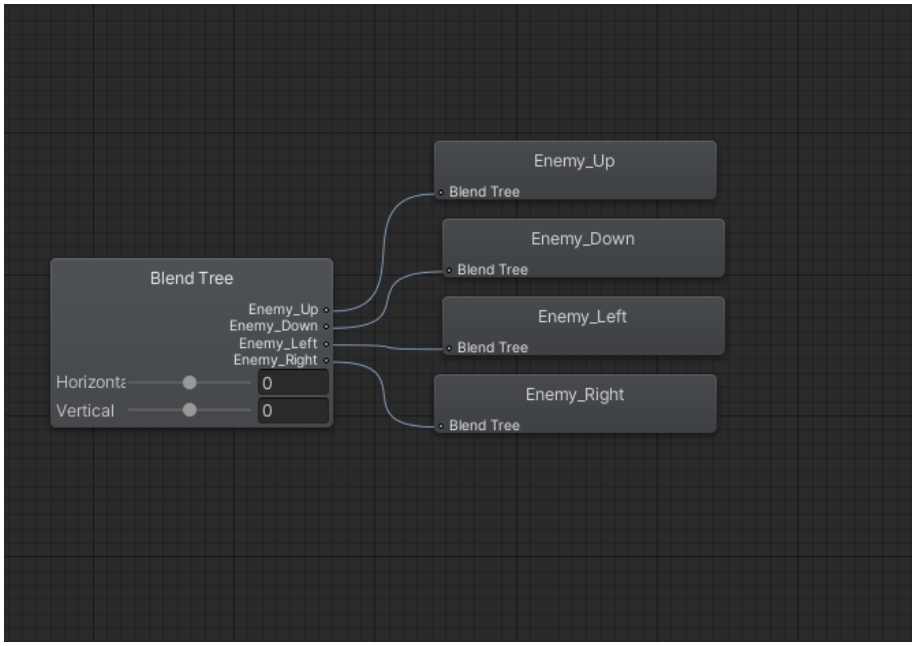

Figura 11. Blend tree del movimiento del enemigo

Jugador

Las animaciones del jugador se han dividido en dos animators, uno para el control del sprite del jugador propiamente dicho, y otro para el control de los ataques. Esta división ha permitido no tener que duplicar las animaciones de movimiento con cada una de las armas.

Movimiento

El animator del movimiento tiene dos estados: inactivo (Idle) o andando (Walk) (Figura 12) y se hace transición entre ambas mediante la velocidad, dirección del movimiento y la orientación del personaje.

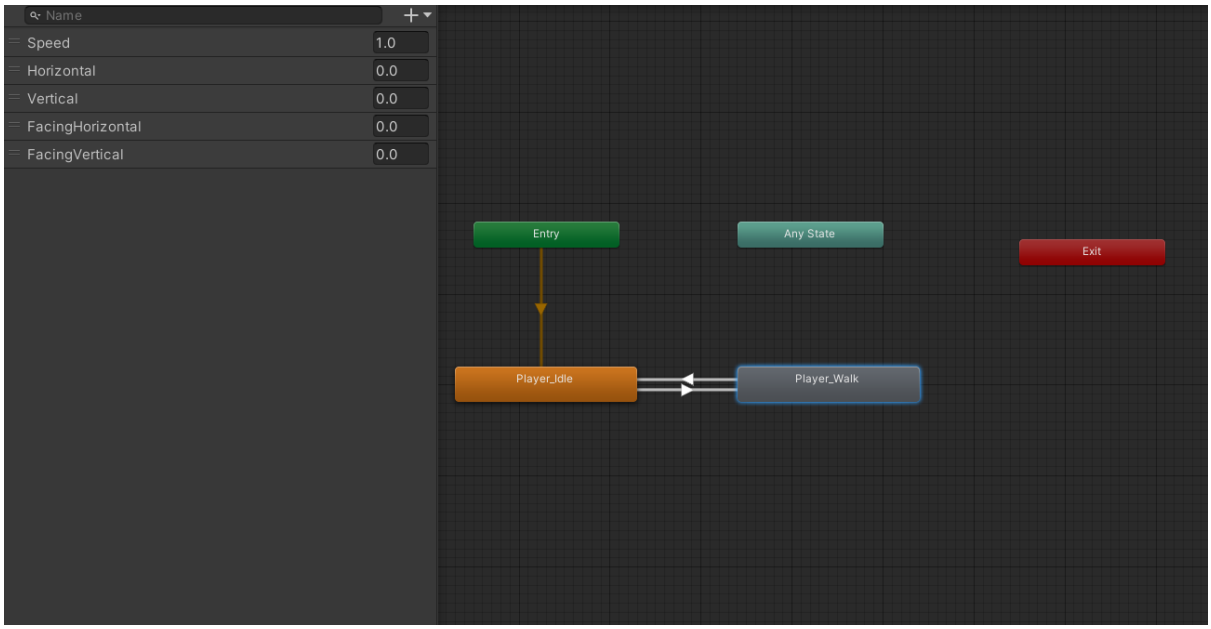

Figura 12. Animator del movimiento del jugador

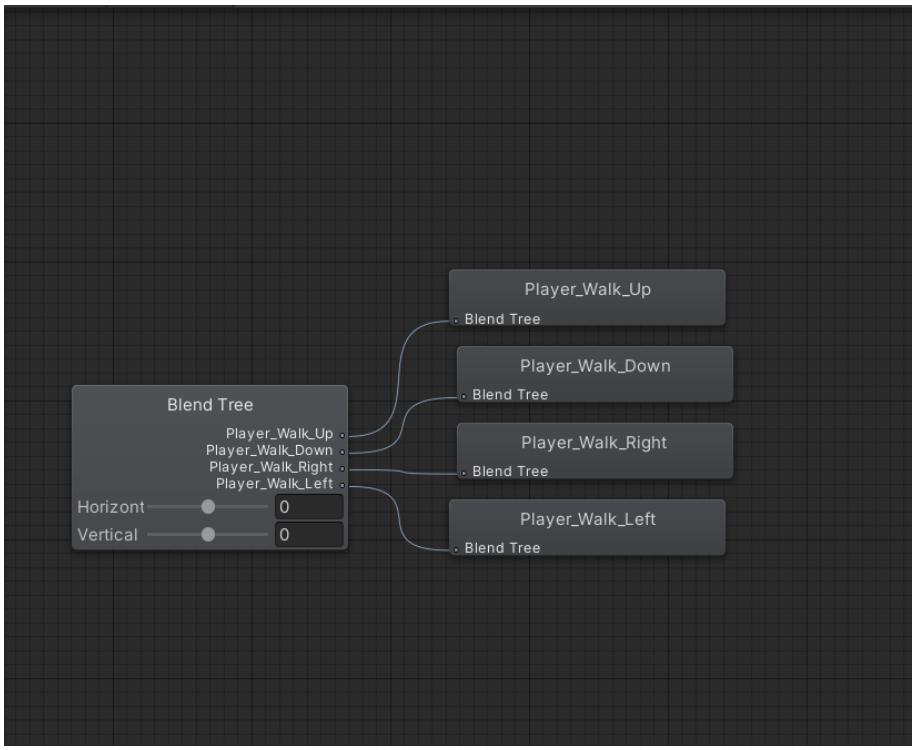

Figura 13. Blend tree del movimiento del jugador

Armas

El control de las animaciones de las armas se realiza con otro animator (Figura 14). En este caso se realiza el control con la orientación del jugador y con triggers para saber que arma se está usando.

Cada una de las animaciones de ataque utiliza eventos para indicar en que punto pueden realizar el daño o disparar el proyectil.

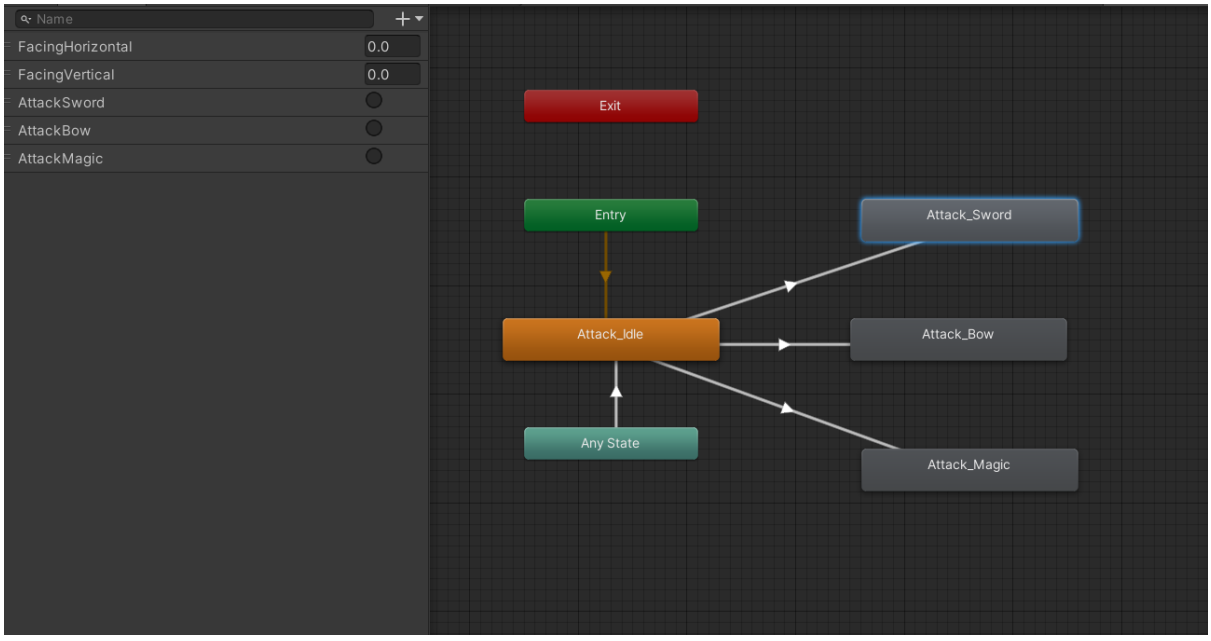

Figura 14. Animator de las armas

4.3.2. IA de los enemigos

La IA de los enemigos hace que cuando se encuentren a una cierta distancia del jugador se dirijan hacia él. Una vez hagan colisión con el jugador le harán daño y le harán retroceder. Si los enemigos son golpeados estos reciben el daño, son aturdidos y sufren un retroceso.

4.3.3. Armas

El juego cuenta con tres tipos de armas: espadas, arcos y magia. El jugador puede tener un arma de cada tipo a la vez y puede cambiar según la situación.

La espada es el arma inicial del juego y tiene la ventaja que es de uso ilimitado. Sirve para el combate cuerpo a cuerpo y es la que más peligro conlleva al tener que acercarse a los enemigos. Cada tipo de espada tiene un alcance y daño diferente.

El arco y la magia son dos armas de larga distancia y ambas requieren de un tipo de munición propia. Por un lado, el arco utiliza flechas, las cuales se pueden conseguir a lo largo de los niveles y se pueden acumular de forma ilimitada. La magia utiliza maná para realizar los hechizos. El mana es limitado y se puede recuperar cogiendo pociones azules por la mazmorra.

4.3.4. Mazmorra

El juego consiste en una mazmorra con diferentes niveles. Cada nivel de la mazmorra tiene una serie de habitaciones dispuestas en una cuadricula. El número de habitaciones por nivel va aumentando según se avanza en la mazmora. Cada una de las habitaciones del nivel puede contener enemigos y pickups como armas y pociones.

Para poder descender al siguiente nivel de la mazmorra es necesario derrotar a todos los enemigos del nivel. Una vez derrotados, la puerta hacia el piso inferior se abrirá. La salida del nivel se situará en la habitación con una única entrada más lejana del comienzo, en caso de no encontrarse ninguna habitación con una única entrada se colocará en la más lejana.

5. Implementación

En este apartado se profundizará en más detalle en los aspectos técnicos de implementación del juego.

5.1. Generación de mazmorra

La creación de la mazmorra se controla por el manager del juego, el cual indica el número de niveles y el tamaño de cada uno. El tamaño de las habitaciones de cada mazmorra se controla en cada mazmorra de forma individual. Aunque cada mazmorra se genera de forma aleatoria se puede elegir el seed inicial para poder reproducir la generación de la mazmorra. Los pasos para generar la mazmorra son los siguientes:

Paso 1. Crear una habitación en la posición (0,0) y marcarla como entrada. Esta habitación será desde la cual el jugador empezará cada mazmorra. Se añaden las posiciones adyacentes libres a una lista de posibles ubicaciones.

Paso 2. Seleccionar de forma aleatoria una de las ubicaciones libres y crear una habitación. Añadir las nuevas ubicaciones posibles a la lista. Recopilar las posibles ubicaciones para hacer spawn de objetos y enemigos. Nota: este paso no se realiza en la habitación principal para no colocar enemigos justo al comienzo.

Paso 3. Repetir paso 2 hasta tener el número de habitaciones deseados.

Paso 4. Usar un algoritmo de relleno (flood-fill) para recorrer todas las ubicaciones empezando desde la principal para marcar las conexiones entre habitaciones.

Paso 5. Crear los suelos, esquinas, paredes y puertas de todas las habitaciones en función de la información recopilada en el paso anterior.

Paso 6. Seleccionar la habitación que tendrá la salida del nivel. Esta habitación será la más lejana a la habitación principal con una sola puerta, en caso de no encontrar ninguna habitación con una sola puerta que no sea la principal se seleccionará la habitación más lejana sin importar el número de puertas.

Paso 7. Colocar la salida en la habitación seleccionada en el paso 6. Antes de colocar la salida hay que eliminar esa posición de la lista de ubicaciones válidas para hacer spawn de enemigos y objetos. La salida estará marcada como cerrada hasta que no se maten a todos los enemigos del nivel.

Paso 8. Hacer spawn de los enemigos y pickups en las posiciones de spawn disponibles. El generador de la mazmorra tiene una serie de porcentajes con los que puede elegir que generar o no generar nada en cada una de las posiciones.

Paso 9. Calcular el tamaño y centro de la mazmorra para colocar un objeto con el exterior y de esta forma no ver el fondo de por defecto de Unity.

El manager del juego creará varias mazmorras siguiendo este algoritmo y las ubicará una encima de otra, pero los niveles que no se están jugando se desactivarán.

5.2. Definición de armas

Como se ha comentado anteriormente, el juego dispone de tres tipos de armas: espadas, arcos y magia. Para definir las características de cada una, como el alcance y el daño de cada una, se ha creado un estructura de datos por tipo de arma. Esto permite generar una gran variedad de armas, ya sea por generación aleatoria y diseñando a mano usando ficheros JSON.

La ventaja de esta implementación es que se desacopla la definición de las armas y la implementación de las mecánicas en el juego. Los ficheros JSON son fáciles de modificar y no suponen tener que reescribir gran parte de los scripts del juego con cada modificación.

Los ficheros JSON con todas las armas definidas son parseados al comienzo del juego y se guardan en una lista. De esta forma, cuando el jugador coje un arma en el juego, lo único que hay que hacer es elegir el arma correspondiente de esta lista en función de un índice proporcionado por el pickup.

6. Conclusiones y líneas de futuro

6.1. Conclusiones

A lo largo de este proyecto se ha podido profundizar en las diferentes áreas de trabajo que son necesarias para crear un videojuego, partiendo desde la planificación hasta la publicación en una plataforma de distribución.

Se ha podido comprobar como la planificación inicial de las tareas ha ayudado a definir el enfoque y alcance del juego. Si embargo, no se han podido desarrollar y pulir todas las tareas detalladas inicialmente, y se ha tenido que reajustar la planificación con el tiempo. Esto da una clara idea de porque dedicar tiempo a esta planificación inicial al principio del proyecto es tan importante, ya que hay que definir y estimar todas las tareas necesarias para crear el juego y añadir suficiente buffer de tiempo para hacer frente a los imprevistos que puedan surgir.

6.2. Futuras mejoras

A continuación, se expondrán las posibles mejoras que se pueden hacer al juego.

6.2.1. Generación de niveles

La generación actual de los niveles utiliza un único tipo de habitación. El primer paso de mejora sería crear varios diseños de habitaciones con diferentes obstáculos. Esto ayudaría al jugador a distinguir fácilmente en que parte de la mazmorra se encuentra cuando está volviendo sobre sus pasos.

También se marcarían diferentes habitaciones con un tags específico para saber que se puede hacer spawn en función del nivel en el que utiliza. Esto permitiría poder distinguir habitaciones de combate, tesoro o suministros, en la actualidad se puede hacer spawn de cualquier objeto en todas las habitaciones.

Las habitaciones también tendrán definidos los posibles puntos de spawn, adaptable al nivel de la mazmorra. De esta forma se puede conseguir una distribución de enemigos y objetos más natural.

También se quiere experimentar con el algoritmo de selección de posiciones de las habitaciones. Actualmente se selecciona una posición aleatoria dentro de la lista de posiciones disponibles. Se quiere probar si modificando la probabilidad de elegir una posición u otra en función de las habitaciones ya elegidas se puede conseguir mazmorras con una distribución con mas ramificaciones en lugar de cluster de habitaciones entorna a la inicial con pequeñas ramificaciones.

Otra forma de modificar el algoritmo sería inicializar varias puntos de la mazmorra en posiciones no contiguas e ir creciendo ambas hasta que se conecten. Para ello también sería necesario modificar las probabilidades se las nuevas posiciones para favorecer la conexión entre las distintas áreas iniciales.

6.2.2. Armas

En el futuro se quiere expandir el sistema de armas para poder dar más variedad a las armas actuales. En el diseño actual las armas de largo alcance funcionan de forma muy similar, y en particular la magia

ofrece pocas ventajas respecto al arco. Una posible mejor que se puede hacer es ampliar los tipos de hechizos que se pueden hacer. De esta forma se pueden incluir hechizos con daño en área y hechizos que provoquen estados en los enemigos como confusión y veneno. También se ampliaría el mana máximo del jugador en función del nivel en el que se encuentra.

Otro aspecto que no se ha podido incluir en esta primera versión es incluir la cadencia de los ataques, tanto cuerpo a cuerpo, como de distancia. Esta es una característica muy importante, ya que, junto con el daño por ataque, permite dar mayor granularidad al diseño de las armas al poder calcular el daño total que se puede infligir por segundo.

6.2.3. Sonido

Uno de los elementos mas importantes del juego que no se ha incluido en el proyecto es el sonido. Principalmente por la falta de tiempo y conocimiento en el área. El sonido es uno de los aspectos más importantes en el desarrollo de un juego, ya que es una parte fundamental del game feel de este.

Bibliografía

Adobe Inc. (1990). Adobe Photoshop. Obtenido de https://www.adobe.com/products/photoshop.html

Dodge Roll. (2016). Enter the Gungeon.

Fumoto, O. (2015). Downwell.

Gif. (s.f.). Obtenido de https://twitter.com/gif_not_jif

Igara Studio S.A. (2001). Aseprite. Obtenido de https://www.aseprite.org/

Kenney. (s.f.). Obtenido de https://kenney.nl/

Leaf Corcoran. (2013). Itch.io. Obtenido de https://itch.io/

Microsoft. (2008). GitHub. Obtenido de https://github.com/

Mossmouth. (2008). Spelunky.

Mossmouth. (2020). Spelunky 2.

Motion Twin. (2018). Dead Cells.

Nicalis Inc. (2014). The Binding of Isaac: Rebirth.

Nintendo EAD. (1991). The Legend of Zelda: A Link to the Past.

RogueBasin. (2008). *Berlin Interpretation*. Obtenido de http://www.roguebasin.com/index.php?title=Berlin_Interpretation

Subset Games. (2012). FTL: Faster Than Light.

Supergiant Games. (2020). Hades.

The Audacity Team. (2000). Audacity. Obtenido de https://www.audacityteam.org/

Torvalds, L. (2005). Git. Obtenido de https://git-scm.com/

Unity Technologies. (2005). Unity. Obtenido de https://unity.com/