Underfox

Autor: Jan Campo-Cossío Reche
Tutor: Gustau Marcos Ballester
Profesor: Joan Arnedo Moreno

Grado de Ingeniería Informática
TFG - Videojuegos

18/06/2023

Créditos/Copyright

Algunos de los sprites utilizados para los enemigos y objetos fueron descargados de la página web “Itch.io”. Estos son libres para su uso personal y comercial.

Los efectos de sonido utilizados para el juego han sido obtenidos del repositorio *online* de pistas de audio “Freesound”. Este permite realizar las descargas de audio bajo dominio público y licencias sin uso comercial.

Los sprites utilizados para el jefe final fueron descargados de la página web “The Sprites Resource” (dedicada principalmente al archivo de sprites e imágenes de diferentes juegos). Permite la descarga y utilización de imágenes bajo permiso de uso no comercial, se han utilizado para todos los estados y ataques del enemigo final.

La fuente utilizada en el juego “Press Start 2P” ha sido descargada de la página web “Google Fonts”. Y esta es libre para su uso comercial en productos y proyectos.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Underfox*
Nombre del autor:	*Jan Campo-Cossío Reche*
Nombre del colaborador/a docente:	*Gustau Marcos Ballester*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega (mm/aaaa):	*18/06/2023*
Titulación o programa:	*Tecnologías de la información*
Área del Trabajo Final:	*Videojuegos*
Idioma del trabajo:	*Español*
Palabras clave	Videojuego, plataformas 2D, Teoría de grafos

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

Mi trabajo se centra en la creación de un videojuego, titulado “Underfox”, del género plataformas en 2D. Este, además de incluir los elementos y mecánicas típicos de un juego de plataformas, contará con un sistema, en algunos niveles concretos, que generarán el escenario de manera procedural. El cual empleará como base la teoría de grafos para conseguirlo.

El objetivo que pretendo alcanzar mediante este es conocer y documentar todas las etapas de la realización de un videojuego. Concretamente, los pasos seguidos desde su “Concepción” hasta su “Lanzamiento”; todos los roles que intervienen en el proceso; las dificultades y problemas con los que nos hemos encontrado, y toda la experiencia adquirida en comparación con la inicial.

Como herramienta principal para su desarrollo emplearé un “*Game engine*”. Entre otras razones, estos marcos de software facilitan el desarrollo de juegos al incorporar librerías, objetos, y un acceso directo a toda su documentación. Además, otra razón importante, que ya se explicará más adelante es que la intención es exportar el producto final a diferentes plataformas. Y muchos de los “*Game engines*” actuales permiten realizar esta exportación fácilmente.

El propósito final de todo el proyecto será conseguir una versión pulida y jugable del trabajo. Como el tiempo máximo para su creación es el de un semestre, procuraré establecer unos objetivos factibles acordes a la fecha de entrega. Así, como ciertos aspectos del juego todavía faltan por concretarse, y no dispongo de un período de realización ilimitado, he empleado la metodología de desarrollo de software ágil.

GitHub.

Vídeos de demostración.

Abstract (in English, 250 words or less):

My work is focused on the creation of a videogame, titled “Underfox”, of the 2D platform genre. This, in addition to including the typical elements and mechanics of a platform game, will have a system, in some specific levels, that will generate the environment procedurally. Which will use as a basis the theory of graphs to achieve it.

The objective that I intend to reach is to know and document all the stages of the making of a video game. Specifically, the steps followed from its "Conception" to its "Launch"; all the roles involved in the process; the difficulties and problems we have encountered, and all the experience gained compared to the initial one.

As the main tool for its development, I will use a game engine. Among other reasons, these software frameworks makes easier the game development process by incorporating libraries, objects, and direct access to all their documentation. Also, another important reason, which will be explained later, is that the intention is to export the final product to different platforms. And many of the current game engines allow this export to be done without many troubles.

The ultimate goal of the entire project will be to get a polished and playable version of the work. As the maximum time for its creation is one semester, I will try to establish feasible objectives according to the delivery date. Thus, as certain aspects of the game have yet to be finalized, and I do not have an unlimited period of realization, I have used the agile methodology for software development.

Dedicatoria

A mis padres y mi hermana, las personas que me han ayudado y alentado a lo largo de mis años de carrera.

¡Muchas gracias por todo vuestro apoyo!

Abstract

My work is focused on the creation of a videogame, titled “Underfox”, of the 2D platform genre.

The objective that I intend to achieve through this is to know and document all the stages of the making of a video game. Specifically, the steps followed from its "Conception" to its "Launch"; all the roles involved in the process; the difficulties and problems we have encountered, and all the experience gained compared to the initial one.

As the main tool for its development, I will use a "Game engine". Among other reasons, these software frameworks makes easier the game development process by incorporating libraries, objects, and direct access to all their documentation. Also, another important reason, which will be explained later, is that the intention is to export the final product to different platforms. And many of the current "Game engines" allow this export to be done without many troubles.

The ultimate goal of the entire project will be to get a polished and playable version of the work. As the maximum time for its creation is one semester, I will try to establish feasible objectives according to the delivery date. Thus, as certain aspects of the game have yet to be finalized, and I do not have an unlimited period of realization, I have used the agile methodology for software development. Link to the game.

Keywords

Videogame, 2D Platforms, TFG, Graph theory

Resumen

Mi trabajo se centra en la creación de un videojuego, titulado “Underfox”, del género plataformas en 2D.

El objetivo que pretendo alcanzar mediante este es conocer y documentar todas las etapas de la realización de un videojuego. Concretamente, los pasos seguidos desde su “Concepción” hasta su “Lanzamiento”; todos los roles que intervienen en el proceso; las dificultades y problemas con los que nos hemos encontrado, y toda la experiencia adquirida en comparación con la inicial.

Como herramienta principal para su desarrollo emplearé un “Game engine”. Entre otras razones, estos marcos de software facilitan el desarrollo de juegos al incorporar librerías, objetos, y un acceso directo a toda su documentación. Además, otra razón importante, que ya se explicará más adelante es que la intención es exportar el producto final a diferentes plataformas. Y muchos de los “Game engines” actuales permiten realizar esta exportación fácilmente.

El propósito final de todo el proyecto será conseguir una versión pulida y jugable del trabajo. Como el tiempo máximo para su creación es el de un semestre, procuraré establecer unos objetivos factibles acordes a la fecha de entrega. Así, como ciertos aspectos del juego todavía faltan por concretarse, y no dispongo de un período de realización ilimitado, he empleado la metodología de desarrollo de software ágil. Enlace al juego.

Palabras clave

Videojuego, Plataformas 2D, TFG, Teoría de grafos

Notaciones y Convenciones

Los tecnicismos que no se han traducido y/o no pertenecen a un nombre aparecen escritos en inglés y letra cursiva.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1.Introducción

1.1. Justificación del área elegida

El trabajo de fin de grado es sin duda el proyecto más importante que se desarrollará en una carrera. Es la prueba definitiva, en la que se debe demostrar tanto tu conocimiento como tus capacidades adquiridas hasta ahora. Por esta razón, parece lógico seleccionar un área motivadora y que, en este caso, incluso fue uno de los motivos para escoger esta ingeniería.

Desde el punto de vista de un proyecto, la realización de un videojuego involucra todas las fases del proceso de desarrollo de software. Desde su conceptualización y definición, pasando por el desarrollo e implementación de sus componentes, hasta su lanzamiento final. En mi caso, es una oportunidad para experimentar por primera vez la gestión de software desde una perspectiva personal. Es por todo ello por lo que creo que el diseño y desarrollo de un videojuego es una elección perfecta como área para mi proyecto de fin de grado.

Personalmente, los videojuegos siempre han sido de gran interés para mí. Cuando era pequeño me divertía con ellos. Y ahora de adulto, además de seguir disfrutando con ellos, soy capaz de apreciarlos desde una perspectiva más técnica. Mejor dicho, ahora soy capaz de entender mejor las funciones y componentes que se ocultan tras las acciones y mecánicas de los elementos de un juego. Y es este último hecho el que me ha alentado a crear mi propio producto a partir de mis conocimientos.

Además, sería importante mencionar que los videojuegos son una industria que, desde hace ya casi una década, ha tenido un crecimiento económico constante. En general, tras la pandemia de la COVID-19, el ocio digital ha experimentado un auge. En 2021 la Asociación española de videojuegos (AEVI) declaró en su anuario que esta industria facturó 1.795 millones de euros en España [1], [2]. Y, según el informe del Global Entertainment and Media Outlook 2022-26 de PWC, se espera que la industria mundial alcance un valor de 321 millones de dólares a nivel mundial [3]. De manera que se trata de una industria que continuará creciendo y que, por lo tanto, va a resultar muy interesante profesionalmente.

Y, como curiosidad, el crecimiento de la industria de los videojuegos ha permitido que las producciones de los juegos sean cada vez más profesionales. La evolución de las mejoras gráficas y de computación han permitido que cada nuevo lanzamiento busque acercarse cada vez más al estilo y narrativa del cine. En consecuencia, el presupuesto de estas producciones ha crecido considerablemente. Hasta el punto de que, para ciertos lanzamientos de compañías profesionales, se hablan de decenas de millones invertidas en su desarrollo.

La intención tras este proyecto no es terminar consiguiendo un producto innovador para la industria del videojuego. Sino en profundizar en el proceso de elaboración de un producto de software desde cero, a la vez que se adquirirá práctica en este campo. Ya que, como punto de partida, como autor del trabajo no poseo demasiada experiencia en el área de desarrollo de software. Esta sería la primera vez en la que tuviera que planear y ejecutar un proyecto de esta envergadura hasta obtener un producto final. Y, orientar este proceso a un entorno

más creativo, como lo es la realización de un videojuego, conseguiría un factor motivacional importante.
Así, de manera resumida, el objetivo mediante la realización de este proyecto es el de realizar el diseño, desarrollo de un videojuego, y dejarlo preparado para su distribución.

1.2. Descripción

El videojuego que se tiene pensado desarrollar es del género de plataformas 2D. Es decir, la mecánica principal consiste que el jugador maneje al personaje protagonista haciéndolo atravesar una serie de escenarios, y sortear un conjunto de obstáculos y enemigos hasta alcanzar el objetivo final del juego.

Asimismo, como característica especial para este proyecto, se ha decidido hacer más interesante la experiencia para el jugador implementando una función que generará de forma procedural los escenarios de determinados niveles. Es decir, cada vez que el jugador acceda al nivel se encontrará con una distribución diferente de los obstáculos y la salida. Para ello, se ha utilizado la teoría de grafos como base para su concepción. Haciendo que las estancias a las que el jugador pueda acceder actúen como nodos. Cuya accesibilidad con sus vecinos se decidirá por una versión del algoritmo de búsqueda DFS (Depth First Search). Todo este proceso se explicará de manera mucho más detallada en el apartado de diseño.

La estética general del juego se ha pensado para servir como homenaje a los primeros juegos de plataformas creados. Por tanto, ha habido un esfuerzo en tratar de mantener una estética pixel art a lo largo de todo el juego, y utilizar ciertos sonidos de sintetizador para el audio que evocasen aquellos de las primeras generaciones de consolas. No obstante, debido a cierta falta de experiencia y de tiempo, cabe admitir que ha habido un par de excepciones. Como, por ejemplo, los fondos creados para el menú principal o el de niveles han sido creados en una resolución más alta. O, para el sonido de ciertos elementos como los Checkpoints o los ataques del jefe final se han utilizado pistas de audio normales.

Como título se ha escogido “Underfox”, un juego de palabras del término inglés “Underdog” (usado para referirse a las víctimas de injusticia y/o perseguidas). Se seleccionó este término debido a que parecía que casaba bien con el contexto original del juego. El cual, giraba en torno a la llegada de una flota de máquinas que pretendían deforestar un bosque. Y uno de sus habitantes, un zorro, indignado con esta situación, decide tomar la situación por su mano y acabar con todas ellas él solo.

Para el transcurso del juego, la idea principal era que el jugador atravesara una serie de niveles en los que los enemigos y los obstáculos fuesen variando al avanzar de forma progresiva. Sin embargo, dado que el tiempo para la realización del proyecto es limitado, se ha tenido que racionar la cantidad de niveles a completar. Así, como resultado final el juego ha quedado con un número de 6 niveles completos. De los cuales, el primero sirve como tutorial y los otros alternan entre niveles de superficie con secciones procedurales, y niveles completamente procedurales.

Como resultado final, y a modo de resumen, se ha obtenido un producto totalmente funcional, pulido y que cumple con las expectativas que este tipo de videojuegos ofrecen. Y, aunque no es largo o demasiado complejo, es entretenido de jugar y asegura al usuario pasárselo bien.

Además, al principio se contempló la posibilidad de distribuirlo entre las plataformas de juego más populares, Pc y Smartphone, por la facilidad de exportación que se posee al crear el juego con un *Game Engine*. No obstante, debido a que la magnitud del proyecto ha sido superior a la esperada en un principio, solo se ha preparado para distribuirlo a la plataforma Windows. A pesar de esto, no se ha negado la posibilidad de exportar el juego a plataformas móviles en el hipotético caso de que tuviera una buena acogida en Pc.

1.3. Objetivos generales

Cuando termine la realización del trabajo, los objetivos finales que se pretenden alcanzar serían los siguientes:

1.3.1. Objetivos principales

Objetivos de la aplicación/producto/servicio:

- Obtener una versión del juego sólida y sin errores.
- Crear un sistema que cree escenarios aleatorios para determinados niveles.
- Crear una manejabilidad del personaje fácil y agradable para jugar.
- Crear un mínimo de 10 niveles diferentes.
- Crear una IA básica para los distintos enemigos.
- Crear un conjunto de objetos con los que el jugador pueda interactuar durante la partida.

Objetivos para el cliente/usuario:

- Procurar que disfrute de una interacción con el juego que resulte cómoda y fluida.
- Procurar que el entorno de juego sea satisfactorio, pero no demasiado fácil para que genere en el jugador cierto “gancho”.
- Procurar que la historia sea coherente y con diferentes fases.

Objetivos personales del autor del TF:

- Adquirir experiencia tras haber planeado y gestionado un trabajo de software.
- Haberse familiarizado más con el entorno *Game engine* empleado.
- Obtener experiencia en el ámbito de programación de videojuegos creando este primer juego.

1.3.2. Objetivos secundarios

Objetivos adicionales que enriquecen el TF.

- Conseguir que la inteligencia de los enemigos tenga la capacidad suficiente para que estos se adapten a los escenarios en los que se encontrarán con el jugador.
- Conseguir un nivel de dificultad asequible para el jugador (uno que no sea fácil para que no sea aburrido, pero tampoco sea demasiado difícil para que no abandone el juego).
- Alcanzar de un diseño de niveles bien planteado para crear mapas con secciones interesantes y divertidos de jugar.

1.4. Metodología y proceso de trabajo

Entre las posibles estrategias sugeridas para realizar el trabajo, se ha escogido desarrollar un nuevo juego desde cero. Y, como ya se mencionó anteriormente, el autor de este trabajo no poseía demasiada experiencia en el ámbito del desarrollo de un videojuego, o el de desarrollo de software en general. Así, realizar toda esta primera aproximación por su parte, ha sido una manera de obtener experiencia tanto en el ámbito del desarrollo del software como en el de desarrollo de videojuegos.

Para alcanzar el objetivo, se propuso una idea para el producto y un plan general explicado en el apartado de “Planificación”. Para el desarrollo se optó por utilizar un motor de juegos (o *Game Engine*). Ya que, estos incluyen una gran variedad de elementos y herramientas que agilizan el desarrollo del proyecto (como librerías de físicas o sistemas de audio y sonido), y en general, facilitan mucho más la creación de un juego que realizando toda la programación desde cero.
Para este trabajo se eligió Unity porque, como ya se explicará más adelante, poseía las características que mejor se adaptaban al juego, y permitía al autor aplicar parte su experiencia con un lenguaje de programación orientado a objetos al utilizar C#. Para el proceso de desarrollo se ha estudiado la documentación y las guías oficiales de la página web de Unity; además de los foros disponibles para la comunidad de este motor, en el que se han consultado ciertos problemas y dudas que han aparecido durante el transcurso del proyecto.

Finalmente, la metodología de desarrollo de software escogida para este proyecto ha sido Ágil. Esta es una metodología iterativa, en la que la carga de trabajo total se dividide en entregas de valor periódicas y continuas con un número concreto de objetivos a alcanzar en cada una. Se tomó esta decisión porque, este es un tipo de metodología muy flexible que prioriza dar respuesta a los cambios inmediatos por encima de seguir un plan estrictamente fijo. Y, tal y como estaba planteado el proyecto, ciertos objetivos estaban claros y bien definidos de cara al producto final, pero también existía cierta incertidumbre sobre otros aspectos. Como las mecánicas que se implementarían en el juego con unos conocimientos no demasiado extensos, la organización de ciertos elementos de la arquitectura general, o el alcance total que debía tener. Los cuales han ido modificándose a lo largo de todo el desarrollo.

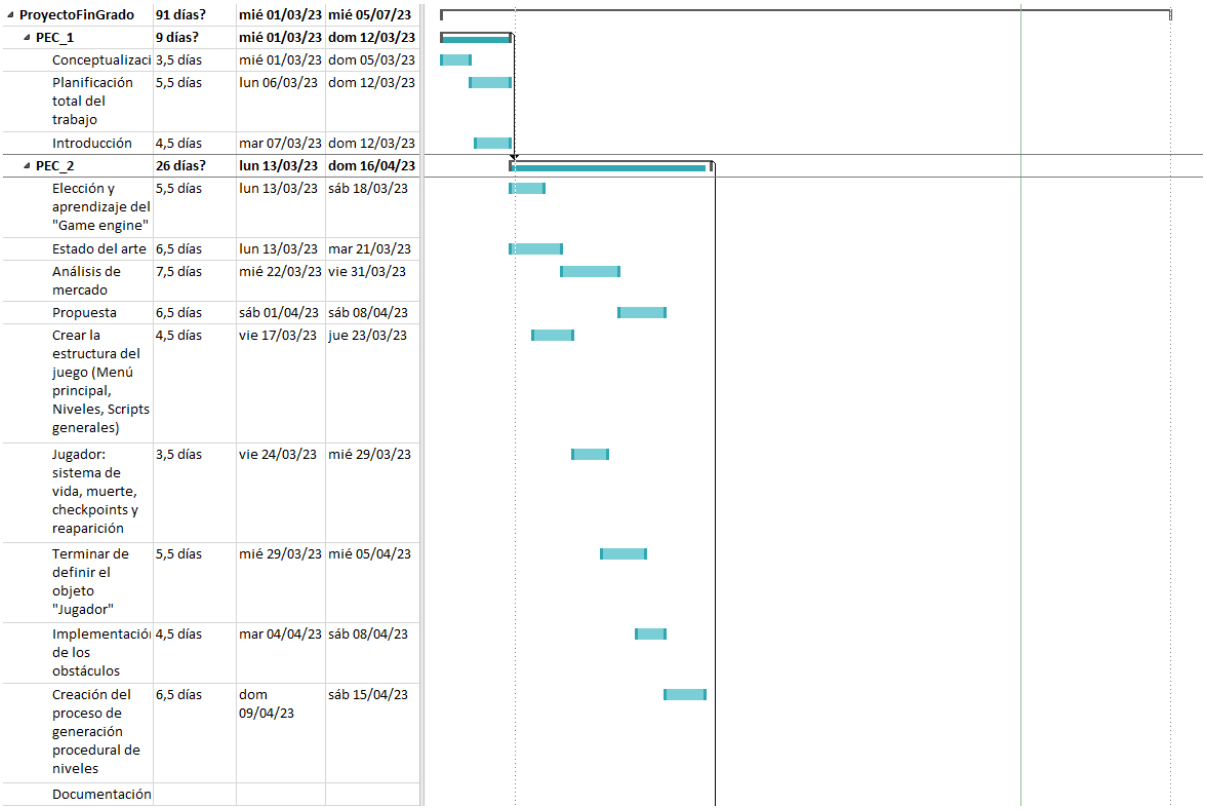
Por todas estas razones, para este proyecto en concreto la Ágil parecía ser la mejor metodología que aplicar en comparación a otras como Waterfall, por ejemplo. Ya que, esta exigía tener un concepto muy claro de cómo debía ser el producto final, y no daba demasiado margen a los cambios que pudiesen suceder durante el desarrollo de este.

1.5. Planificación

Para asegurar que el trabajo se realice a un ritmo adecuado, y se garantice una calidad mínima en cada fase, se ha decidido basar las fechas claves en la entrega de las PECs propuestas. Es decir, se ha decidido dividir la carga de trabajo total repartiendo las tareas indicadas para cada PEC.

1.5.1. Planificación inicial

Como se mencionaba anteriormente, tanto el concepto como los diferentes elementos del proyecto han ido modificándose a lo largo de todo el proceso. De forma que, los primeros periodos de realización de las actividades entregables eran los siguientes:



ProyectoFinGrado	**91 días?**	**mié 01/03/23**	**mié 05/07/23**
PEC_1	**9 días?**	**mié 01/03/23**	**dom 12/03/23**
Conceptualizaci	3,5 días	mié 01/03/23	dom 05/03/23
Planificación total del trabajo	5,5 días	lun 06/03/23	dom 12/03/23
Introducción	4,5 días	mar 07/03/23	dom 12/03/23
PEC_2	**26 días?**	**lun 13/03/23**	**dom 16/04/23**
Elección y aprendizaje del "Game engine"	5,5 días	lun 13/03/23	sáb 18/03/23
Estado del arte	6,5 días	lun 13/03/23	mar 21/03/23
Análisis de mercado	7,5 días	mié 22/03/23	vie 31/03/23
Propuesta	6,5 días	sáb 01/04/23	sáb 08/04/23
Crear la estructura del juego (Menú principal, Niveles, Scripts generales)	4,5 días	vie 17/03/23	jue 23/03/23
Jugador: sistema de vida, muerte, checkpoints y reaparición	3,5 días	vie 24/03/23	mié 29/03/23
Terminar de definir el objeto "Jugador"	5,5 días	mié 29/03/23	mié 05/04/23
Implementació	4,5 días	mar 04/04/23	sáb 08/04/23
de los obstáculos			
Creación del proceso de generación procedural de niveles	6,5 días	dom 09/04/23	sáb 15/04/23
Documentación			

PEC_3	**26 días?**	**lun 17/04/23**	**dom 21/05/23**
Diseño	20,5 días	lun 17/04/23	dom 14/05/23
Creación del objeto enemigo	9,5 días	lun 17/04/23	vie 28/04/23
Crear niveles 1-3 (Temática de bosque)	5,5 días	lun 17/04/23	lun 24/04/23
Crear niveles 4-6 (Temática de playa)	6,5 días	mar 25/04/23	mié 03/05/23
Crear niveles 7-9 (Temática de fábrica)	6,5 días	jue 04/05/23	vie 12/05/23
Crear nivel 10, de jefe final (Temática de torre)	5,5 días	sáb 13/05/23	vie 19/05/23
Cinemáticas del juego	10,5 días	lun 08/05/23	dom 21/05/23
Documentación hasta el momento, examinada y corregida	25,5 días	lun 17/04/23	dom 21/05/23
PEC_4	**21 días?**	**lun 22/05/23**	**dom 18/06/23**
Implementació	5,5 días	lun 22/05/23	dom 28/05/23
Demostración	5,5 días	lun 29/05/23	lun 05/06/23
Líneas de futuro y conclusiones	6,5 días	mar 06/06/23	mié 14/06/23
Corrección de posibles incoherencias y errores en el código	7,5 días	jue 01/06/23	lun 12/06/23
Últimas mejorías en el apartado gráfico	7,5 días	mié 07/06/23	vie 16/06/23
Documentación terminada y revisada	11,5 días	mar 30/05/23	mié 14/06/23
Semana_de_reces	7 días	lun 19/06/23	mar 27/06/23
PEC_5	**6 días?**	**mié 28/06/23**	**mié 05/07/23**
Defensa ante el tribunal	5,5 días	mié 28/06/23	mié 05/07/23

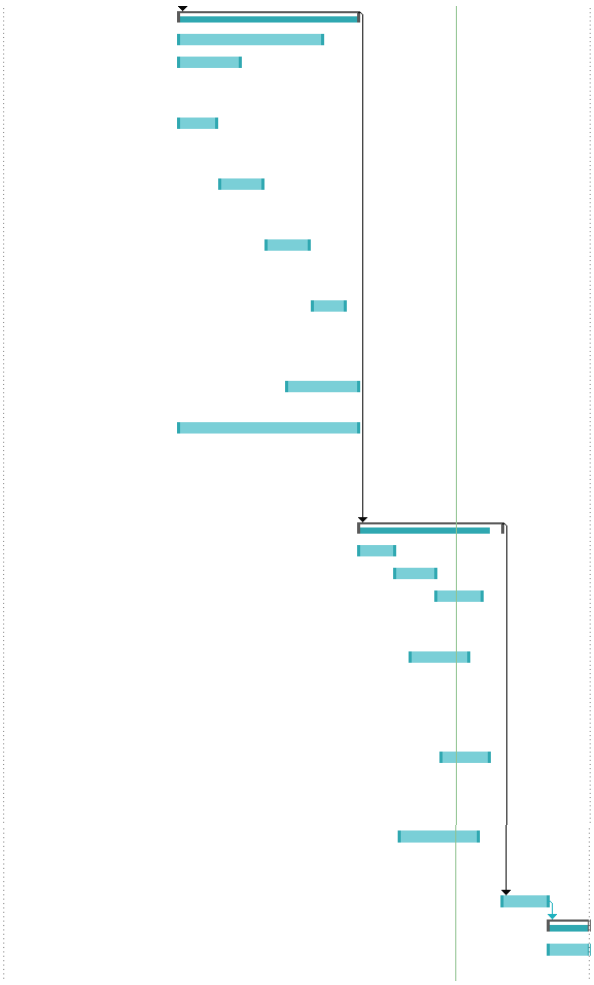


Figura 1: Diagrama de Gantt de la planificación inicial del proyecto

Básicamente, se componía en:

PEC 1 Diseño (3/2/2023)->(12/3/2023)

- Hallar y definir la idea del juego
- Conceptualización
- Introducción
- Planificación total del trabajo

PEC 2 Versión parcial (13/3/2023)->(16/4/2023)

- Análisis de mercado
- Propuesta
- Elección y aprendizaje del Game engine.
- Creación de una pequeña “Testroom” para las pruebas.
- Creación del objeto jugador:

    - Sistema de movimiento (andar y correr).
    - Sistema de recogida de Items.
    - Sistema de habilidad (doble salto, agacharse, subir escaleras y rebotar en las paredes).
    - Animaciones.
    - Sistema de vida.
    - Sistema de ataque mediante disparo.
- Creación del objeto enemigo:
    - Máquina de estados.
    - Sistema de movimiento.
    - Animaciones.
    - Seguir al jugador.
    - Sistema de vida.
    - Mecánica de disparo o ataque.
- Sistema de Interfaz visual del jugador:
    - Sistema de vida (apartado visual).
    - Sistema de recuento de puntos.
- Documentación:
    - Estado del arte
    - Análisis de mercado
    - Propuesta

PEC3 Versión jugable (17/4/2023)->(21/5/2023)

- Mapa exterior que permitirá acceder a los otros niveles al cruzarlo.
- Crear niveles 1-3 (Temática de playa):
    - Diseño.
    - Creación del nivel.
    - Creación del jefe del último nivel.
    - Pruebas.
- Crear niveles 4-6 (Temática de bosque):
    - Diseño.
    - Creación del nivel.
    - Creación del jefe del último nivel.
    - Pruebas.
- Crear niveles 7-9 (Temática de Fábrica):
    - Diseño.
    - Creación del nivel.
    - Creación del jefe del último nivel.
    - Pruebas.

- Crear nivel 10, de jefe final (Temática de torre):
    - Diseño.
    - Creación del nivel.
    - Pruebas.
- Instalación.
- Demostración.
- La documentación elaborada hasta el momento estará examinada y corregida.
-

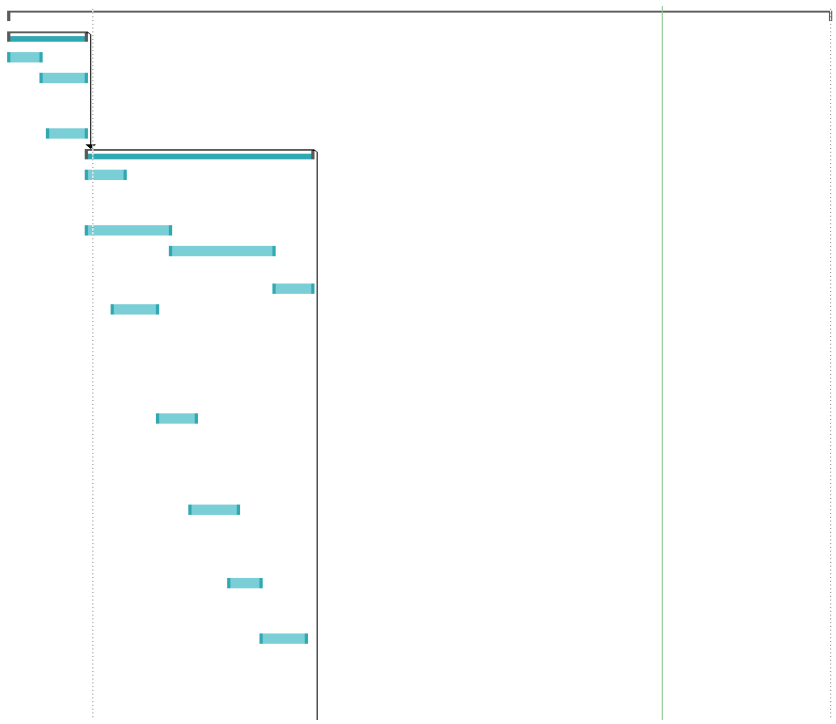
PEC4 Memoria y productos finales (22/5/2023)->(18/6/2023)

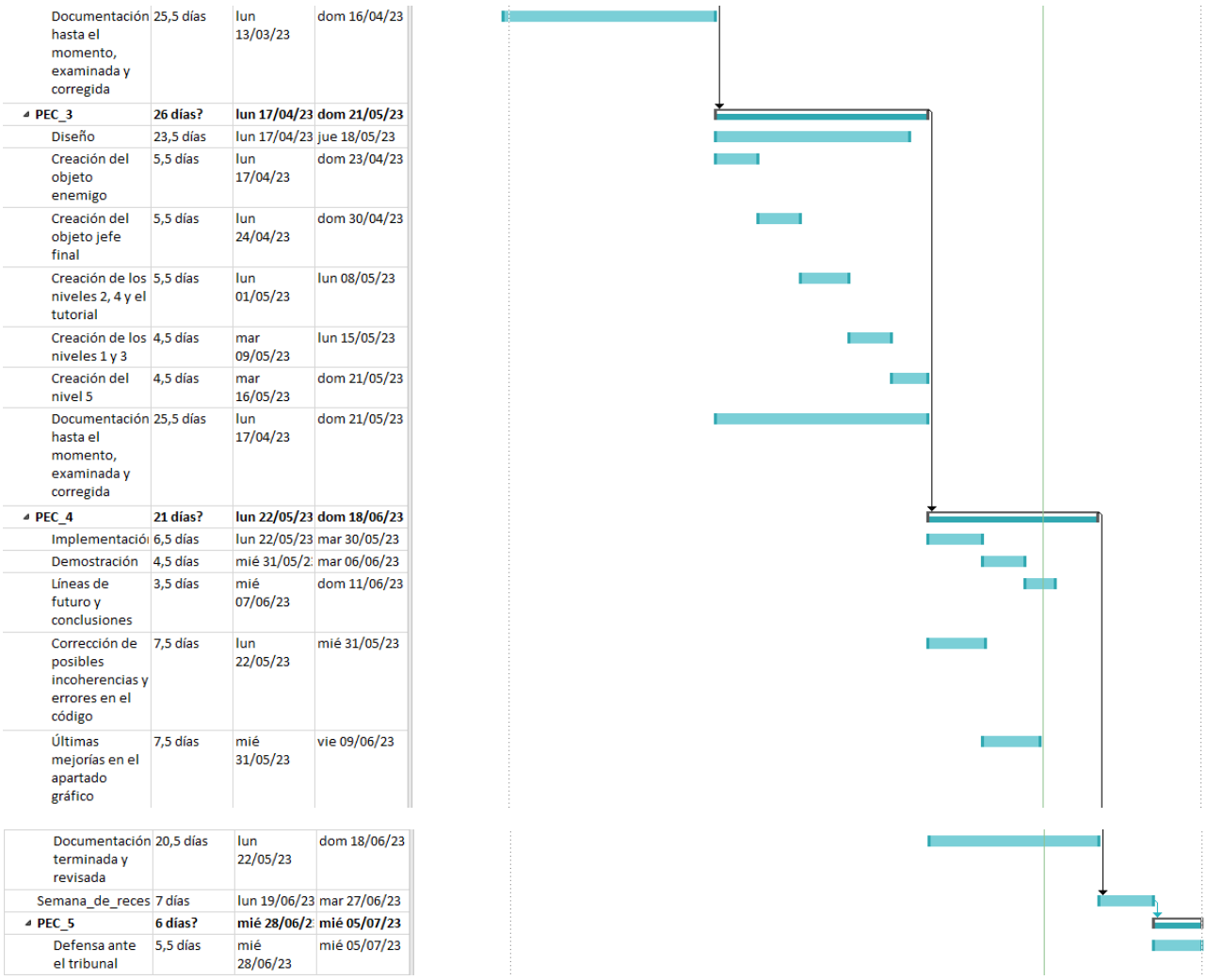
- Instalación.
- Demostración.
- Conclusiones y líneas de futuro.
- Corrección de posibles incoherencias o errores en el código.
- Últimas mejoras en el apartado gráfico.
- La documentación elaborada hasta el momento estará examinada y corregida.

1.5.2. Desarrollo real del proyecto

Como resultado final del proyecto, se produjeron diferentes cambios y modificaciones a lo largo de la realización de la PEC 2 y la PEC 3. De manera que la planificación del desarrollo real del proyecto ha quedado de esta manera.

ProyectoFinGrado	**91 días?**	**mié 01/03/2**	**mié 05/07/23**
PEC_1	**9 días?**	**mié 01/03/2**	**dom 12/03/23**
Conceptualizaci	3,5 días	mié 01/03/2	dom 05/03/23
Planificación total del trabajo	5,5 días	lun 06/03/23	dom 12/03/23
Introducción	4,5 días	mar 07/03/2	dom 12/03/23
PEC_2	**26 días?**	**lun 13/03/23**	**dom 16/04/23**
Elección y aprendizaje del "Game engine"	5,5 días	lun 13/03/23	sáb 18/03/23
Estado del arte	10,5 días	lun 13/03/23	sáb 25/03/23
Análisis de mercado	11,5 días	dom 26/03/23	lun 10/04/23
Propuesta	4,5 días	mar 11/04/2	dom 16/04/23
Crear la estructura del juego (Menú principal, Niveles, Scripts generales)	4,5 días	vie 17/03/23	jue 23/03/23
Jugador: sistema de vida, muerte, checkpoints y reaparición	3,5 días	vie 24/03/23	mié 29/03/23
Terminar de definir el objeto "Jugador"	5,5 días	mié 29/03/23	mié 05/04/23
Implementació de los obstáculos	4,5 días	mar 04/04/23	sáb 08/04/23
Creación del proceso de generación procedural de niveles	6,5 días	dom 09/04/23	sáb 15/04/23





Documentación hasta el momento, examinada y corregida	25,5 días	lun 13/03/23	dom 16/04/23
PEC_3	**26 días?**	**lun 17/04/23**	**dom 21/05/23**
Diseño	23,5 días	lun 17/04/23	jue 18/05/23
Creación del objeto enemigo	5,5 días	lun 17/04/23	dom 23/04/23
Creación del objeto jefe final	5,5 días	lun 24/04/23	dom 30/04/23
Creación de los niveles 2, 4 y el tutorial	5,5 días	lun 01/05/23	lun 08/05/23
Creación de los niveles 1 y 3	4,5 días	mar 09/05/23	lun 15/05/23
Creación del nivel 5	4,5 días	mar 16/05/23	dom 21/05/23
Documentación hasta el momento, examinada y corregida	25,5 días	lun 17/04/23	dom 21/05/23
PEC_4	**21 días?**	**lun 22/05/23**	**dom 18/06/23**
Implementació	6,5 días	lun 22/05/23	mar 30/05/23
Demostración	4,5 días	mié 31/05/2	mar 06/06/23
Líneas de futuro y conclusiones	3,5 días	mié 07/06/23	dom 11/06/23
Corrección de posibles incoherencias y errores en el código	7,5 días	lun 22/05/23	mié 31/05/23
Últimas mejorías en el apartado gráfico	7,5 días	mié 31/05/23	vie 09/06/23
Documentación terminada y revisada	20,5 días	lun 22/05/23	dom 18/06/23
Semana_de_reces	7 días	lun 19/06/23	mar 27/06/23
PEC_5	**6 días?**	**mié 28/06/2**	**mié 05/07/23**
Defensa ante el tribunal	5,5 días	mié 28/06/23	mié 05/07/23

Figura 2: Diagrama de Gantt de la planificación real del proyecto

PEC 1 Diseño (3/2/2023)->(12/3/2023)

- Hallar y definir la idea del juego
- Conceptualización
- Introducción
- Planificación total del trabajo

PEC 2 Versión parcial (13/3/2023)->(16/4/2023)

- Estado del arte
- Análisis de mercado
- Propuesta
- Diseño
- Elección y aprendizaje del *Game engine*.
- Creación de una pequeña *Testroom* para las pruebas.

- Creación del objeto jugador:
    - Sistema de movimiento (andar y correr).
    - Sistema de recogida de Ítems.
    - Sistema de habilidad (doble salto, agacharse, subir escaleras y rebotar en las paredes).
    - Animaciones.
    - Sistema de vida.
    - Sistema de ataque mediante disparo.
- Creación del sistema de generación procedural de niveles:
    - Objeto nodo.
    - Selección del nodo correspondiente a la entrada.
    - Algoritmo de búsqueda aleatorio DFS.
    - Selección del nodo correspondiente a la salida.
    - Creación de las habitaciones.
- Sistema de Interfaz visual del jugador:
    - Sistema de vida (apartado visual).
    - Sistema de recuento de puntos.
- Documentación:
    - Estado del arte
    - Análisis de mercado
    - Propuesta

PEC3 Versión jugable (17/4/2023)->(21/5/2023)

- Diseño
- Creación del objeto enemigo:
    - Máquina de estados.
    - Sistema de movimiento.
    - Animaciones.
    - Seguir al jugador.
    - Sistema de vida.
    - Mecánica de disparo o ataque.
- Creación del objeto “Jefe final”:
    - Máquina de estados.
    - Sistema de movimiento.
    - Animaciones.
    - Sistema de vida.
    - Mecánica de disparo o ataque.
- Crear niveles 2 y 4 (Niveles procedurales):
    - Diseño.

    - Creación del nivel.
- Crear niveles 1 y 3 (Niveles híbridos):
    - Diseño.
    - Creación del nivel.
- Crear nivel 5 (Jefe final):
    - Diseño.
    - Creación del nivel.
- Conclusiones y líneas de futuro.
- La documentación elaborada hasta el momento estará examinada y corregida.

PEC4 Memoria y productos finales (22/5/2023)->(18/6/2023)

- Instalación.
- Demostración.
- Conclusiones y líneas de futuro.
- Pruebas en el funcionamiento del juego.
- Corrección de posibles incoherencias o errores en el código.
- Últimas mejoras en el apartado gráfico.
- La documentación elaborada hasta el momento estará examinada y corregida.

1.6. Presupuesto

Para calcular el coste total de este proyecto, se han simulado los gastos que hubiera tenido si se hubiese realizado de forma profesional. En general, se han empleado licencias gratuitas, y software de código libre para las herramientas utilizadas. Y, para elementos más secundarios como el audio o las imágenes, se ha utilizado objetos de dominio público o que puedan ser utilizado bajo licencia de no comercialización. La idea general era invertir lo mínimo posible en el proyecto. Aun así, se decidió que el presupuesto podía ser lo suficientemente flexible para invertir una pequeña cantidad de dinero en ciertos elementos descargables.

Como aproximación, se ha calculado que la duración total de las horas del proyecto sería de alrededor de 450. Se obtiene esta cifra si de la cantidad de días disponibles para realizar el proyecto (110) solo contamos los laborales (75) y asumimos que cada día equivaldría a una jornada laboral de 6 horas.

1.6.1. Hardware

Para el cálculo del hardware utilizado, se decidió que todo el proyecto se desarrollaría en un mismo ordenador, seleccionando así un portátil de la marca MSI. Además, como en un principio se planteó realizar el lanzamiento simultáneo a la plataforma móvil, se incluyó un

smartphone de la marca IPhone en las primeras versiones. No obstante, durante el transcurso del proyecto no se ha llegado a utilizar, y por tanto no se ha incluido en la versión final.

Como aclaración, para obtener la factura total de la electricidad consumida, se ha consultado el sitio web “Selectra.es” [4]. Donde se ha llegado a la aproximación que la media de electricidad contratada a lo largo de todo el proyecto sería de unos 0.14516 €/kWh. Y, además, se ha anotado que la potencia que el transformador del portátil es de 600W.

Objeto	**Descripción**	**Cantidad**	**Coste**
MSI GS70 6QE	Ordenador portátil en el que se desarrollará el proyecto.	1	1.450,00 €
Consumo eléctrico	Aproximadamente 0.14516 €/kWh	Aproximadamente, 450 horas en total.	39,20 €
			Total: 1.489.2 €

Tabla 1: Presupuesto del Hardware empleado

1.6.2. Software

Los elementos de software que se han empleado a lo largo del proyecto han sido:

Objeto	**Descripción**	**Coste**
Unity	Se ha elegido la licencia de Unity Personal. La cual es gratuita siempre que los beneficios obtenidos con los productos creados no superen los 100.000€	0,00 €
GitHub	Un servicio de alojamiento online para el desarrollo de software y el control de versiones basado en Git.	0,00 €
GitHub Desktop	Una herramienta de código abierto que permite la interacción del usuario con el servicio GitHub por medio de una interfaz gráfica.	0,00 €
Inkscape	Un editor de gráficos vectoriales gratuitos y de código abierto.	0,00 €
		Total: 0,00 €

Tabla 2: Presupuesto del Software empleado

1.6.3. Tareas realizadas

Para realizar los cálculos de esta tabla se ha considerado que el precio por hora de un programador junior en España que deba realizar las tareas sería de 25 € la hora. Esta cifra ha sido calculada consultando la plataforma de contrato de trabajo "Zaask" [5], y aplicando una media aritmética a sus precios mínimo y máximo. Además, para el reparto de tiempo se ha utilizado la cantidad total aproximada de horas calculadas antes, y se ha repartido según la cantidad necesaria que se cree para cada apartado. En este caso, este apartado también se ha tratado como un borrador para obtener una aproximación del coste total que supondrían las tareas a realizar. También se espera que los porcentajes de tiempo utilizados varíen más adelante.

Producto	**Descripción**	**Cantidad**	**Coste**
Horas de análisis	25,00 €/h	28 horas	700,00 €
Horas de diseño	25,00 €/h	Aproximadamente, 98 horas dedicadas en tota	2.450,00 €
Horas de programación	25,00 €/h	Aproximadamente, 200 horas dedicadas en total	5.000,00 €
Horas de estudio de Mercado	25,00 €/h	33 horas	825,00 €
Horas de verificación con pruebas	25,00 €/h	55 horas	1.375,00 €
Horas de despliegue	25,00 €/h	18 horas	450,00 €
Horas de mantenimiento	25,00 €/h	18 horas	450,00 €
			Total: 11.250 €

Tabla 3: Presupuesto de las tareas realizadas

1.6.4. Otros recursos (*assets* y *sprites*)

La lista de *assets* y recursos gráficos utilizados para el desarrollo del juego son los siguientes:

Objeto	**Descripción**	**Coste**
Asset Pack "SunnyLand"	Descargado en la Unity Store. Contiene animaciones para el personaje principal, animaciones para los enemigos, y un *tileset* con el que crear diversos escenarios. Enlace.	0,00 €
26 Animated PixelArt Robots	Descargado en Itch.io. Contiene las animaciones de 26 robots distintos (terrestres y aéreos) utilizadas para los enemigos del juego. Enlace.	0,00 €

Sprite sheet de Concrete Man (el jefe final)	Descargado en la página web “The Spriters Resource”. Este sprite sheet contiene todas las animaciones de “Concrete Man” un antagonista de los juegos de Megaman (de Capcom). Estas imágenes, bajo permiso de uso no comercial, se han utilizado para todos los estados y ataques del enemigo. Enlace.	0,00 €
Lifebar pixelart sprites 16x16	Descargado en Itch.io. Contiene 4 sprites de distintos corazones utilizados para el objeto HearthLife y el vector de vidas de la interfaz gráfica. Enlace.	0,00 €
Smoke-Particle	Descargado en Itch.io. Contiene sprites de una animación de una nube de polvo. Fue usado en el efecto para caminar y correr del jugador. Enlace.	0,00 €
Fuente de texto “Press Start 2P”	Descargada en GoogleFonts. Ha sido la fuente de texto utilizada para los menús y botones de la interfaz gráfica. Enlace.	0,00 €
Rampaging Robot	Imagen de un robot utilizada para el diseño del fondo del menú principal del juego. Enlace.	0,00 €
Anime Landscape: Forest Background	Imagen de un bosque utilizada para el diseño del fondo del menú principal y el menú de niveles. Enlace.	0,00 €
		Total: 0,00 €

Tabla 4: Presupuesto de otros recursos utilizados

1.6.5. Música y efectos de sonido

A excepción de las canciones todos los *assets* y recursos de audio empleados en el juego han sido descargados de la página web “Freesound”. Esta funciona como un repositorio online de pistas de audio, licenciadas con Creative-Commons, que permite descargarlas bajo dominio público y licencias sin uso comercial. La lista de estos es la siguientes:

Objeto (Efectos de sonido)	**Descripción**	**Coste 0,00 €**
Chihuahua Puppy Whine	Efecto de sonido utilizado para la acción de recibir daño del jugador. Enlace.	0,00 €
Retro video game sfx - jump	Efecto de sonido utilizado para la acción de salto del jugador. Enlace.	0,00 €
Impact-3-full	Efecto de sonido utilizado para el efecto de aterrizaje del jefe final. Enlace.	0,00 €
8-bit Hurt	Efecto de sonido utilizado para el lanzamiento de un proyectil (tanto por parte del jugador como de los enemigos). Enlace.	0,00 €
Game-over	Sonido utilizado para la acción de despliegue del menú de derrota en el juego. Enlace.	0,00 €

8-Bit - Error	Efecto de sonido utilizado para la función del jugador de perder una vida. Enlace.	0,00 €
Magic_game_win _success_2	Efecto de sonido utilizado para la función del jugador de recuperar una vida. Enlace.	0,00 €
Completed	Efecto de sonido utilizado para la acción de despliegue del menú de victoria en el juego. Enlace.	0,00 €
Item_respawn	Efecto de sonido utilizado para la función de reaparición del jugador en el nivel. Enlace.	0,00 €
Item Sparkle	Efecto de sonido utilizado para la función de recoger un item en el nivel. Enlace.	0,00 €
Explosion	Efecto de sonido utilizado para la función de destruir a un enemigo cuando este es derrotado. Enlace.	0,00 €
Car Breaking Skid 01	Efecto de sonido utilizado para el estado de carga del jefe final. Enlace.	0,00 €
Imperius - Face Punch	Sonido utilizado para el puñetazo del estado “PunchRain” del jefe final. Enlace.	0,00 €
8 Bit Slam	Efecto de sonido utilizado para la acción de dañar a un enemigo o caja. Enlace.	0,00 €
8-bit Jump	Efecto de sonido utilizado para la acción de saltar del jefe final. Enlace.	0,00 €
Old Video Game 4	Efecto de sonido utilizado como la melodía de victoria tras la derrota del jefe final. Enlace.	0,00 €
Comedic Boing, A	Efecto de sonido utilizado para indicar la activación de un Checkpoint. Enlace.	0,00 €
Boing	Efecto de sonido utilizado para el efecto de impulso del trampolín sobre el jugador. Enlace.	0,00 €
8 bit Death sound	Efecto de sonido utilizado para la colisión de un proyectil contra algún objeto o superficie. Enlace.	0,00 €
Objeto (Canciones)	**Descripción**	**Coste 0,00 €**
Hurry_up_and_run	Canción incluida en el Pack “Sunnyland” utilizada como música de fondo para los niveles de superficie. Enlace.	0,00 €
Under the rainbow	Canción incluida en el Pack “Sunnyland” utilizada como música de fondo para los niveles de cuevas y la batalla contra el jefe final. Enlace.	0,00 €
Maniac	Canción incluida en el Pack “Sunnyland” utilizada como música de fondo para el menú inicial. Enlace.	0,00 €
Up_and_right	Canción incluida en el Pack “Sunnyland” utilizada como música de fondo para el menú de niveles. Enlace.	0,00 €
		Total: 0,00 €

Tabla 5: Presupuesto de la música y sonidos del juego

1.6.6. Coste total del trabajo

Con todas estas consideraciones mencionadas en los apartados anteriores, el cálculo definitivo del coste total del trabajo quedaría de la siguiente manera:

Objeto	Descripción	Coste
Hardware	Coste total de todo el hardware empleado en el proyecto.	**1.489,2 €**
Software	Coste total de todo el software necesario para el proyecto.	**0,00 €**
Tareas realizadas	Coste total de las tareas realizadas durante el proyecto.	**11.250 €**
Otros recursos	Coste total de todo el software necesario para el proyecto.	**0,00 €**
Música y efectos de sonido	Coste total de todo el software necesario para el proyecto.	**0,00 €**
Lanzamiento del juego en Steam	Coste inicial para publicar el juego en la plataforma.	**100 €**
Lanzamiento del juego en Google Play	Coste inicial para publicar el juego en la plataforma, (a condición de las ventas de Pc).	**25 €**
		Total: 12.864,20 €

Tabla 6: Cálculo del coste total

Así, según todos los cálculos realizados en las diferentes tablas, en esta primera instancia, el precio total del trabajo sería de 12.864,20 €.

1.7. Estructura del resto del documento

Para los siguientes 6 capítulos del trabajo, el plan para su realización ha sido el siguiente:

- **Estado del arte**: Estudio del género de juego escogido y los videojuegos más relevantes para este género.

- **Análisis de mercado**: Análisis detallado de la situación actual del mercado en la que se enmarca el producto del proyecto.

- **Propuesta**: A partir del estudio de mercado realizado, en este capítulo se presentará el plan de comercialización para el producto.

- **Diseño**: En este apartado se explicarán los detalles del desarrollo y los recursos, como la arquitectura general de la aplicación, el funcionamiento de los diferentes elementos y scripts, y los recursos gráficos utilizados para el juego.

- **Implementación**: Para este capítulo se detallarán los requisitos y componentes necesarios para para descargar, instalar y ejecutar el producto.

- **Demostración**: En esta sección se describirán las instrucciones de uso y la guía de usuario.

- **Conclusiones y líneas de futuro**: Finalmente, en este último apartado se recogerán las conclusiones personales sobre el proyecto, el proceso de trabajo y los resultados finales, y las predicciones y sugerencias que podrían realizarse para ampliar el proyecto en un futuro.

2. Estado del arte

2.1. Historia y evolución del género

Los juegos de plataformas es uno de los géneros más longevos que han existido en el mundo de los videojuegos. No solo se originó en un momento muy temprano de la industria, con las primeras generaciones de consolas, sino que han evolucionado progresivamente a la par que la tecnología. Haciendo que aparecieran nuevas mecánicas y formas de jugarlos al aprovechar las capacidades que los nuevos dispositivos ofrecían a lo largo del tiempo. Convirtiéndose así en uno de los géneros de los videojuegos más flexibles que existen.

Aunque, para comprender adecuadamente el género de plataformas, es necesario conocer un poco sobre su historia y evolución. Así, a continuación, se expondrá un resumen del recorrido de este estilo de videojuego desde sus inicios hasta su situación actual.

El género de plataformas se originó al principio de la década de los 80. Algunos juegos destinados a los Arcade, como “Space Panic” o “Crazy Climber”, se les considera precursores de este género. Ya que, incluían mecánicas que permitían al jugador trepar entre superficies y desplazarse horizontalmente sobre ellas. Pero, en general, se considera a “Donkey Kong” como el primer juego de plataformas al conseguir implementar la mecánica de salto entre ellas.

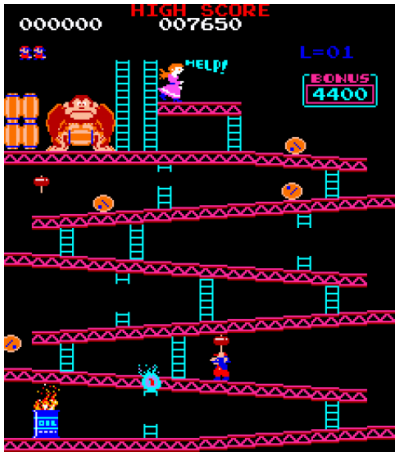


Figura 3: Donkey Kong (1981)

Hasta ese momento, los niveles en estos juegos se trataban de niveles estáticos que sucedían en la misma pantalla, y en los que el jugador era trasladado a la siguiente cuando las completaba. Solo un par de meses después del lanzamiento de “Donkey Kong” llegaría un nuevo juego que introduciría la mecánica de movimiento de cámara por el escenario, llamado “Jump Bug”. Este nuevo sistema ofrecía mucha más libertad para el jugador, al poder seguirlo a lo largo de un escenario que se revelaba a medida que avanzaba en dirección horizontal o vertical. Además, la llegada del videojuego recreativo “Moon Patrol” en 1982 introdujo el movimiento *parallax* del fondo de pantalla por primera vez.

Figura 4: Moon Patrol (1982)

Y, con el éxito del lanzamiento de “Super Mario Bros” en el 1985, para la consola doméstica “Nintendo Entertainment System”, muchas compañías empezaron a considerar relevantes los juegos de plataformas. Este juego introdujo una gran cantidad de nuevos elementos y mecánicas para los plataformas, convirtiéndose así en todo un referente para el género. Y, en consecuencia, haciendo que este estilo de juego se volviera muy popular durante todo el periodo de la generación de los 8-bits (la tercera generación de consolas).

Con la llegada de las consolas de 16-bits, el nuevo hardware más potente permitió que los desarrolladores pudieran introducir nuevas mejoras y elementos en los juegos. Un claro ejemplo de esta situación fue el lanzamiento de “Sonic” (de Sega) en 1991, para la “Sega Genesis”. En respuesta al éxito de Mario, la compañía Sega trató de realizar su propio lanzamiento exitoso por años. En esta nueva generación de hardware, los escenarios del juego podían expandirse tanto horizontal como verticalmente, se introdujeron curvas y loopings, e incorporaba un sistema de físicas que permitía al jugador recorrer los escenarios a toda velocidad.
En general, en esta generación surgieron algunos de los juegos de plataformas más icónicos de todos los tiempos como “Mega Man X” (Capcom), “Super Metroid” (Nintendo), “Castlevania: Symphony of the night” (Konami) o, secuelas de algunas sagas como “Super Mario World”.

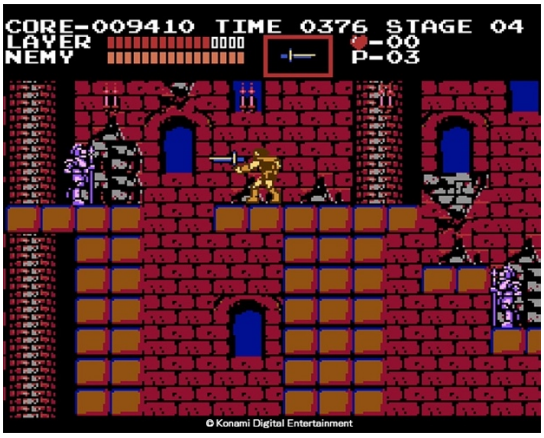


Figura 5 (Izquierda): Mega Man X (1993)

Figura 6 (Derecha): Castlevania (1986)

No obstante, con la llegada de los 32-bits, el 3D empezó a aparecer en escena. El nuevo hardware provocó que la atención de los jugadores se alejara de los géneros en 2D, y que ciertas compañías quisieran experimentar evolucionando antiguas entregas de sus juegos en nuevos formatos. Y, debido a la inexperiencia con el nuevo hardware para adaptar los juegos de plataformas al 3D, y aprovechando la llegada de este nuevo salto gráfico, nació un nuevo estilo de juego. El cual se desarrolló con la misma mecánica de movimiento en 2D pero, con Sprites de personajes y enemigos renderizados en 3D. Un nuevo género de juegos que posteriormente recibiría el nombre de 2.5D. Algunos ejemplos como “Donkey Kong Country” de Nintendo, o “Clockwork Knight” y “Pandemonium” de Sega, pertenecen a este estilo.

Durante los 80, ocasionalmente, aparecieron juegos que intentaban recrear el funcionamiento del género plataformas simulando una tercera dimensión (conocido como “Pseudo-3D”). Con técnicas como la perspectiva y el desplazamiento del escenario. No obstante, la llegada de los “juegos de plataformas que funcionaban en verdadero 3D” no ocurrió hasta la siguiente década, los 90, con la llegada de la quinta generación de consolas (según, “Guiness World Records” el videojuego “Jumping Flash!”, de 1995, se considera el primero de este nuevo estilo).

Figura 7: Jumping Flash! (1995)

Y, no fue hasta 1996 con el lanzamiento de “Super Mario 64” cuando se produjo un cambio de paradigma. Este juego también se volvió un referente para los juegos de plataformas en 3D, al ofrecer niveles abiertos en 3D, que fomentaban la exploración y permitían una mayor libertad al jugar, en comparación con otros productos presentes del momento. Posteriormente, juegos como “Banjo Kazooie” o “Donkey Kong 64” heredarían este formato.

Figura 8: Super Mario 64 (1996)

Por otro lado, ese mismo año Sony (quien había entrado en escena hace solo un par de años) lanzó “Crash Bandicoot” y adaptó su franquicia “Tomb Raider” a 3D. En el caso de la última, la mecánica de plataformas 3D se combinó con un sistema *shooter* e incluyó la resolución de puzles para determinados apartados. Esta terminó por convertirse en una de las franquicias para PlayStation 1 más vendidas.

Figura 9: Crash Bandicoot (1996)

Con la llegada del año 2000, Sony lanzó su nueva consola doméstica, “PlayStation 2”, que, a pesar de que no tuvo una buena acogida al principio, se hizo popular al presentar una nueva generación de juegos de plataformas. Estos incorporarían nuevas mecánicas y modos de juego que hicieron que empezasen a pertenecer a otros subgéneros. Por ejemplo, “Ratchet and Clank” (Insomniac) introdujo el combate con su primera entrega, “Jak and Daxter” (Naughty Dog) experimentó con el mundo abierto y “Sly Cooper” (Sucker Punch) introdujo las primeras mecánicas de sigilo.
En este periodo, Nintendo lanzó “Super Mario Sunshine”, para la GameCube en 2002, y Sega lanzó Sonic Adventure” como título de lanzamiento para la “Sega Dreamcast”.

Lamentablemente, esta se retiró, poco después, del negocio de consolas debido a las pérdidas económicas por la “Sega Saturn” y su creciente rivalidad con Sony.

Ya entrada la primera década de los 2000, con la llegada de la séptima generación de consolas, el género plataformas tenía una presencia menor en la industria. Sin embargo, ciertos lanzamientos como “Super Mario Galaxy” y “Ratchet and Clank Future: Tools of destruction” demostraban que este todavía era capaz de reinventarse si mismo.

Como otros ejemplos de este caso, Electronic Arts lanzó “Mirror’s Edge” que combinaba la acción del género plataformas con la perspectiva en primera persona; Naughty Dog dejó atrás el estilo animado de Crash Bandicoot para crear “Uncharted”, una nueva franquicia que combinaba las plataformas con la acción *shooter* en tercera persona y la narrativa cinematográfica. Y en 2008, de la mano de Sony, llegó “LittleBigPlanet” (por Media Molecule), un juego que combinaba la jugabilidad tradicional de los plataformas 2D con un sistema de edición de físicas que además permitía compartir el contenido creado por los usuarios.

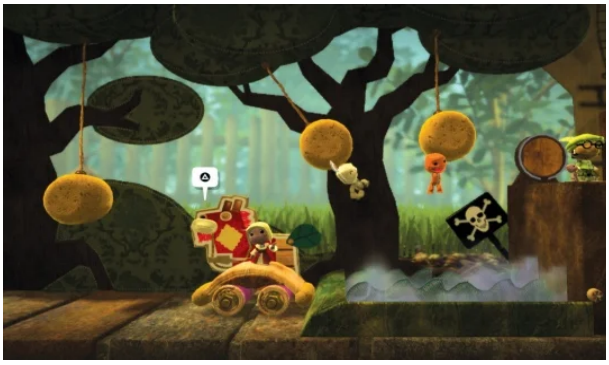


Figura 10 (Izquierda): Uncharted: Drake’s Fortune (2007)

Figura 11 (Derecha): Little Big Planet (2008)

Los juegos de plataformas se seguían considerando un género relevante, pero nunca recuperó su antigua popularidad. Uno de los motivos principales por lo que sucedió esto fue porque, fallaron en evolucionar e innovar con otros géneros contemporáneos de la industria. Se quedaron “estancados” y, progresivamente, los lanzamientos parecían cada vez más orientados al público infantil.

Afortunadamente, con la llegada de la década de los 2010, el género fue gradualmente experimentando un pequeño “renacimiento”. Ya que, numerosos desarrolladores *indie* empezaron a debutar con el lanzamiento de juegos que pretendían recuperar el espíritu de los primeros plataformas al volver a implementarlos en 2D.
Algunos títulos como Limbo, Braid o Super Meat Boy popularizaron el desarrollo Indie al convertirse en grandes éxitos. Lo que provocó que se recuperase cierto interés por parte del público.

Figura 12 (Izquierda): Braid (2008)

Figura 13 (Derecha): Limbo (2010)

Llegando ya al periodo actual, a pesar de que no es un género que siga siendo tan apoyado por las grandes compañías, al ser menos rentable, algunas de estas siguen intentando reinventarse con él. Por ejemplo, Nintendo con la implementación de mundo abierto en “Super Mario Odyssey” en 2017, o Sony con “Sackboy: A Big Adventure”, el spin-off de la saga “LittleBigPlanet” que permitió por primera vez el sistema de juego en 3D.
Al mismo tiempo, diferentes desarrolladores Indies han seguido lanzado títulos como “Inside”, “Cuphead” o “Hollow Knight”. Con los que se han centrado en innovar tanto en la narrativa como en las mecánicas de interacción del jugador, a la vez que tenían la intención de que el género retornase a sus orígenes.

Figura 14 (Izquierda): Cuphead (2017)

Figura 15 (Derecha): Hollow Knight (2017)

Desde aquí, parece difícil predecir cuál será el desarrollo de estos juegos en el futuro. Afortunadamente, un factor importante que parece haberse observado tras el repaso de toda su historia es que, este género posee una gran maleabilidad, innovación y adaptabilidad hacia otros formatos y modos de juego. Así, existe la esperanza de que, a pesar de que seguramente mantendrá su estatus como un género no tan común, algunos lanzamientos ocasionales harán que continue evolucionando y siga siendo un género disfrutable en el futuro.

Entre los diferentes tipos de juegos de plataformas que existen, algunos de los subgéneros más distinguibles son:

- **Plataformas de Puzles**
  Se caracterizan por emplear la estructura de los juegos de plataformas para establecer un juego cuyo objetivo es resolver un puzle determinado.

- **Plataformas Run and Gun**
  Estos juegos se centran menos en la habilidad del jugador para desplazarse entre las plataformas, y más en la jugabilidad con la mecánica de disparo que debe ser usada contra los enemigos (quienes aparecen de manera constante).

- **Plataformas isométrico**
  Este estilo de plataformas presenta un espacio en 3 dimensiones empleando la perspectiva isométrica. Estos juegos acostumbran a simular un espacio tridimensional en un entorno de 2 dimensiones, permitiendo desplazar al jugador mediante ciertas técnicas de proyección. O bien, utilizan un entorno en verdadero 3D utilizando la misma perspectiva.

- **Plataformas de aventura (o de estilo Metroidvania)**
  Este subgénero combina los elementos característicos de los plataformas con los juegos de acción y aventura. Concretamente, permiten la capacidad de explorar un área libremente, permitiendo el acceso a otras áreas ya sea adquiriendo nuevos objetos o avanzando en el mundo o la historia del juego. Además, durante el trayecto permite desbloquear nuevas habilidades o comprar nuevos artículos para el inventario.

- **Plataformas del género "*Endless runner*"**
  Este género de juego se trata de un juego de plataformas en los que el jugador se desplaza de manera constante a través de un nivel sin fin. La jugabilidad de estos se enfoca en los reflejos del jugador para esquivar los obstáculos que van apareciendo a medida que avanza. Este estilo de juegos se ha hecho popular entre las plataformas móviles, debido al pequeño conjunto de controles que requieren.

3. Análisis de mercado

3.1. Público objetivo y perfiles de usuario

Para definir y ejecutar correctamente la estrategia de marketing del producto desarrollado es vital que se conozca al público objetivo a quién se le pretenderá vender. La finalidad no es dirigirlo a la mayor audiencia posible, sino hacerlo hacia aquella a la que pueda interesarle más. Para ello, es más fácil si se analiza este supuesto grupo de usuarios en diferentes categorías. Como lo serían: su situación geográfica, su edad, sus preferencias de entretenimiento, el interés que tengan en comprarlo, y la cultura o culturas a las que vayan dirigidas. Pero, antes de volcarnos con cada uno de estos puntos, sería interesante analizar previamente el contexto de mercado.

En principio, solo se pretende lanzar “Underfox” para el mercado de España. Aunque, esto no significa que si el juego tiene buena acogida se plantee la posibilidad de exportarlo a otras zonas geográficas. España se encuentra actualmente en el puesto 23 en el ranking de países con un número de jugadores de videojuegos más activos (con un 83 % de la población entre 10 y 65 años que se reconocen como jugadores) [15]. Según el informe del 2022 de la firma de investigación y análisis Newzoo, España contribuyó con 2,3 mil millones de dólares en la industria del juego [13]. Poniéndolo en la posición número 13 entre los países que más facturaron ese año. Así que con este planteamiento podríamos asumir que, aunque solo se enfoque el producto al mercado español, el número inicial de usuarios ya es relativamente alto.

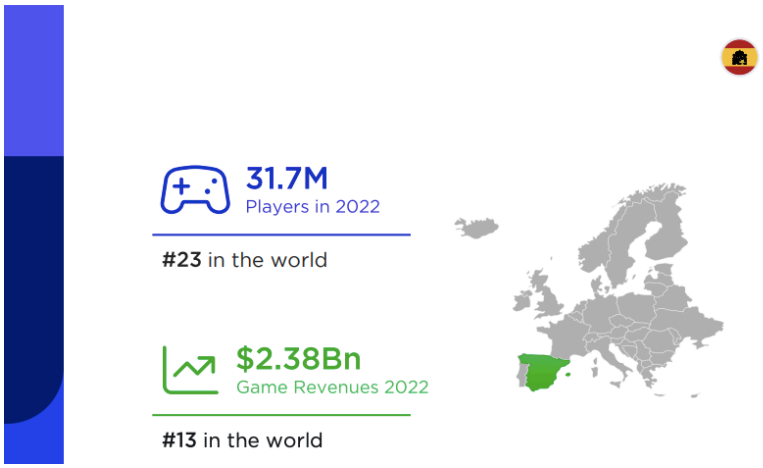


Figura 16: Imagen del informe de Newzoo sobre el mercado español de videojuegos (2022)

Desde el punto de vista de las edades de los jugadores, “Underfox” está pensado para que pueda llegar a un rango de edad amplio. Ya que, por una parte, se pretende conseguir un manejo de los controles accesible y sencillo. Que incluso las personas con poca experiencia jugando a videojuegos puedan encontrarlo cómodo e intuitivo. Y, por otra, el producto en sí tiene una temática más bien neutra. No se pretende que los personajes ni el contexto del

juego representen ningún hecho o situación del mundo real. Sencillamente, se utilizan como medio para relatar la historia y formar un pequeño contexto para el jugador, independientemente de su cultura o experiencia.

Además, desde la perspectiva de restricción de edad, el juego no posee ninguna característica específica, como violencia o secuencias gráficas que aterroricen a audiencias jóvenes, para que se le imponga una limitación de edad importante. De esta manera, según el sistema de clasificación europeo PEGI (Pan-European Game Information), se opina que el producto estaría en la restricción de edad más baja PEGI 3 (orientado a jugadores mayores de 3 años). Esta calificación hace que su contenido sea apropiado para grupos de personas de todas las edades.

Así atraeríamos a una audiencia relativamente amplia ya que, según las estadísticas recogidas en el mismo anuario [13], el grupo de edad que más jugaba a videojuegos eran preadolescentes entre los 11 y 14 años (con un 78% de difusión), seguidos por el grupo conformado por niños de entre 6 y 10 años (con una difusión del 76%).

A continuación, si hablásemos del nivel de interés de estos futuros jugadores, se espera que el juego atraiga a personas con varias “filosofías” de juego diferente. Por una parte, con su relativa facilidad en los controles y temática neutra se pretendería atraer a una audiencia más bien novata (*Newbies*) con menos experiencia en el mundo de los videojuegos, y jugadores ocasionales (*Casual Gamers*), que no tienen los videojuegos como afición seria, pero, disfrutan jugando de vez en cuando.

A pesar de que, existen dudas sobre el porcentaje total de jugadores que habría en el primer caso, la previsión inicial es que la gran mayoría de usuarios serían ocasionales. Esto se debe a estos usuarios no perciben los juegos como una forma principal de entretenimiento, sino que su objetivo es entretenerse un rato con ellos de manera casual. En general su intención es no invertir demasiada cantidad de tiempo o dinero en ello, sino en entretenerse de vez en cuando sin ningún compromiso.
Así, la distribución del juego en niveles y el sistema de guardado físico de la posición del personaje en el mapa (mediante el uso *Checkpoints*) haría que el juego fuese más fácil para jugar de forma casual. Permitiendo al jugador dejar de jugar o retomarlo cuando él quiera.
Además, a pesar de que la opción de no desbloquear los niveles secuencialmente (como en juegos como “Megaman”) hubiese favorecido a este estilo de juego, no se ha querido agregar a la versión final. En vez de eso se ha preferido mantener el sistema de desbloquearlos consecutivamente al superarlos. Para que, aunque solo juegue de vez en cuando, se mantenga el interés en el jugador y quiera seguir progresando en el juego.

Por esta misma razón, se podría afirmar con seguridad que atraer a una audiencia de jugadores cuya afición por los juegos es más seria (*Core gamers*), no sería posible. Ya que, este tipo de usuarios no considera a los videojuegos como cualquier otra forma de entretenimiento, para ellos son una devoción. Además de disfrutar jugándolos, su objetivo es tener una sensación de recompensa superando cada obstáculo y reto que el juego plantea. Sobre todo, estos usuarios tienen preferencia por los productos *triple A* lanzados por las

compañías (que aseguran la calidad de un estudio y prometen un número elevado de horas de juego). Solo ocasionalmente la salida de algún producto Indie menos profesional llama su atención.

Esto no encaja con el carácter del producto. Ya que, desde un principio, su jugabilidad en los niveles no estaba pensada para que supusiera un reto, o se incluyesen elementos que fomentaran su “re-jugabilidad”. Sino para que el jugador pudiera divertirse al hacerlo. Se podría pensar que habría una pequeña esperanza de que aspectos como la generación procedural en ciertos niveles pudiera llegar a llamar la atención de algunos usuarios serios. Sin embargo, no se debería contar con ello. Además de que, debido al reducido tiempo de producción, el objetivo del producto no es que tenga un acabado profesional, sino funcional.

Elegir el género de un videojuego en el momento de su conceptualización es uno de los puntos más importantes que definirán su éxito potencial. Para conocer un poco más acerca del estado de los plataformas en 2D en el mercado actual de los videojuegos, se ha tomado como referencia una gráfica del análisis de las ventas y la popularidad de los juegos indies de Steam. Que realizó la página “*How to market a game*” [14]. Más específicamente, en esta gráfica se aprecian la cantidad de juegos vendidos de cada género (con una gráfica de barras en azul), y la mediana de ingresos alcanzada por cada uno (con una gráfica de líneas en rojo).

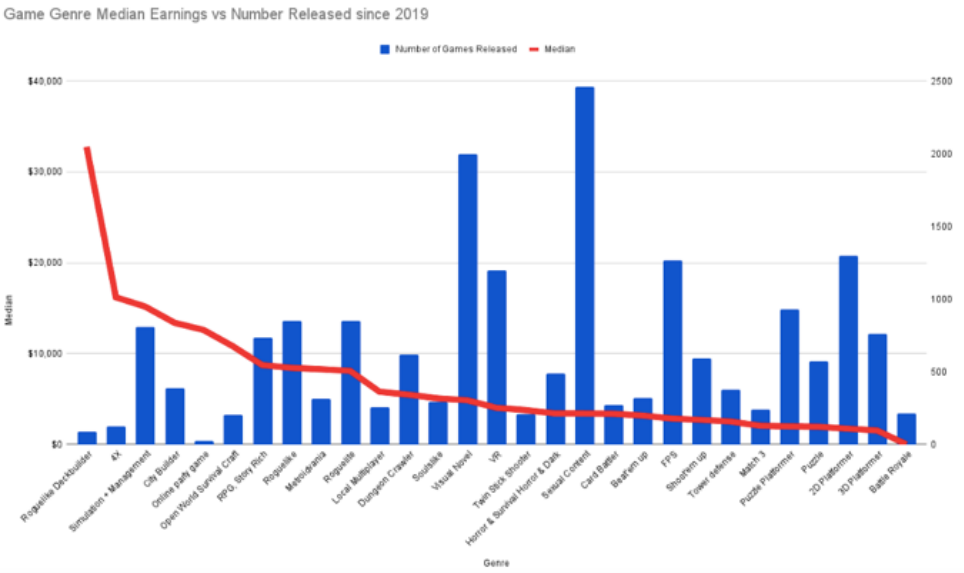


Figura 17: Gráfica de los géneros de juegos más vendidos en *Steam* 2019 - 2022

En la tabla, el género de las plataformas en 2D se encuentra en la antepenúltima posición. Este posee una barra azul que sobrepasa los 1.500 juegos lanzados, y una mediana roja que parece que apenas llega a una cuarta parte de los 10.000$ indicados. Esto indica que a pesar de ser un género indie con una gran cantidad de lanzamientos en Steam, el rendimiento de la venta de estos juegos no es demasiado alto. Así, se podría definir a los plataformas 2D como un género popular pero, es también uno que está saturado.

Para estudiar mejor el rango de los ingresos por género, se ha tomado como referencia otro gráfico publicado en la misma página [14]. Se trata de un gráfico de velas japonés, en estos los ingresos se representan de la siguiente manera: la punta del “hilo” representa la cantidad ganada por el 1% de los juegos; la parte superior de la barra representa el cuartil superior; la parte inferior de la misma barra representa la mediana, y el fin del “hilo” representa el cuartil inferior.

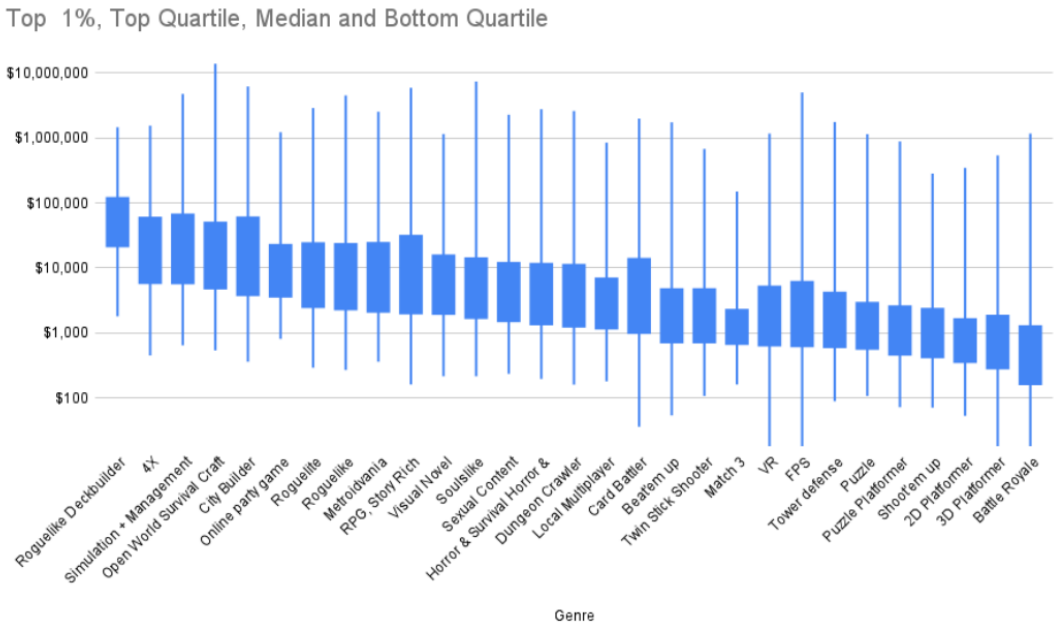


Figura 18: Gráfica de velas los géneros de juegos más vendidos en *Steam* 2019 - 2022

Como se puede ver, el cuartil inferior del género de plataformas 2D llega a algo menos de los 100$, la mediana llega hacia la mitad entre los 100 y los 1.000$; el cuartil superior sobrepasa un poco los 1.000$, y la cima de la vela (representando el 1%) alcanza hasta la mitad entre los 100.000 y los 1.000.000$. Esto confirma que, en la mayoría de los casos, los lanzamientos de plataformas no alcanzan mucho más de los 1.000$. Y, solo en alguna excepción concreta algún lanzamiento indie de plataformas ha conseguido un rendimiento elevado.

De esta manera parece que el género escogido para el proyecto a desarrollar es uno en el que es difícil destacar. Debido a la poca demanda que existe y la gran competencia que hay por la oferta, parece seguro asumir que solo los lanzamientos de ciertos juegos tienen el rendimiento esperado. En esta situación solo quedaría procurar que nuestro producto final incluya ciertas características específicas (como la generación procedural en ciertos niveles) que consiguieran diferenciarlo lo suficiente de la competencia para que pueda destacar y conseguir un buen rendimiento de ventas.

Finalmente, para terminar de definir a este público objetivo se debería concretar las plataformas de juegos a las que estaría dirigido el producto. Para ello, primero deberíamos revisar las cifras del contexto de mercado en España. En 2021 la AEVI registró que, entre los dispositivos favoritos para jugar, un 27% de los usuarios (empatando con las consolas)

prefería su *smartphone*; seguido de cerca por los PCs (por un 22% de los usuarios) y las *tablets* (con un 12% de usuarios totales) [1].
En este caso, no se estaría enfocando la salida del juego para consolas, ya que ese público acostumbra a ser en su gran mayoría jugadores serios y, como ya se razonó antes, no tienen costumbre de prestar interés a videojuegos más pequeños o menos profesionales. Así, parece que la mejor opción serían las plataformas de PC y *Smartphone*.

En el caso del primero, la plataforma de PC cuenta con una audiencia más variada. Entre su audiencia, además de jugadores serios, también se puede encontrar usuarios más casuales, e incluso jugadores novatos. Es decir, el rango de preferencias para jugar de los clientes es mayor en esta plataforma. Por lo que, en un principio, el juego tendría buena acogida enfocando su lanzamiento en ella.

Ahora regresando a los *smartphones*, son una plataforma de juegos perfecta para la categoría casual. Ya que, en general, esta nueva industria ha facilitado el acceso de los desarrolladores independientes a un público general (que hasta ese momento no eran demasiado conocidos en el sector), a la vez que, jugadores más bien casuales, han podido recrearse de forma más fácil y directa. Haciendo que sea una plataforma perfecta para lanzamientos de juegos que podrían considerarse menos serios o con un público no tan devoto al mundo de los videojuegos, como se hablaba antes.

Desde el que se podría considerar su nacimiento en 2008, con el lanzamiento del primer **iPhone** y la **AppStore** de Apple, esta industria ha ido adquiriendo cada vez más relevancia con el paso de los años. En 2015 se registró que el porcentaje de ingresos recaudados por las ventas de juegos de móvil (un 34%) empató con las ventas de juegos para Pc (según Newzoo [20]). Y en 2020, la llegada de la pandemia global de la Covid-19 provocó que la industria de juegos móviles, junto con el resto de la industria de videojuegos, experimentase un crecimiento. Llegando a registrar, hasta 2,5 billones de jugadores de móvil a nivel mundial (un 12% más que en el año pasado), y una recaudación total de 77,2 billones de dólares a lo largo de ese mismo año [23]. En 2022, se registró alrededor de 110 billones de dólares recaudados en este sector (que representaría cerca de un 60% del total recaudado a nivel mundial por la industria de videojuegos ese año).
Por todos estos motivos, también parece una buena idea enfocar el lanzamiento del juego a esta plataforma.

3.2. Competencia

A continuación, se realizará una comparativa de la competencia a la que se enfrentaría el juego si se decidiera sacarlo en este momento. Para ello se ha optado por comparar sus características con algunos de los plataformas 2D más populares del servicio de distribución de videojuegos Steam. Por el hecho de que se ha planeado lanzar el juego en ella cuando esté terminado, como se explicará más adelante en el apartado de propuesta. El análisis de estos productos se ha realizado mediante el uso de 2 tablas comparativas: para definir sus características y para determinar sus pros y contras.

Título	Fecha de lanzamiento	Géneros	Plataformas	Precio	Breve descripción
Pizza Tower	26 de enero, 2023	Plataformas, Acción, Comedia, Retro	Microsoft Windows	19,50€	- Plataformas 2D de habilidad que hace uso del humor absurdo e incorpora gráficos con estilo de dibujos animados.
Super Meat Boy	20 de octubre, 2010	Plataformas, Jugabilidad difícil, Acción, Comedia	Xbox 360, Microsoft Windows, OS X, Linux, PlayStation 4, PlayStation Vita, Wii U, Nintendo Switch	1,39€ - 15,99€	Plataformas 2D de habilidad y agilidad en el que el personaje controlado “Meat Boy” debe atravesar más de 200 niveles para rescatar a su novia.
Brawlhala	17 de octubre, 2017	Lucha, Free to Play, Multijugador, Casual	macOS, PlayStation4, Microsoft Windows, Nintendo Switch, Xbox One, Android, iOS	0,00€	- Videojuego de lucha 2D con acceso *Free to play* y permite el *cross platform*. Características perfectas para el juego casual.
Have a nice Death	8 de marzo, 2022	Roguelike de acción, Comedia, Hack and Slash, Metroidvania	Microsoft Windows, Nintendo Switch	24,99€	- Plataformas 2D de estilo Metroidvania con habilidad y combates cuerpo a cuerpo personalizables con armas.
Rain World	22 de marzo, 2023	Plataformas, Puzles, Jugabilidad difícil, Aventura, Retro	PlayStation4, Microsoft Windows, Nintendo Switch	15,99€ - 24,50€	- Plataformas de exploración y aventura que incorpora entornos ambientales y envolventes, y enemigos con una IA compleja.
Celeste	25 de enero, 2018	Jugabilidad difícil, Retro, Aventura, Cinemático, Exploración	Linux, macOS, Nintendo Switch, Play Station 4, Microsoft Windows, Xbox One, Stadia	19,99€	- Plataformas 2D de habilidad y precisión en el que el objetivo es llevar a la protagonista “Madeline” hasta la cima de una montaña.
Noita	15 de octubre, 2020	Roguelike, Retro, Metroidvania, Sandbox, Acción y aventura	Microsoft Windows	18,99€	- Sandbox de aventura y exploración que incorpora la generación procedural de niveles y recursos.

Tabla 7: Exposición de la principal competencia del juego

Título	Pros	Contras

Pizza Tower	- Comedia. - Re-jugabilidad por desafíos. - Acción.	- Duración relativamente corta. - Disponible solo para plataforma Windows.
Super Meat Boy	- Precio bajo según la plataforma de compra. - Disponible en muchas plataformas. - Duración larga.	- Dificultad elevada en la jugabilidad.
Brawlhala	- Acceso gratuito (Free to play). - Acertado para el jugador casual. - Comunidad amplia.	- Servidores del juego con poco mantenimiento. - Solo permite jugar en red. - Videojuego de combate.
Have a nice Death	- Estilo visual atractivo. - Combates personalizables con armas. - Acción.	- Solo disponible en 2 plataformas. - Posee el precio más elevado de todos.
Rain World	- Estilo visual atractivo. - Los mundos del juego son ambientales y envolventes. - Los enemigos poseen una IA compleja. - Libertad de juego.	- Dificultad elevada en la jugabilidad. - El castigo cuando el jugador muere es severo.
Celeste	- Estilo visual conmemora plataformas clásicos. - Disponible en muchas plataformas. - Jugabilidad ágil y cómoda. - Re-jugabilidad por logros.	- Dificultad elevada en la jugabilidad. - Duración relativamente corta.
Noita	- Estilo visual conmemora plataformas clásicos. - La generación procedural hace única cada partida. - Re-jugabilidad por *Sandbox*.	- Generación aleatoria hace que la dificultad varíe mucho según el nivel. - Solo disponible en la plataforma Windows.

Tabla 8: Análisis de la principal competencia del juego

3.3. Análisis DAFO

Debilidades y Fortalezas

Desde el punto de vista interno, existen un par de características que pueden suponer un inconveniente para que el producto se desarrolle como debería.

Para empezar, como ya se había comentado anteriormente. El autor de este proyecto no posee demasiada experiencia con el entorno de programación de Unity. El transcurso de este trabajo ha demostrado que este factor es vital. Ya que, aprender sobre el *game engine* al mismo tiempo que estos conocimientos son aplicados, requiere más tiempo y esfuerzo que si ya los tuviera consolidados desde el principio. Esto no significa que la calidad del juego vaya a ser inferior a la que se esperaba, pero es probable que, hacia el final, con la nueva experiencia adquirida, se perciba que ciertos aspectos de la calidad del código podrían mejorarse.

Por otra parte, tras repasar detalladamente cada uno de los juegos, se ha podido observar que el aspecto gráfico pixelado suele ser adoptado por numerosos juegos en este género. No

hay que confundirse, esto no es exactamente malo. Al igual que la intención inicial de “Underfox”, estos juegos también pretenden evocar el estilo de los plataformas clásicos a la vez que los conmemoran. No obstante, un requisito importante respecto a su lanzamiento y estado actual del género es tratar de destacar en algún aspecto en comparación con los otros productos. Y, es casi seguro que el aspecto pixelado no conseguirá ser lo suficientemente llamativo para lograrlo. Para solucionar esto, en una situación en la que aún se pudiese corregir, se podría o bien, optimizar el estilo pixelado mejorando la animación y el estilo, o trabajar con un estilo de gráficos diferente con la intención de que llame la atención del cliente (por ejemplo, Pizza Tower tiene un estilo poco pulido, pero por esta misma razón llama la atención del usuario).

Figura 19: Portada del juego “Pizza Tower”

Siguiendo con la competencia, se contempla el inconveniente de que el producto final es más corto e incorpora menos elementos en comparación (tanto en los escenarios como en la jugabilidad). Parece evidente que la intención que tiene un comprador con un juego es “rentabilizarlo”. Mejor dicho, da por hecho que el producto adquirido durará una cierta cantidad de horas que justifiquen su precio. No obstante, “Underfox” solo incluye una cantidad de 6 niveles. Haciendo que, como mucho, se pueda superar en pocas horas. Se espera que los niveles de generación procedural aporten cierta re-jugabilidad a la experiencia del cliente. Sin embargo, todavía está por determinar si esto será suficiente.

Como fortalezas, el producto tiene la intención de llegar a una audiencia amplia con el enfoque de la “filosofía del juego casual”. Para empezar, los controles del personaje se han implementado de manera que jugar con él resulte cómodo. También, se ha decidido establecer un nivel asequible en la dificultad general del juego. En principio, estas características están dirigidas al público de juegos casuales, que no posee demasiado interés en aprender cómo jugar o enterarse de trucos para facilitar su jugabilidad. Por esta misma razón, se ha dividido el juego en niveles no demasiado largos y se ha creado el objeto “*Checkpoint*” que permite guardar la posición del jugador en el escenario en determinados lugares de cada nivel. De esta forma, tendrán la libertad para guardar sus avances periódicamente y así poder dejarlo o retomarlo cuando quieran. Y, todavía existe la esperanza que la generación procedural en ciertos niveles pueda atraer a jugadores serios como un pequeño porcentaje de la audiencia total.

Por otra parte, se espera que las cualidades de niveles intuitivos con una dificultad no muy alta puedan ser aplicadas a la edad de la audiencia. Ya que, debido a su temática neutra e historia simple, la intención es que pueda interesar tanto a niños como adultos.

Por último, se puede observar una posible ventaja en el precio. Y es que, debido a que los principales lanzamientos del género son Indies, los precios no alcanzan una cantidad demasiado alta. No obstante, se prevé que el producto que se está desarrollando tenga un precio de lanzamiento incluso menor que ellos. Algunos de estos juegos como Super Meat Boy (2010) ya tienen unos años desde su lanzamiento, pero, por lo que se ha observado en las tablas, generalmente los productos Indies de plataformas suele encontrarse entre los 15€ y los 20€. Esto se puede ver con Rain World que, siendo el juego más caro de los expuestos, alcanza solo un precio de 24,99€. Este hecho puede suponer un pequeño punto a favor ya que, al ser más reducido, se tenía planeado lanzar el producto con un precio no superior a 4,99€. Un precio reducido es una característica importante a favor. Ya que, si la jugabilidad y otros aspectos están lo suficientemente pulidos, podría ser lo suficientemente llamativo para el cliente.

Amenazas y Oportunidades

Por otra parte, desde el punto de vista externo, la mayor amenaza que se ha podido observar en la sección del público objetivo es que el género de plataformas 2D está saturado. Es decir, el mercado presenta una oferta muy grande de juegos de plataformas y una demanda pequeña de ellos. Por lo que se ha analizado, es un estilo de juego que ya casi no apoyan las plataformas de consolas o las grandes compañías que desarrollan videojuegos de mayor envergadura. Y, en la actualidad, se consigue mantener “vivo” sobre todo por medio de lanzamientos Indies que consiguen tener éxito de forma ocasional.

Al mismo tiempo, se ha llegado a la conclusión de que la clave de estos triunfos casuales es que cada uno de estos productos aporta una distinción suficiente en el género en comparación con el resto de los productos disponibles. Por ejemplo, incorporando mecánicas novedosas en su jugabilidad o que, en general, se “salgan del molde convencional” al pulir algún aspecto que pudiera resultar en un beneficio para la experiencia del jugador.

Ante esta situación parece lógico asumir que, para conseguir que el producto a desarrollar consiga alcanzar las ventas necesarias, es imprescindible que incorpore una mejora o distinción en algún aspecto para sobresalir en comparación con los demás.

Nuestra oportunidad para destacar entre la multitud de juegos se basaría en la mecánica de generación procedural de zonas en determinados niveles. Esta es una mecánica que no se ha visto todavía en demasiados juegos (a excepción de algunos como el ya mencionado “Noita”) y, empleando una publicidad adecuada para este concepto, podría atraer la atención de usuarios interesados y compradores.

4. Propuesta

4.1. Definición de las especificaciones del producto

El producto propuesto es un videojuego del género Plataformas en 2D. Este está diseñado para un solo jugador y no requiere ningún tipo de conexión a Internet. Ha sido desarrollado con el *Game engine* Unity, y estará disponible para la plataforma Windows y, en el caso de que tuviera éxito también móvil. Ya que, se ha analizado que en estas dos conseguiría llegar a una audiencia más amplia. Pudiéndose obtener a través de la plataforma de distribución de Steam en un principio.

Como ya se comentó previamente, el videojuego solo estaría disponible en español. No obstante, no se ha negado la posibilidad de traducirlo a otros idiomas en el caso de que alcance cierta tasa de éxito.

La historia y el contexto del juego son sencillos: en una isla remota, un día aparece una gran corporación que pretende acabar con todos los recursos naturales existentes y edificar una fábrica. Para ello, despliega una horda de robots en diferentes zonas para ejecutar su operación. En consecuencia, uno de sus habitantes, un zorro, decide tomarse la situación por su mano y plantar cara a la malvada corporación acabando con todo el ejército él mismo.

4.2. Modelo de negocio

Como modelo de negocio se propone la compra del producto completo en un único pago, en las plataformas de distribución online en las que se haya decidido publicarlo. Se plantea este método porque se prevé que el juego resultante será pequeño. Y, por tanto, otras opciones como la venta por capítulos o la compra de expansiones que requieren pagos adicionales, no parecen un modelo provechoso.

En esta situación, tras analizar donde puede tener más éxito este género de videojuego en el apartado anterior, se decidió orientar su lanzamiento a las plataformas de Windows, y posteriormente Smartphone. Porque se previó que tendría una buena acogida en ellas. Teniendo este hecho en mente, parece que los sitios web más adecuados donde publicar el producto serían los servicios de distribución digital Steam y Google Play debido a su amplio reconocimiento en ambas plataformas.

En el caso de Steam, según la información facilitada por diferentes artículos [[26],[27],[28]] se debe pagar una cantidad inicial de 100€ para poder lanzar tu juego en la plataforma. Además, la plataforma se hace con una comisión del 30% de todos los ingresos obtenidos por cada juego publicado. Por otro lado, en Google Play sucede algo parecido. Primero, se debe pagar una cantidad inicial de 25€ para poder publicar tu juego en esta plataforma (según otros artículos sobre esta plataforma [29]). Y, este servicio también se hace con una pequeña

comisión del 15% por cada descarga de una aplicación de pago y compras dentro de la aplicación.

En principio, se pretende lanzar este juego con un precio de 4,20€ en la plataforma Steam (y, con suerte, posteriormente en Google Play). Tras analizar los precios de los juegos principales en el apartado anterior, se concluyó que el precio de un juego indie de plataformas en 2D se encuentra entre los 15€ y los 20€. Sin embargo, todos estos casos anteriores se trataban de juegos con una cantidad aceptable de horas de juego y una calidad mínima esperable. Por tanto, la cantidad propuesta de 4,20€ para su lanzamiento parece aceptable considerando el alcance y características del producto.

Por tanto, siguiendo con las especificaciones previas de cada plataforma de Steam, con cada venta se obtendría una cantidad de 2,80€. Teniendo en cuenta que el presupuesto total del proyecto es de 12.015,34€ se calcula que para que el producto empiece a ser rentable, en cada lanzamiento aislado, se necesitaría vender o bien un total de 4.291 copias vendidas en Steam. Esta, tal vez, podría ser una cantidad relativamente alta para la audiencia que esperamos. Afortunadamente se ha planeado que, si las ventas llegan a alrededor de un 40% en Steam, se preparará el producto para lanzarlo también a Google Play exportándolo a plataformas móviles. En esta situación, cada descarga en Google Play aportaría una cantidad de 3,57€ por descarga. De esta forma la magnitud total se distribuiría entre las ventas de ambas páginas. Por ejemplo, en una situación hipotética en la que ambas plataformas generarían la mitad de los beneficios: en total se deberían vender 2.146 copias en Steam y 1.683 copias en Google Play. Facilitando de esta manera la cantidad de ventas a alcanzar.

4.3. Estrategia de marketing

Para conseguir que el producto en desarrollo tenga éxito, tanto por la aprobación del público general como económicamente, es vital establecer una campaña de marketing adecuada para darlo a conocer. A continuación, se han propuesto varias estrategias que se podrían seguir para promocionarlo. No obstante, se debe mencionar que solo se han escogido algunas que no implicaran una carga demasiado severa para el esfuerzo económico u organizativa del proyecto (ya que, en desarrollos profesionales, lo más normal es que una parte importante del presupuesto del proyecto se destine a la publicidad). Así, algunas de las estrategias más adecuadas para el contexto de este proyecto serían:

- Para empezar, al estar desarrollando un juego que se podría categorizar como “Indie”, un buen plan sería tratar de promocionarlo en redes sociales mostrando actualizaciones en su proceso de creación bajo la etiqueta de “#Gamedev”. En este momento existe una comunidad extensa, en algunas plataformas sociales, de personas que se dedican al diseño de videojuegos indie, ya sea de manera profesional o como una simple afición. De manera periódica publican actualizaciones de su trabajo en redes como Twitter o Reddit para dar a conocer su producto a un público interesado en este ámbito y otros desarrolladores.

- Crear un tráiler del juego. En general este debería verse llamativo y enfocarse en un estilo que concuerde con el grupo de usuarios que se ha concebido como público objetivo, y estar lo suficientemente bien editado para poder usarlo en entornos como redes sociales o las propias páginas de venta. El tráiler tendría que promocionar el juego visualmente, mostrando grabaciones del propio “Gameplay” en las que se aprecien las características que lo diferencian de la competencia (ya que podemos suponer que, como en análisis de mercado, también habrá saturación de tráilers de juegos de plataformas).

- La creación de una página web en la que publicar información sobre el juego. Concretamente, tendría que ser un espacio online personalizado en el que se publicaría toda la información detallada sobre el juego. De esta manera cuando alguna persona interesada decida buscar información sobre el juego pueda encontrar una página oficial con toda la información que deseaba obtener. Como desventaja esta estrategia no se centra en dar a conocer el producto, sino en dar apoyo las tácticas anteriores cuya función sí es promocionarlo.

- Contactar con creadores de contenido para promocionar el juego sería otra opción. En este caso, sería apropiado no tratar de dirigirse a creadores con un número de seguidores elevado, sino enfocarse en aquellos que podrían considerarse de nuestro nivel. Por ejemplo, creadores de plataformas como “Twitch” o “Youtube” con un número de espectadores alrededor de 1.000. Ya que, lo más seguro es que los creadores famosos reciban una cantidad elevada de peticiones parecidas y, en consecuencia, se terminen descartando o ignorando. No les interesará el juego de un desarrollador con menos experiencia, a menos que el autor ya posea cierto reconocimiento.

- Por último, siguiendo otras estrategias económicas que siguen algunos desarrolladores, también se ha barajado rebajar el precio del juego a medida que trascurra el tiempo. Es decir, un buen método para llamar la atención del público y seguir manteniendo presencia en las páginas en las que se ha publicado. Sobre todo, en épocas en las que aparecen rebajas generales en las páginas en las que se han publicado. Sin embargo, este proceso debe hacerse con cuidado porque, si se rebaja el precio demasiado rápido en un periodo relativamente corto tras el lanzamiento, puede dar una impresión errónea al público (por ejemplo, que el juego no posee una calidad suficiente). En un principio, se había pensado en rebajar el precio del producto en un 25% un año y medio después de su lanzamiento. Los planes posteriores a esa fecha para esta estrategia todavía están por determinarse en función de su rendimiento.

5. Diseño

5.1. Entorno de desarrollo

5.1.1. Selección del entorno de desarrollo

Como el objetivo final es desarrollar un producto cuyas características sean lo más parecidas al concepto que se había planeado, y tanto el tiempo como los otros recursos son limitados, la mejor opción es trabajar con un Game Engine. Pero, antes de investigar a fondo los diferentes motores de juegos que se han propuesto como candidatos, es importante que se definan bien las características concretas del producto que se pretende desarrollar.
Para empezar, será un juego de plataformas en 2D sencillo, que no requerirá acceso online ni actualizaciones. Debe incorporar un sistema de físicas suficiente para poder implementar las mecánicas de salto y agilidad del jugador. La programación debe ser suficiente para la implementación en 2D del método de generación procedural de cavernas. Además de los sistemas de movimiento y ataque, basados en estados, de los enemigos. Además, la carga gráfica será baja ya que, se han elegido *sprites* con poca resolución, no se ha empleado un sistema de iluminación y las animaciones tienen una cantidad de FPS más bien baja. Y por último, a ser posible, debe emplear un lenguaje de programación ya conocido por el desarrollador. O como mínimo que sea fácil de utilizar.

Según las opiniones de diferentes fuentes sobre este tema, los motores de juegos más apropiados para desarrollar el producto serían:

- **Godot**: Godot es un motor de juegos multiplataforma de código abierto, lanzado en enero del año 2014. Este sistema incorpora un editor intuitivo con una interfaz de usuario accesible, haciendo que la creación de un juego sea más fácil y sencilla para sus desarrolladores. En general, Godot tiene como objetivos ofrecer un entorno de desarrollo de juegos totalmente integrado. Incorporando así una gran cantidad de herramientas, para que el usuario no recurra a otras eternas a las ya incluidas. Las más útiles son su motor de físicas, “Bullet”; sus herramientas de animación, y sus opciones de multijugador en red.
Para la programación con Godot permite utilizar C++, C# o su propio lenguaje incorporado basado en secuencias de comandos, llamado “GDScript”. Este es un lenguaje dinámico y de alto nivel, sintácticamente similar a Python.
En referencia al precio, al ser de código abierto, Godot es totalmente gratuito. Este hecho también permite exportar el juego creado a entornos de sistemas de escritorio (Windows, macOS, Linux), móviles (Android, iOS), web (HTML5), e incluso consolas (con la ayuda de una editora externa).
Por último, Godot posee un manual detallado en su página web oficial sobre los diferentes aspectos del motor tanto para el manejo del sistema como para el desarrollo de juegos. Además de una comunidad de foros muy activa que provee ayuda a los usuarios que tienen preguntas o necesitan ayuda para solucionar determinados problemas durante el desarrollo.

- **Unity**: Unity es una plataforma de desarrollo de videojuegos que fue lanzada en 2005 por Unity Technologies. Fue concebido en un principio como motor de juegos exclusivo de OS X

(de Apple), y desde entonces el motor se ha ido ampliando gradualmente. Admitiendo plataformas de consolas, móviles y de realidad virtual o aumentada.
Su sistema puede emplearse para desarrollar juegos en 2D y 3D, así como simulaciones interactivas u otras experiencias. Posee una interfaz de usuario intuitiva que facilita el proceso de aprender y crear juegos; un motor de físicas incorporado; gráficos de alta calidad; un sistema de animación, y función multiplataforma.
Para su programación, Unity emplea el lenguaje C# para sus Scripts. En cuanto al precio, al contrario que Godot, Unity es un producto comercial de pago con un acuerdo de licencia. Es decir, dependiendo del plan escogido, Unity ofrece unas funcionalidades u otras. Según la página web oficial de Unity, existen 7 licencias diferentes: **Unity Personal** (como un plan gratuito con la mayoría de las funcionalidades incluidas. Y una tasa máxima de ingresos de 100.000$ anuales); **Unity Plus** (con 369$ anuales o 40$ mensuales. Y unos ingresos máximos de 200.000$ anuales); **Unity Pro** (diseñado para personas o equipos profesionales); **Unity Industry** (cuyo uso exige que los beneficios superen el 1.000.000$ en el último periodo anual); **Unity Enterprise** (dirigido ya para empresas serias), y finalmente estarían las licencias para el **plan de estudiante** o el **plan de educador**.
Por último, Unity una guía ubicada en su página web oficial. Y, al igual que Godot, tiene una comunidad muy activa que se dedica a resolver dudas y preguntas en los foros online (posiblemente la más grande de los 3). Cosa que los usuarios encuentran muy útil para aprender, y resolver problemas con su software.

- **GameMaker**: GameMaker es un motor de juegos comercial multiplataforma que está enfocado especialmente en el desarrollo de juegos en 2D. Este fue lanzado el 15 de Noviembre del 1.999 como herramienta gráfica. Lanzamientos progresivos redirigieron su enfoque a la creación de juegos en 2D.
Su editor visual también es simple e intuitivo. Este funciona mediante la funcionalidad “Drag and Drop” (o “Arrastrar y soltar”). En cuanto a las otras funcionalidades que se han analizado previamente como las físicas, GameMaker utiliza la biblioteca de “Box2D”. La cual es capaz de aportar un control total sobre este apartado, pero no es tan intuitivo como en los anteriores ejemplos.
En la programación, GameMaker utiliza su propio lenguaje llamado GML (GameMaker Language), que incluye aspectos de lenguajes como JavaScript, C++ y C#. Cuya sintaxis es fácil de utilizar según las opiniones de sus usuarios.
En referencia al precio, GameMaker también ofrece diferentes planes de suscripción como Unity. Sin embargo, la capacidad de exportar juegos hacia otras plataformas depende del nivel de suscripción que el usuario haya pagado. Según la página web oficial de GameMaker, existen: **el Plan Gratuito** (permite el acceso principal a Gamemaker y la posibilidad de exportar el juego a la plataforma web GXGames); **el Plan Creador** (un plan de 49,99$ anuales o 4,99$ mensuales, con los beneficios del plan gratuito además de permitir la exportación a Windows, MacOS y Linux); **el plan Indie** (un plan de 99,99$ anuales o 9,99$ mensuales, con los beneficios del plan creador además de permitir la exportación a HTML5, iOS y Android), y finalmente, **el Plan de Empresas** (un plan de 799,99$ anuales o 79,99$ mensuales, que incluye todas las características y la exportación a consolas).
Finalmente, igual que Unity, GameMaker también posee una guía en su web con directrices sobre cómo utilizar la interfaz y el sistema del motor de juegos. Y, una comunidad activa en

los foros que también se dedica a resolver dudas o problemas. Sin embargo, la guía es menos detallada en comparación.

5.1.2. Entorno de desarrollo elegido

Tras haber comparado en detalle los 3 motores de juego, se ha optado por emplear Unity para el desarrollo del proyecto. Las razones principales por las que se ha llegado a esta decisión han sido la programación con C# y el hecho de que Unity contiene todos los módulos de exportación incluidos en su licencia gratuita.
En el caso del primero, C# es un lenguaje de programación orientado a objetos con el que el autor del proyecto ya está familiarizado, e incluso ya ha trabajado con otros similares como Javascript. No parecía lo más lógico aprender un nuevo lenguaje para crear el juego (por muy intuitivo que GML o GDScript se presentasen) cuando la experiencia podía jugar a favor.
En cuanto a los módulos de exportación, al aprender más sobre los planes de pago de GameMaker, este quedó descartado. Ya que, para exportar el juego a las plataformas a las que se habían previsto, se necesitaría pagar como mínimo el Plan Indie. Este hecho tampoco convencía demasiado cuando Unity podía realizar esa misma exportación con su licencia gratuita. Además, Unity permitía utilizar VisualStudio Code como IDE. Que es una herramienta muy versátil y facilita el proceso de Debug en caso de la aparición de errores.
Y por último, como ya se mencionó, Unity es el entorno con una comunidad más grande. Lo que significa que tendrá más abundancia en tutoriales y más actividad en los foros de dudas entre otras cosas.

5.1.3. Requisitos del entorno elegido

Los requisitos técnicos para emplear el editor de Unity 2021.1, según su página web oficial, serían:

- **Sistema Operativo: Windows**, versiones 7, SP1 o posterior, y 10 (ambas únicamente en las versiones de 64 bits); **macOS,** High Sierra 10.13 y posteriores, y **Linux**, Ubuntu 16.04, Ubuntu 18.04 y CentOS 7.

- **CPU:** arquitectura x64 con apoyo para el conjunto de instrucciones SSE2.

- **API Gráfica:** requiere GPU con capacidad DX10, DX11 y DX12.

- **Requisitos adicionales: Windows**, drivers oficialmente apoyados por el proveedor de Hardware; **macOS,** drivers oficialmente apoyados por Apple, y para **Linux**, el entorno de desarrollo Gnome corriendo sobre el sistema X11, el controlador oficial patentado de Nvidia, o el controlador de gráficos AMD Mesa.

5.2. Herramientas utilizadas

Además del entorno de desarrollo descrito en los apartados anteriores, para el desarrollo del proyecto también se ha hecho uso de las herramientas siguientes:

- **Visual Studio Code 2019 para Windows:** es el IDE (Integrated Development Environment) que se ha utilizado para la programación de los Scripts del proyecto. Contiene todo lo necesario para programar con C# y realizar el proceso de depuración y Debug en el caso de errores.

- **GitHub:** es un servicio de alojamiento online para el desarrollo de software y el control de versiones basado en Git. Al igual que otras herramientas basadas en Git, permite trabajar con diferentes ramas del proyecto, actualizar contenido mediante el comando “push” o “commit”, o descargar contenido mediante el comando “pull” entre otras. Se ha utilizado para almacenar periódicamente el contenido del código del proyecto en él. De esta manera se ha podido controlar el estado del trabajo, controlar las modificaciones realizadas con cada actualización, y, en el caso de que se hubiera necesitado, revertir el cambio en determinadas versiones.

- **GitHub Desktop:** es una herramienta de código abierto que permite la interacción del usuario con el servicio GitHub por medio de una interfaz gráfica en vez de mediante líneas de comandos o un navegador web. Se utilizó para sincronizar los cambios del proyecto almacenado en el PC con el espacio del servidor de GitHub, y que así estén presentes en la última versión online.

- **Microsoft Word 2021:** es un procesador de texto desarrollado por Microsoft que se ha utilizado para crear la memoria de este proyecto.

- **Piskel App:** es una aplicación de pixel art online que ofrece la función de crear o editar sprites de manera gratuita. Esta herramienta ha sido utilizada para la creación de imágenes del juego que no se podían encontrar en sitios como Itch.io o GameDevMarket (a pesar de haber realizado una búsqueda exhaustiva en ellos).

- **Draw.io:** es una herramienta online gratuita de diseño gráfico que se ha utilizado para crear los diagramas de flujo y ciertos esquemas del proyecto.

- **Inkscape:** es un editor de gráficos vectoriales de código abierto que se ha utilizado para crear los fondos de pantalla del menú inicial y el menú de niveles.

5.3. *Assets* y recursos

5.3.1. Recursos gráficos

- **Sunny Land**: De este pack se ha utilizado su Tileset para construir el terreno de juego, y determinados elementos y obstáculos (concretamente las púas, las plataformas delgadas, las

escaleras, las plataformas móviles, las cajas destructibles y los objetos de puntuación como las cerezas y los diamantes). Además de todas las animaciones empleadas para el repertorio de habilidades del jugador. Enlace.

- **26 Animated PixelArt Robots**: este pack contiene las animaciones de hasta 26 enemigos distintos tematizados como robots. Los cuales alternan entre terrestres y aéreos (lo que es perfecto para utilizarlos en enemigos del juego con estas mismas habilidades). Enlace.

- **Sprites de Concrete Man** (el jefe final): los sprites utilizados para el jefe final fueron descargados en un documento PNG de la página web "The Sprites Resource" (dedicada principalmente al archivo de sprites e imágenes de diferentes juegos). Este contenía todas las animaciones del repertorio de ataques y proyectiles que utilizaba este enemigo. Enlace.

- **Sprites de corazón**: proporciona 4 *sprites* diferentes de corazón que se han empleado para el objeto "HearthLife" (que permite recuperar una vida), y el vector con las vidas restantes del jugador situado al lado de la barra de salud en la interfaz gráfica. Enlace.

- **Fuente de texto "Press Start 2P"**: Descargada en GoogleFonts. Ha sido la fuente de texto utilizada para los menús y botones de la interfaz gráfica. Enlace.

- **Rampaging Robot**: Imagen de un robot utilizada para el diseño del fondo del menú principal del juego. Enlace.

- **Anime Landscape: Forest Background**: Imagen de un bosque utilizada para el diseño del fondo del menú principal y el menú de niveles. Enlace.

- **Elementos hechos a mano**: para este último caso, aparecieron ciertos elementos del juego durante su desarrollo que requerían imágenes o animaciones que no se pudieron encontrar. Por ejemplo, para la torreta ya se había planeado la creación de un enemigo inmóvil que supusiese un obstáculo y con un ataque a distancia. Pero no encontró ningún **Asset** con las características gráficas requeridas. Así, al contar con un poco de experiencia en pixelart, el autor decidió crearlos por su cuenta siempre que estos no fuesen demasiado complejos. En el caso de este enemigo solo se creó la animación para el estado "Idle" (moviendo su cabeza adelante y atrás), y el estado "Attack" cuya animación hace ver que escupe el proyectil. El resto de los elementos creados para el juego fueron: las piñas que el jugador puede usar como proyectiles; las pizarras en las que se explican los controles en el nivel tutorial; los proyectiles del enemigo Torreta, los *sprites* para la animación de los Checkpoints, el *Sprite* de la meta en cada nivel, las animación del trampolín, y los ítems de la tirita y el botiquín. Las imágenes utilizadas para estos objetos se encuentran dentro de la carpeta "Props" de la sección "*Assets*".

Figura 20 (Superior-Izquierda): Bandera de Checkpoint

Figura 21 (Superior-Derecha): Trampolín

Figura 22 (Inferior-Izquierda): Torreta

Figura 23 (Inferior -Derecha): Proyectil de piña

5.3.2. Recursos de audio

Sonidos

- **Chihuahua Puppy Whine**: Efecto de sonido utilizado para la acción de recibir daño del jugador. Enlace.

- **Retro video game sfx -jump**: Efecto de sonido utilizado para la acción de salto del jugador. Enlace.

- **Impact-3-full**: Efecto de sonido utilizado para el efecto de aterrizaje del jefe final. Enlace.

- **8-bit Hurt**: Efecto de sonido utilizado para el lanzamiento de un proyectil (tanto por parte del jugador como de los enemigos). Enlace.

- **Game-over**: Sonido utilizado para la acción de despliegue del menú de derrota en el juego. Enlace.

- **8-Bit – Error**: Efecto de sonido utilizado para la función del jugador de perder una vida. Enlace.

- **Magic_game_win _success_2**: Efecto de sonido utilizado para la función del jugador de recuperar una vida. Enlace.

- **Completed**: Efecto de sonido utilizado para la acción de despliegue del menú de victoria en el juego. Enlace.

- **Item_respawn**: Efecto de sonido utilizado para la función de reaparición del jugador en el nivel. Enlace.

- **Item Sparkle**: Efecto de sonido utilizado para la función de recoger un ítem en el nivel. Enlace.

- **Explosion**: Efecto de sonido utilizado para la función de destruir a un enemigo cuando este es derrotado. Enlace.

- **Car Breaking Skid 01**: Efecto de sonido utilizado para el estado de carga del jefe final. Enlace.

- **Imperius - Face Punch**: Sonido utilizado para el puñetazo del estado “PunchRain” del jefe final. Enlace.

- **8 Bit Slam**: Efecto de sonido utilizado para la acción de dañar a un enemigo o caja. Enlace.

- **8-bit Jump**: Efecto de sonido utilizado para la acción de saltar del jefe final. Enlace.

- **Old Video Game 4**: Efecto de sonido utilizado como la melodía de victoria tras la derrota del jefe final. Enlace.

- **Comedic Boing, A**: Efecto de sonido utilizado para indicar la activación de un Checkpoint. Enlace.

- **Boing**: Efecto de sonido utilizado para el efecto de impulso del trampolín sobre el jugador. Enlace.

- **8 bit Death sound**: Efecto de sonido utilizado para la colisión de un proyectil contra algún objeto o superficie. Enlace.

Canciones

- **Hurry_up_and_run**: Canción incluida en el Pack “Sunnyland” utilizada como música de fondo para los niveles de superficie. Enlace.

- **Under the rainbow**: Canción incluida en el Pack “Sunnyland” utilizada como música de fondo para los niveles de cuevas y la batalla contra el jefe final. Enlace.

- **Maniac**: Canción incluida en el Pack “Sunnyland” utilizada como música de fondo para el menú inicial. Enlace.

- **Up-and_right**: Canción incluida en el Pack “Sunnyland” utilizada como música de fondo para el menú de niveles. Enlace.

5.4. Arquitectura del juego

Para mostrar todas las interacciones que el jugador puede realizar en el juego, y qué respuestas se puede esperar en función de ellas, se ha creado el siguiente diagrama de estados para mostrarlo.

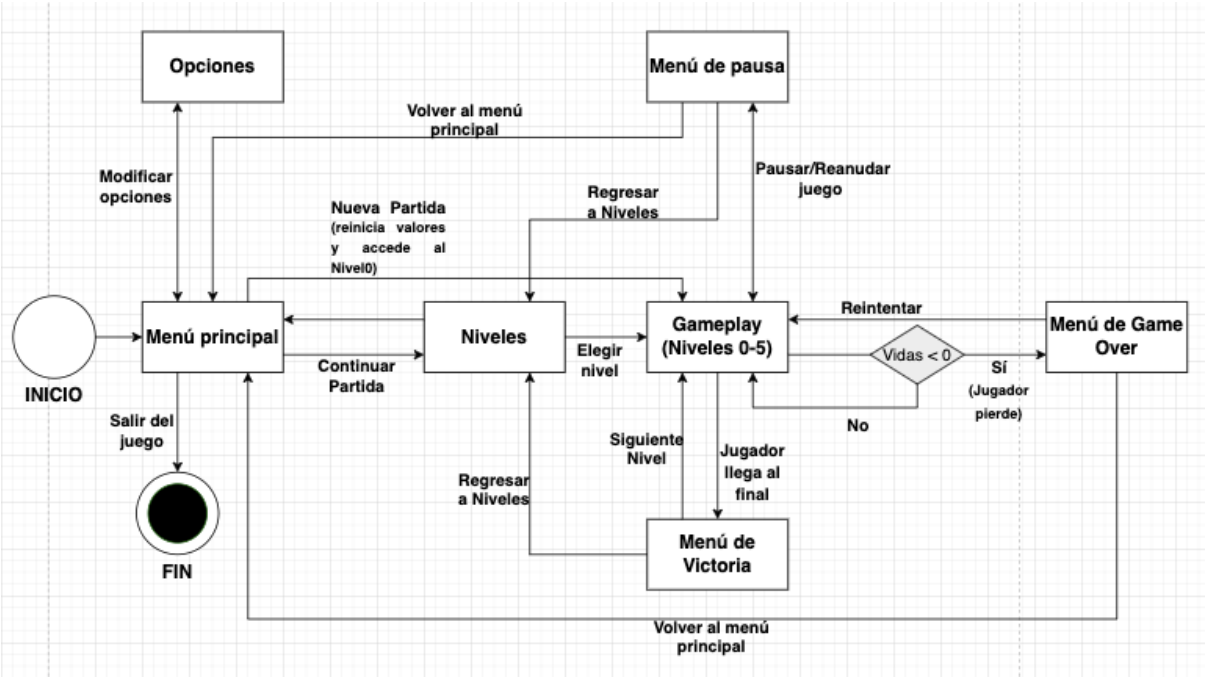


Figura 24: Diagrama de estados de “Underfox”

El funcionamiento es el siguiente: en total existen 8 escenas distintas. Seis de ellas están dedicadas a los niveles jugables, constituidos por el tutorial y los niveles oficiales del 1 al 5. Las otras 2 se han utilizado para el menú principal y el menú de niveles respectivamente.
El menú principal es la primera escena que se carga para el jugador, los botones del cual funcionan mediante el Script “MainMenu”. Desde ahí, se ofrecen las opciones de iniciar una Nueva Partida mediante el botón con el mismo nombre (su funcionalidad consiste en llevar al jugador al nivel 0 y, en caso de partida previa, borra los valores almacenados de la partida actual); acceder al menú de niveles, con los niveles desbloqueados hasta el momento por el jugador, con el botón “Continuar Partida”; acceder a las opciones disponibles de audio y gráficos con el botón “Opciones”, o cerrar el programa con el botón “Salir del juego”.

Si el jugador elige acceder a niveles, el programa cargará la escena que da acceso a la lista de niveles. La única forma que tendrá el jugador para desbloquearlos será ir superándolos consecutivamente. Para ello, este menú tiene asociado el script “LevelMenu”, que se ocupa de permitir el acceso a los niveles que ya haya superado el jugador previamente. Su funcionamiento se basa en un vector con los botones disponibles y una variable que almacena la cantidad de niveles que el jugador ha superado entre sesiones de juego. El Script solo

activará la cantidad lineal de botones que coincida con ese número, el resto los dejará desactivados. Por último, el usuario también tendrá la opción de regresar al menú principal.

Si el jugador selecciona un nivel, se cargará la escena correspondiente a este. El objetivo de cada nivel es el mismo, llegar hasta la línea de meta. Y, como reto para superar, existirán un conjunto de enemigos y obstáculos que le dificultará poder conseguirlo.
Durante el transcurso de cada nivel el jugador podrá detener el juego accediendo al menú de pausa con el botón correspondiente que tenga la versión. En este proceso el tiempo de ejecución del entorno del juego se detiene y se le muestran 3 nuevas opciones al jugador: continuar el juego reanudando el tiempo de ejecución, regresar al menú de niveles, y volver al menú principal (desde el que podría salir del juego).

Si es capaz de llegar hasta el objetivo final, se activará el menú de Victoria de esa misma escena. Este se encarga de dar una retroalimentación positiva al jugador indicándole que lo ha conseguido aplicando un leve filtro verde a la pantalla y mostrando la frase “Nivel superado”. A la vez, el tiempo de ejecución del entorno de juego de la escena se detiene. Finalmente, este menú ofrece 2 posibilidades al jugador, continuar accediendo al siguiente nivel o regresando al menú de niveles.

Si no es capaz de superar el nivel y pierde todas sus vidas, se activará el menú de Game Over de esa escena. Al contrario que el anterior, este se encarga de dar una retroalimentación negativa al jugador indicando que la partida ha terminado y no lo ha conseguido. En este proceso, el personaje del jugador queda destruido y la cámara se queda estática. Al mismo tiempo, el menú aparece como un leve filtro rojo en pantalla y muestra la frase “Game Over”. Desde aquí, se ofrecen otras 2 nuevas posibilidades al jugador, reintentar el nivel o regresar al menú principal.

5.5. Elementos del juego

5.5.1. GameManager

El GameManager es el objeto más importante en los niveles del juego. Este consiste en un script, que está presente en todas las escenas jugables, creado con el modelo “singleton”. Este sistema hace que solo exista una única instancia de este objeto en todos los niveles, y esta siempre es única para cada nivel.
Su tarea principal consiste en gestionar diferentes elementos del juego durante la partida, y también manejar diferentes valores y funciones durante esta. Concretamente, las tareas más importantes que posé son las de gestión de los menús de derrota y victoria; el sistema de muerte y reaparición del jugador, o la actualización de la posición de reaparición del jugador mediante la activación de *checkpoints* entre otros.
Por ejemplo, para realizar el primer proceso, el GameManager posee una variable que hace referencia al objeto jugador del nivel, y un vector que almacena la posición en la que el jugador debería reaparecer. Cada vez que el script “Healthbar” se queda a 0, accede a la función “Kill()” del GameManager. La cual elimina el objeto actual del jugador e instancia uno nuevo en la posición de reaparición que tenga almacenada.

5.5.2. AudioManager

Para el sistema de audio del juego se han creado 2 objetos diferentes: el AudioManager y el objeto ClipSound.

El objeto ClipSound consiste en un script que contiene 2 elementos: su pista de sonido correspondiente, y el nombre con el que se ha nombrado a esta. Este último será importante más adelante porque, se empleará como un identificador para buscar la pista de audio concreta cuando el juego ordene reproducir un sonido específico en una situación determinada.

Al igual que el GameManager, posee un script creado con el modelo “singleton”. Por lo que solo existe una instancia de este para cada escena, y esta siempre es única para cada nivel. Su función es la de reproducir las canciones que suenan de fondo, y los efectos de sonido que aparezcan en el juego a medida que el jugador interactúa con los diferentes objetos. Básicamente, este script posee una lista del objeto ClipSound correspondiente a las canciones, y otra que almacena todos los efectos de sonidos que el jugador podría escuchar en una escena.

Para simplificar el proceso de localización de las pistas de audio al invocarlas, se ha utilizado el sistema de Array de Unity. Se decidió de esta manera porque, este sistema permite almacenar fácilmente los efectos de sonido asignándoles un identificador (el nombre de la pista en este caso). Así, cuando se solicite utilizar ese mismo efecto de sonido de la lista, mediante el uso de la función correspondiente, el sistema se ocupará de identificarlo y obtenerlo mediante este. Es un método más sencillo que recordar en qué posición de un vector estaba cada efecto concreto.

A su vez, el AudioManager en sí contiene 2 objetos hijos que constituyen la fuente de audio en la que se reproducen las canciones, y la fuente en la que se reproducen los efectos de sonido. De esta manera, cada uno se ocupa de una tarea distinta, y además el jugador puede configurarlos individualmente en el menú de opciones. Durante la ejecución del juego, la fuente de las canciones reproduce de forma automática y en bucle la pista de audio correspondiente, nombrada como “LevelSong”. Mientras que, la fuente de los efectos de sonido posee una función que es convocada por otros objetos cuando estos cambian de estado o el jugador realiza una acción sobre ellos. Concretamente, el objeto llama a la función pasándole como parámetro el nombre del clip de audio, y si se encuentra en la lista es reproducido por la fuente.

5.5.3. Interfaz gráfica

La interfaz gráfica de los niveles se divide en los siguientes componentes:

- **Barra de salud**: situado en la esquina superior izquierda, es una barra verde gestionada por el script “HealthBar” que indica la cantidad de salud restante que posee el jugador. Cada

vez que el jugador es dañado por un obstáculo o enemigo este script asociado actualiza la barra restándole una cierta cantidad de daño establecida por el objeto que le ha agredido. Si la barra llega a cero, esta se reinicia e invoca una función del script “LifeCount” para descontar una vida del script del vector de vidas. Por último, el valor de salud que el jugador posee cuando termina un nivel, se mantiene cuando este acceda al próximo. Pero, si el jugador abandona la partida (si accede el nivel al menú de niveles o al principal) o el nivel es reiniciado se reiniciará su valor a la cantidad original (100 unidades de salud).

- **Vector de vidas**: situado en la zona superior junto a la barra de vida, es un vector gestionado por el script “LifeCount” que muestra las vidas restantes que posee el jugador. En un inicio el jugador dispone de 4 vidas, estas se van restando mediante una función que es convocada por el script “Healthbar” como ya se ha explicado. Si la cantidad de vidas visibles llega a 0, el objeto jugador no vuelve a reaparecer y se ordena al GameManager que muestre en pantalla el menú de derrota. Finalmente, la cantidad de vidas que el jugador posee cuando termina un nivel, también se mantiene cuando este accede al próximo. Pero, si el jugador abandona la partida al menú de niveles o al menú principal, o el nivel es reiniciado se reiniciará su valor a la cantidad original de 4 vidas.

- **Contador de munición**: es un contador administrado por el script “ItemManager” que se encuentra en la zona central superior de la pantalla. En cada nivel este inicia con una cantidad de 0, y se incrementa cada vez que el jugador recoge uno de los ítems de proyectiles del escenario (como se verá más dependiendo del ítem pueden sumar una cantidad de 1, 5 o hasta 10 proyectiles). La cantidad máxima de munición de la que puede disponer un jugador es de 15 proyectiles. Y, como en los objetos relacionados con la salud y vidas del jugador, su valor se mantiene entre los diferentes niveles del juego pero es reiniciado a 0 cuando el jugador abandona el nivel al menú de niveles, el menú principal o si el nivel es reiniciado.

- **Contador de diamantes recogidos**: también es un contador administrado por el script “ItemManager”. Se encuentra en la zona superior de la pantalla, a la derecha de la munición. En cada nivel este inicia con una cantidad de 0, y se incrementa cada vez que el jugador recoge uno de los ítems de diamantes del escenario. Como en los casos anteriores su valor es persistente entre niveles, y si se abandona o reintenta el nivel se reiniciará a 0.

- **Contador de cerezas recogidas**: es un contador que se incrementa cada vez que el jugador recoge uno de los ítems de cereza del escenario. Se encuentra en la zona superior derecha de la pantalla. También se inicia a 0 en cada nivel, y es controlado por el “ItemManager”. Como en los elementos anteriores su valor también es persistente entre niveles, y si se abandona o reintenta el nivel se reiniciará a 0.

- **Menú de victoria**: aparece desactivado durante la partida hasta que el GameManager lo activa cuando el jugador consigue llegar al final del nivel. En este proceso el tiempo de ejecución del juego se detiene. En el menú aparecen 2 botones con opciones diferentes, acceder al siguiente nivel o regresar al menú de niveles.

- **Menú de derrota**: también aparece desactivado durante la partida hasta que es activado por el GameManager si el jugador pierde todas sus vidas en un nivel. En este aparecen 2 botones con opciones diferentes, reintentar el nivel o regresar al menú principal.

- **Menú de pausa**: al contrario que los anteriores, este menú puede ser activado o desactivado por el jugador con la tecla “Esc” por su respectivo script. Esta acción también detiene el tiempo de ejecución mientras el menú es visible, dejando así al juego y la música del nivel en pausa. En este menú se ofrecen diferentes opciones con los botones para reanudar el juego, acceder al menú de niveles o volver al menú principal.

5.5.4. Cámara

La función de la cámara es seguir al jugador con un movimiento suave, mediante el script “CameraFollow”. Este script tiene incluido unos valores espaciales máximos y mínimos para hacer que no pueda salirse de un cierto rango de coordenadas. Ya que, de esta forma se impide que el jugador pueda ver zonas que no debería, como el límite de un escenario o el vacío bajo el suelo de un nivel. Además de eso, en sus alrededores la cámara posee 2 colisionadores a los lados y uno encima que actúan como barrera para el jugador. Se implementó así para que el jugador no pudiera volver atrás cada vez que alcanzase un *checkpoint*. Cuando lo hace este objeto transmite nuevos valores espaciales para la cámara. Estos impiden que la cámara pueda retroceder demasiado atrás, y junto a esta el jugador. Además, también posee un colisionador en el suelo que elimina una vida en el caso de que el jugador caiga al vacío en alguna zona (ya que al no poderlo seguir indefinidamente hacia abajo este siempre acaba colisionando). Como problema imprevisto, cuando el jugador moría y la cámara retrocedía hasta el último *checkpoint*, las barreras empujaban a este. Como solución se implementó una función que las desactiva y solo las reactiva cuando está lo suficientemente cerca del jugador.

Finalmente, para aportar una sensación más realista, se ha creado un fondo de nivel que sigue a la cámara con el efecto *parallax*. Este se compone 3 capas diferentes situadas unas delante de otras para simular profundidad, que se mueven con la cámara a velocidades diferentes. Cada capa la forman una serie de imágenes puestas unas al lado de otras. Cuando una imagen se aleja de la cámara a una distancia determinada (porque se deja atrás o el jugador retrocede), esta se transporta al lado contrario para mantener la ilusión.

5.5.5. Jugador

El personaje jugador es el objeto más relevante del programa. Ya que, debido a las características de los juegos de plataformas en 2D, será el elemento principal mediante el que el jugador, interactúe en el juego. De manera que, era importante dotarlo de las capacidades suficientes para que el usuario sintiese cierta sensación de agilidad al jugar con él (los cuales se detallarán en el apartado de Implementación).
Para conseguirlo, sus funciones y valores son gestionados por distintos elementos y scripts. Principalmente, su movimiento y habilidades son gestionados por el script “PlayerMovement”. Este le permite realizar funciones como el doble salto, correr, agacharse, adherirse y rebotar

por las paredes, lanzar piñas y sufrir daño. En el caso de los otros elementos, cada vez que el jugador sufre daño, “PlayerMovement” accede a la función del script “HealthBar” pasándole como valor la cantidad de daño sufrido. Para la habilidad de lanzar proyectiles de piña, “PlayerMovement” primero consulta a “ItemManager” si la cantidad de munición de la que dispone el jugador es superior a 0. Si se da este caso, el script instancia proyectiles en una posición determinada para el jugador. Finalmente, como ya se mencionaba antes, el GameManager interviene plenamente en su sistema de muerte y reaparición.

5.5.6. Enemigos

Todos los scripts que manejan a los enemigos que se describirán a continuación heredan del script padre “Enemy”. Este contiene parámetros fundamentales para el funcionamiento básico de un enemigo, como su cantidad de vida, la posición del jugador en la escena, o funciones como la de restar vida y voltear al enemigo en sentido contrario. A partir de este, las clases hijas se especializan con funciones que hacen que el enemigo se comporte de la forma específica que se había pensado para él. Para hacer sencillas las mecánicas para el jugador, para derrotarlos el jugador debe arrojarles proyectiles, o colisionar con su punto débil que está sobre la cabeza de cada uno (cosa que solo puede hacer saltando sobre ellos). Por último, al ser derrotado, justo antes de destruirse, el script Enemy” posee una función que instanciará un ítem como recompensa para el jugador (eligiéndolo aleatoriamente de una lista de objetos asignada con recompensas acordes a la importancia de ese enemigo). Según la forma en que el comportamiento del enemigo se ha especializado encontramos:

- **Enemigos Terrestres**: estos enemigos pretenden ser un obstáculo para el jugador en el suelo. Su comportamiento consiste en 2 estados (patrulla y ataque) que se alternan según si el jugador está a una distancia menor o igual a la establecida para atacar. En el estado patrulla, el enemigo se dedica a avanzar en una misma dirección hasta que se encuentra con una pared o no detecta más camino. En esta situación cambia de sentido y continúa moviéndose en dirección contraria. Si el jugador está lo suficientemente cerca, accede a su estado de persecución, con el que localizará su posición y tratará de dirigirse directamente hacia él por tierra. También pueden dejarse caer de sitios elevados en este estado, pero solo cuando detectan suelo a una altura inferior o igual a 4 bloques. Con cualquier otra altura que sea superior a esa se quedarán quietos. Según sus características se pueden encontrar:

- **Enemigo terrestre pequeño**: su tamaño es reducido, su velocidad es alta y solo posee un punto de vida.

- **Enemigo terrestre medio**: su tamaño es mediano, su velocidad es media y posee 2 puntos de vida.

- **Enemigo terrestre grande**: su tamaño es grande, su velocidad es baja y posee 3 puntos de vida.

- **Torreta**: este podría considerarse como un enemigo terrestre inmóvil. Ya que, está fijado al suelo y su principal forma de ataque es lanzar proyectiles hacia la posición del jugador. Concretamente, posee los 2 estados (Idle y Ataque). Igual que con los otros, permanece en Idle hasta que su distancia con el jugador es igual o menor a un valor prestablecido como rango de disparo. Entonces, calcula si este está dentro de su ángulo de tiro (entre 60 grados hacia arriba y 60 hacia abajo), y si lo está ataca instanciando un proyectil, (cuya dirección de movimiento es asignada) hacia la posición del jugador. Estos ataques los realiza en intervalos de un segundo y medio. Finalmente, igual que los otros, posee 2 puntos de vida y se le puede derrotar saltándole encima o disparándole.

- **Enemigos aéreos**: la función de estos enemigos es atacar desde el aire. Igual que los terrestres, su comportamiento consiste en los estados patrulla y ataque, que también se alternan según si el jugador está o no dentro de la distancia de ataque establecida. En el estado patrulla, el enemigo realiza un recorrido establecido por la posición de 3 nodos en el espacio. Es decir, su objetivo se establece como ese nodo hasta que lo consigue alcanzar, entonces es el próximo nodo el que se establece como siguiente objetivo. Este proceso se repite de manera cíclica con las 3 posiciones. Si el jugador está lo suficientemente cerca, accede a su estado de persecución, con el que tratará de agredirle dirigiéndose directamente hacia él. Según sus características se pueden encontrar:

- **Enemigo aéreo pequeño**: su tamaño es reducido, su velocidad es media y solo posee un punto de vida.
- **Enemigo aéreo grande**: su tamaño es mediano, su velocidad es baja y posee 2 puntos de vida.

- **Jefe Final**: es el último enemigo al que tiene que enfrentarse el jugador. Y, contrariamente a los otros descritos, este dispone de una alta cantidad de vida, y una inteligencia artificial que dirige sus diferentes estados de comportamiento (el funcionamiento de la cual se ha descrito en detalle más adelante). En general, se encarga de que el objeto realice múltiples ataques de manera secuencial y que acceda al más apropiado según el estado en el que se encuentre. Igual que con los otros enemigos su punto débil está en la cabeza y se daña saltando sobre ella. No obstante, las piñas no pueden dañarle (ya que la pelea se facilitaría mucho de esta forma) y, cada vez que recibe daño existe un pequeño periodo posterior en el que es invulnerable porque se debe recuperar. Finalmente, cuando este es derrotado, accede a una función que se encarga de diferentes tareas antes de destruirse. Concretamente, permite el movimiento de la cámara al jugador hasta la salida (introduciéndole nuevas coordenadas máximas), y reestablece la música de fondo del nivel (que había sido sustituido por la de la batalla contra el jefe).

5.5.7. Obstáculos y objetos

Para ampliar la interacción del jugador con los escenarios y para facilitar la creación de ciertas mecánicas que faciliten la jugabilidad, se han creado un conjunto de obstáculos con los que puede interactuar para modificar el *gameplay*. Estos son:

- **Checkpoints**: Es manejado por el script “Checkpoint” y su función principal es la de establecer un punto de reaparición para el jugador en su propia posición. Es decir, es activado cuando el jugador entra en contacto con él. Entonces, accede a la función del GameManager que permite cambiar la posición de reaparición del jugador introduciéndole como parámetros sus propias coordenadas. Además, también posee una función que modifica los valores mínimos y máximos que limitan el rango movimiento de la cámara. Los nuevos parámetros introducidos están medidos para impedir que la cámara retroceda más allá del *checkpoint*. Si recordamos que la propia cámara posee barreras para el jugador, con este método se impide que el jugador pueda retroceder hacia atrás (y así se facilita que se concentré en superar el nivel en vez de deambular por él).

- **Plataformas delgadas**: Al contrario que el terreno fijo del *tileset*, estas plataformas están programadas para que el jugador pueda atravesarlas saltando desde abajo y quedarse encima de ellas, o permitirle caer a través suyo si se mantiene agachado por un pequeño periodo de tiempo.

- **Cajas**: Este elemento es estático y se encarga de proporcionar al jugador un ítem cuando este consigue romperlo. Para ello, al igual que los enemigos, también posee un sistema de salud que indica cuanto daño debe recibir para que se rompa y un colisionador encima suyo que actúa como punto débil cuando el jugador salta sobre este. Además, que el jugador les lance piñas también provocará que pierdan puntos de salud. Una vez se han roto instancian un ítem que el jugador puede recoger. Según su tamaño se pueden encontrar: cajas pequeñas (que poseen 1 punto de salud y proporcionan cerezas) y cajas grandes (que poseen 3 puntos de salud y proporcionan diamantes o corazones).

- **Escaleras**: En esta ocasión no se trata de un objeto. Más bien se trata de un elemento *tileset* con un colisionador que indica si esa casilla posee algún elemento que permita trepar al jugador. Si en el script “PlayerMovement” se detecta que está colisionando con esta capa, cuando este pulsa hacía, se inicia la función que le permite trepar.

- **Trampolín**: La funcionalidad principal de este objeto estático es la de proporcionar al jugador un empuje vertical en el aire cuando el jugador salta encima de él.

- **Pinchos**: Este objeto posee un script en su colisionador que se encarga de provocar cierta cantidad de daño en el jugador. Se coloca en posiciones y situaciones estratégicas para complicar la partida al jugador.

- **Plataforma móvil**: la funcionalidad de este elemento es parecida al estado de patrulla de los enemigos voladores. La plataforma posee 2 nodos asociados en el espacio que especifican la trayectoria que debe seguir. Así, cada vez que alcanza un nodo, su script inicia un contador que la hace quedarse en esa posición por un corto periodo antes de repetir el proceso con el otro nodo.

- **Checkpoint de jefe**: El script de este objeto, “BossCheckpoint”, hereda las funcionalidades del script “Checkpoint”. Pero, incluye un par de funciones diseñadas para

preparar al jugador para su enfrentamiento contra el jefe en un período de 0,7 segundos tras haber entrado en contacto con él. Estas son activar al jefe; reproducir la música de batalla; establecer la propia posición del jugador como punto de reaparición (ya que con el rango de movimiento del jefe durante la pelea no parecía prudente establecer un punto de reaparición fijo durante esta), y fijar la posición de la cámara dentro de la habitación de duelo para utilizar sus propias paredes como recinto de combate.

5.5.8. Ítems

Para recompensar al jugador, se han creado una serie de ítems que servirán como recompensa o le ayudarán en ciertos aspectos de la partida. Estos son:

- **Cereza**: Este ítem incrementa en una unidad el contador de cerezas del jugador cuando se recoge. Es el ítem más abundante en cada escenario.

- **Diamante**: Este ítem incrementa en una unidad el contador de diamantes del jugador cuando se recoge. Es menos abundante que las cerezas y, por tanto, más preciado de conseguir.

- **Tirita**: La tirita rellena un 25% de la barra de salud del jugador cuando este la recoge. Para este caso, se eligió que el jugador solo la pudiera recolectar cuando su salud fuese inferior a la cantidad máxima establecida de 100 unidades (de esta forma, podría emplearla en el caso de que sufriera daño más adelante). Es decir, cuando el jugador entra en contacto con el ítem, este último consulta en el script “Healthbar” la cantidad de salud del jugador. Si esta es menor a 100 el ítem es recogido, la salud se incrementa en 25 unidades y el ítem es eliminado. En caso contrario, no interactúa.

- **Botiquín**: El botiquín rellena la barra de salud del jugador en un 50% cuando es recogido. Posee el mismo comportamiento que la tirita. Cuando el jugador entra en contacto con él, consulta en el script “Healthbar” la cantidad de salud del jugador. Y, si esta es menor a 100 el ítem es recogido, la salud se incrementa en 50 unidades y el ítem es eliminado. En caso contrario, tampoco interactúa.

- **Corazón**: Este ítem agrega un corazón al vector de vidas restantes del jugador cuando se recoge. Para este caso también se diseñó para que solo se pudiese recoger cuando la cantidad de vidas del vector sea inferior al máximo establecido. Para ello el ítem revisa la cantidad de vidas del script “LifeCount”. Y, si esta es menor a 4, es recogido por el jugador, se incrementa una vida en el vector y el ítem es destruido. De lo contrario no interactúa.

- **Piña**: Este ítem añade un proyectil al contador de munición del jugador cuando se recoge. Para este caso, como también es limitada, se quiso que la munición solo pudiera recogerse cuando fuese necesario. Por tanto, al entrar en contacto con el jugador, el ítem solicita al “ItemManager” la cantidad de munición que posee el jugador antes de poder

añadirlo. Si el contador está lleno, se ordena al ítem no desaparecer ni interactuar. En caso contrario, el script lo agrega y el ítem es destruido.

- **Montón medio de piñas**: Agrega 5 piñas a la cantidad de munición recogida. Igual que con la piña individual, solicita al "ItemManager" la cantidad de munición que posee el jugador antes de poder añadirlo. Si la cantidad de munición está llena (es igual a 15) este ítem no desaparecerá ni se agregará la cantidad al contador.

- **Montón grande de piñas**: Agrega 10 piñas a la cantidad de munición recogida. Igual que en las anteriores, también solicita al "ItemManager" la cantidad de munición que posee el jugador antes de poder agregarlo. Si la munición está llena este ítem no desaparecerá ni se agregará la cantidad al contador.

5.6. Generación procedural de laberintos

Para el método de creación procedural de laberintos se decidió aplicar la teoría de grafos como base para generarlos. De esta forma, el elemento laberinto funciona como un "grafo de cuadrícula cuadrada" (uno cuyos nodos solo se pueden conectar con sus vecinos más cercanos de manera vertical u horizontal), y los nodos que lo forman (y las distintas conexiones entre ellos), representan las encrucijadas y recovecos que se pueden encontrar recorriéndolo.

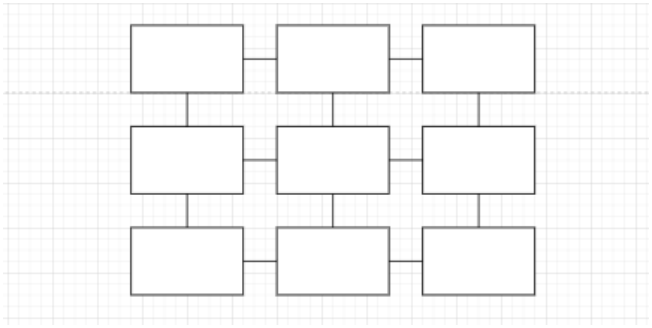


Figura 25: Ejemplo de un "grafo de cuadrícula cuadrada" de 3x3 nodos

Para implementar esta idea, se crearon 2 nuevos objetos: "MazeManager" y "MazeNode". "MazeManager" se ocupa de crear y gestionar el conjunto del grafo. Y, "MazeNode" es un objeto que simula el comportamiento de uno de los nodos del grafo. Es decir, incluye la lista de nodos vecinos a los que tiene acceso; el estado en el que se encuentra dentro del proceso de búsqueda, y las variables necesarias para establecer qué cámara se instanciará con el algoritmo que crea el recorrido.

El recorrido del laberinto se genera implementando el algoritmo de búsqueda DFS. Este algoritmo se emplea básicamente para recorrer todos los vértices que conforman un grafo a partir de un nodo llamado "raíz". No obstante, para esta situación se ha decidido alterar parte

de su funcionalidad para que tanto la asignación del nodo raíz como la búsqueda dentro de la estructura sean aleatorias.

El funcionamiento completo del proceso es el siguiente: primero, “MazeManager” instancia objetos “Node” separados por una distancia horizontal y una altura idénticas a las de las habitaciones que se crearán. La cantidad concreta de objetos “Node” que se creará viene determinada por los parámetros de altura y anchura del grafo que se le introducen (así el grafo puede ser de dimensiones como 2X2 nodos, 3X8 nodos, 4X4 nodos, etc). Al mismo tiempo, los nodos instanciados se almacenan en un vector. Una vez ya se han creado todos los nodos, se recorre este vector y se asigna a cada nodo los vecinos con los que conecta. Debido a la forma del grafo, cada nodo puede tener entre 2 y 4 vecinos dependiendo de su posición.

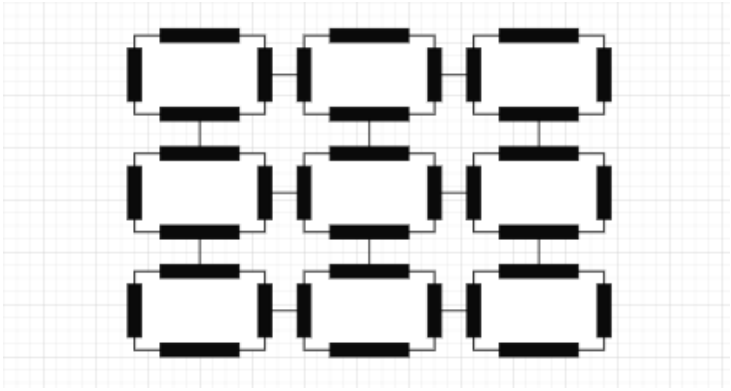


Figura 26: Grafo instanciado por “MazeManager” de 3x3 nodos

El siguiente paso es ejecutar el proceso que crea el recorrido del laberinto. Para ello, primero elige un nodo de forma aleatoria de entre los almacenados en el vector previo, establece su estado como “Actual” y lo asigna a las variables del nodo de inicio o “StartNode”, y “actualNode” (que registra cuál es el nodo en el que está el algoritmo). A partir de ahí, se ejecuta el algoritmo de exploración aleatoria DFS, que básicamente consiste en un bucle cuyo mecanismo es revisar los nodos vecinos que tiene cómo disponibles el nodo seleccionado como actual.

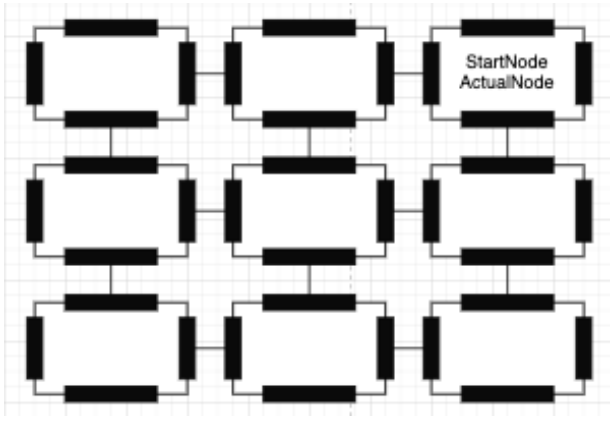
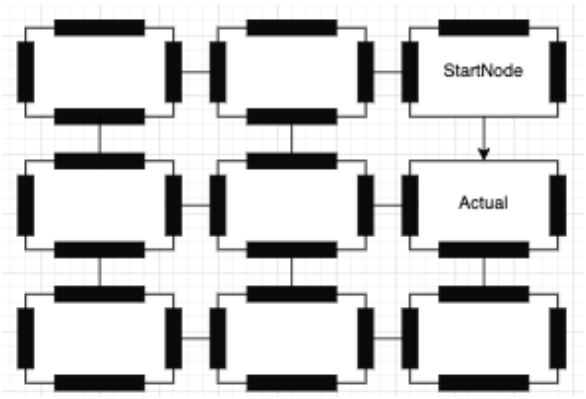


Figura 27: Nodo derecho superior seleccionado como “StartNode” y establecido como “ActualNode”.

Si el nodo actual posee vecinos disponibles, elige uno de ellos de forma aleatoria y lo asigna como nodo actual. Mientras, el nodo actual previo es almacenado en 2 pilas con los nombres "*visited*" (que registrará los nodos que el algoritmo haya revisado) y "*path*" (que se ocupará de almacenar los nodos que conformen el camino desde el nodo inicial "StartNode" hasta el último nodo sobre el que se tenga constancia). Además, cambia los estados del antiguo nodo actual a "*Searching*" y del nuevo nodo actual a "Actual".
Por último, se debe mencionar que, a medida que se elige el próximo vecino disponible, se revisa en qué posición está el nodo actual respecto a él. Y, en consecuencia, modifica unas variables correspondientes a ese nodo actual y al próximo que actúan como "paredes". Que indicarán por dónde ha pasado el algoritmo de búsqueda, ya que en este paso el proceso las "derriba" según su dirección (estableciéndolas a falso).



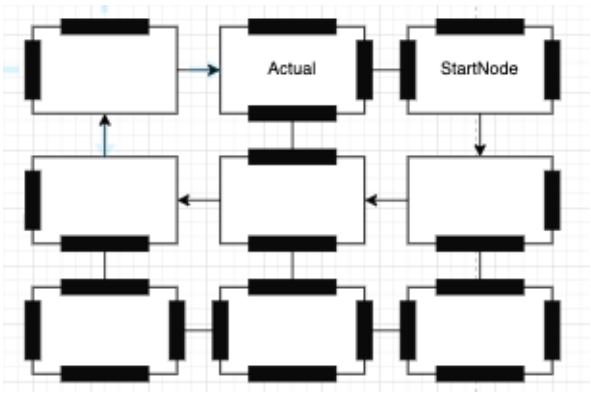


Figura 28 (Izquierda): Se selecciona el nodo de abajo como el próximo y se establece como actual.
Figura 29 (Derecha): El algoritmo avanza hasta que llega a un nodo sin vecinos disponibles.

Por otra parte, si el nodo no posee vecinos disponibles, la pila "*path*" se desapila, y se elige el nodo previo como actual para repetir el proceso de revisión de vecinos disponibles. En este proceso el nodo desapilado se asigna al estado "Done" y el nuevo nodo elegido se asigna a "Actual". Cada vez que retrocede se revisa la cantidad de nodos almacenados en *path*. Así, el nodo que corresponda a la posición más alta de la pila es asignado como nodo de salida o "ExitNode" para el jugador (así siempre es el nodo más lejano respecto al que el jugador aparece). La condición de salida de este bucle es que la cantidad de nodos almacenados en la pila "*visited*" sea igual a la cantidad de nodos almacenados en el vector previamente mencionado (solo así se asegura de que no queden nodos por visitar).

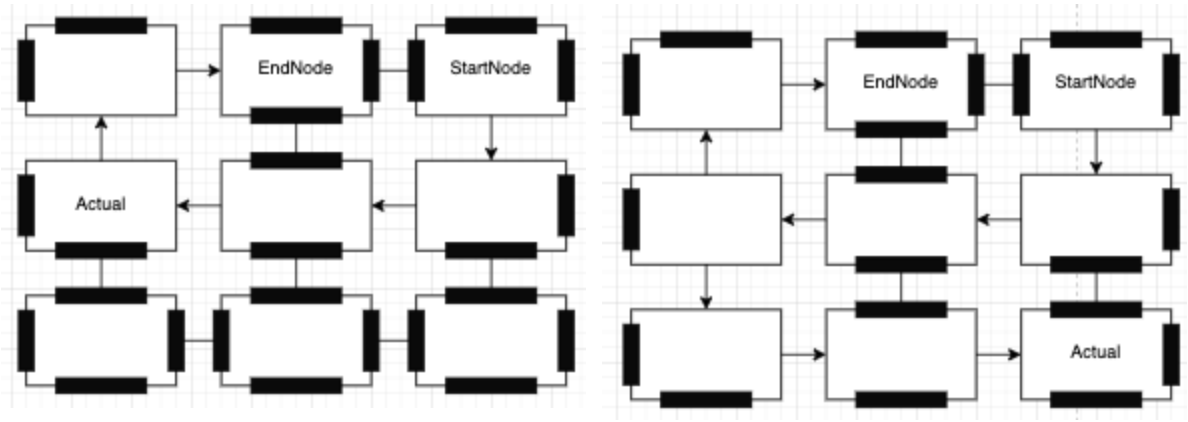


Figura 30 (Izquierda): El algoritmo retrocede hasta el último nodo con vecinos disponibles.
Figura 31 (Derecha): El algoritmo recorre un camino alternativo y termina de recorrer el grafo.

Una vez este camino ya ha sido definido, el último paso es ejecutar el proceso que se ocupará de “construir” las habitaciones en función de qué paredes han sido “tumbadas” y cuales permanecen “en pie” en cada nodo. Para esto “MazeGenerator” recorre el vector principal de nodos y ejecutar en cada uno el método para construir la cámara.

Así, al final del método de DFS, todos los nodos quedan con algunas de estas “paredes” asignadas a falso y otras no. Tras cada actualización que realiza “MazeManager” sobre estas variables, los nodos sobre las que se aplica revisan los cambios y, según sus estados, se establecen a ellos mismos un estado que definirá qué tipo de habitación ha quedado. Como nunca aparecerá la posibilidad de que todas las paredes queden levantadas, existen 15 posibles cámaras distintas: Cross, *horzCorridor, vertCorridor, RightUpL, LeftUpL, RightDownL, LeftDownL, RightT, LeftT, UpT, DownT, CaveLeft, CaveRight, CaveUp y CaveDown*. Por otro lado, “MazeGenerator” posee los “planos” de estas 15 cámaras.

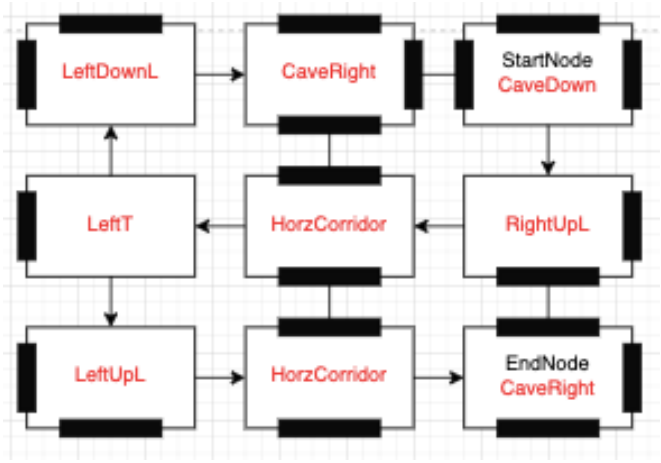


Figura 32: Estado final del grafo tras aplicar el algoritmo.

Así, con este método “MazeGenerator” revisa el tipo de habitación de cada nodo y les asigna uno de estos planos según su propio estado. Al mismo tiempo, “MazeNode” recoge este plano y construye la cámara que le corresponda. Como aclaración, la función edifica la cámara con

los bloques de los diferentes Tilesets, empezando desde la posición espacial que ocupa su objeto nodo en escena. De esta forma cada cámara se crea de izquierda a derecha y de arriba abajo. Así siempre sabe en qué posición concreta debe instanciarse.

Finalmente, en el caso de que sea un nivel que solo conste de un laberinto procedural, se ejecutará una función que se encargue de instanciar el jugador en el nodo seleccionado como "StartNode" y la línea de meta en el nodo seleccionado como el final del laberinto, "endNode". Además de establecer los límites a la cámara del jugador para que quede centrada en el escenario.
Así, como resultado final, se obtiene un laberinto conformado por distintas habitaciones que será diferente cada vez que la escena se ejecute.

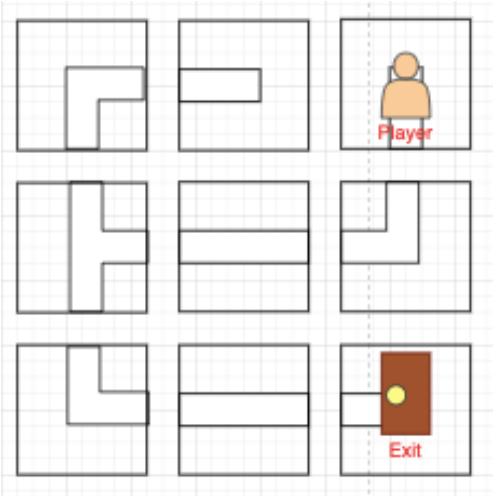


Figura 33: Laberinto resultante tras aplicar la función de construcción.

5.7. IA del jefe final

Para que el jefe suponga un reto final "digno", ha sido implementado con diferentes estados que aplicarán ataques contra el jugador. Estos se ejecutarán de forma consecutiva, de manera que, una vez termine con un ataque, empezará con el siguiente. Los diferentes estados que puede adoptar son: "*Idle*", "*Shot*", "*Jump*", "*Charge*" y "*PunchRain*".

Para que estos ataques no se lancen de forma arbitraria y aprovechen bien su capacidad, se ha creado una máquina de estados en el Script "Boss" para decidir cuál es el más adecuado para ejecutar en cada situación. De esta forma, permitirá aprovechar bien su propia posición y la del jugador para lanzar el ataque que sea más efectivo. A continuación, se ofrece un diagrama que muestra comportamiento de la máquina de estados del jefe.

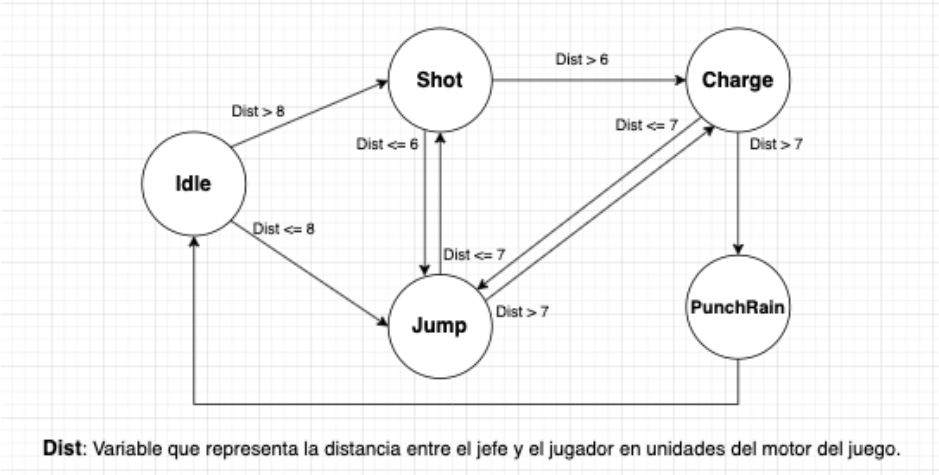


Figura 34: Diagrama de estados de ConcreteMan

Así los posibles estados que puede tener el jefe son:

- ***Idle***: Es un estado en el que el jefe final no realiza ningún movimiento. Este se asigna como estado inicial para que pueda acceder a los estados de corta o larga distancia más efectivos ("*Jump*" y "*Shot*" respectivamente). Ya que, al inicio de la pelea, el jefe se encuentra en la esquina opuesta al jugador en la sala de duelo. Dependiendo si este decide quedarse quieto o atacar inmediatamente, es importante que el jefe pueda contratacar con la misma efectividad.

- ***Shot***: El ataque de este estado consiste en que el jefe realice 3 disparos simultáneos (en las direcciones, vertical, horizontal y diagonal) en 3 ráfagas consecutivas. Debido a que este ataque solo funciona bien a larga distancia, solo puede ser accedido por otros estados en los que el jugador esté a una distancia lo suficientemente lejana. Una vez termine puede acceder al estado "*Charge*", en caso de que el jugador esté a una distancia lo suficientemente lejana. O acceder a "*Jump*" si es que este no está lo suficientemente lejos.

- ***Jump***: En este estado el jefe se posiciona sobre la última posición que ha registrado del jugador y se deja caer en un intento por aplastarlo. Repite este proceso 3 veces hasta que se le ordena cambiar de estado. Como el tiempo con el que el jefe se posiciona sobre el jugador es directamente proporcional a su distancia, este ataque solo es accedido por estados cuyas distancias con el jugador sean más bien pequeñas. Así es más efectivo, porque el jugador no posee tanto margen para esquivarlo. Una vez termine, puede acceder al estado "*Charge*", en caso de que el jugador también esté a una distancia lo suficientemente lejana. O acceder a "*Shot*" si está en una posición más bien cerca.

- ***Charge***: Este estado de ataque el jefe arremeta hacia la posición del jugador a gran velocidad. Y está diseñado para que no se detenga hasta que detecte que ha colisionado con uno de los muros. De esta forma, cuando termine siempre acabará en una de las 2 esquinas de la habitación de duelo. Este hecho facilita saber cuál debería ser su próximo ataque según la distancia con el jugador: volver al estado "*Jump*" en caso de que el jugador no esté

demasiado lejos, o acceder a "*PunchRain*" en caso de que sí lo esté y disponga de espacio para lanzar un ataque de media-larga distancia.

- ***PunchRain***: Este estado consiste en un solo ataque, en el que el jefe, sin moverse del sitio, da un puñetazo y provoca que caigan pedazos de roca sobre el escenario. La posición en la que se generan estas rocas es totalmente aleatoria, de manera que el jugador nunca estará seguro de donde estará a salvo. Se diseñó para hacer que el jugador se distrajera del jefe al presentársele una amenaza más directa y que esto le obligara a ponerse a salvo. Como este ataque es estático se realiza después del estado "*Charge*", su posición al terminar vuelve a ser alguna de las esquinas (igual que al principio de la pelea). Así, el cambio más lógico era devolverle brevemente al estado "Idle" y que pudiese volver a acceder a los ataques de corta y larga distancia más efectivos de los que dispone.

5.8. Diseño de niveles

Antes de describir cómo se han diseñado cada uno de los niveles y qué elementos se han decidido incluir, es importante explicar qué tipos de niveles se han diseñado y cómo se los encontrará el jugador. Así, según la manera en que se haya implementado el nivel, existen 3 clasificaciones para estos:

- **Básico**. Se trata de un nivel con un terreno creado a mano, sin la inclusión de un laberinto. Así, el camino es invariable con cada partida. El único nivel existente que cumple con esta clasificación es el tutorial, ya que solo se diseñó como una guía para enseñar al jugador las diferentes habilidades y como una primera toma de contacto del juego.

- **Procedural.** Es un nivel cuyo terreno es únicamente el propio laberinto generado de manera procedural. Dentro del contexto del juego se ha justificado su existencia dando a entender al jugador que el nivel transcurre dentro de unas cuevas subterráneas. En este proceso "MazeGenerator" crea los nodos de forma procedural, e instancia el objeto "*Player*" y un *checkpoint* en el nodo que se ha escogido como raíz, y el objeto "*Goal*" en el nodo más alejado a este. Al mismo tiempo, asigna los límites a la cámara para que quede dentro del escenario. Para ello tiene en cuenta las dimensiones del grafo y el espacio que ocupan cada una de sus salas.

- **Híbrido**. Este último tipo de nivel se trata de un nivel básico en el que se ha incluido un laberinto procedural en una sección concreta. Es decir, combina secciones de un nivel creado a mano con segmentos en los que se crea un laberinto procedural distinto para cada partida. De esta manera, se pretende hacer que la jugabilidad sea más interesante para al jugador al no saber con qué diseño de laberinto se encontrará en su partida, al mismo tiempo que se mejora la re-jugabilidad al incluirlo. Para conseguirlo, se realizaron ciertas modificaciones en los Scripts "MazeGenerator" y "MazeNode". En el caso del primero, los nodos pueden asignarse manualmente para definir la estructura que tendrá el grafo. Además de que la función para instanciar al jugador y la meta final en los nodos "startNode" y "endNode" se

omite. Y, en el caso del segundo, se puede omitir el proceso de construir la cámara (de esta forma siempre existirá una entrada y una salida fijas para esta sección).

5.8.1. Nivel 0 (Tutorial)

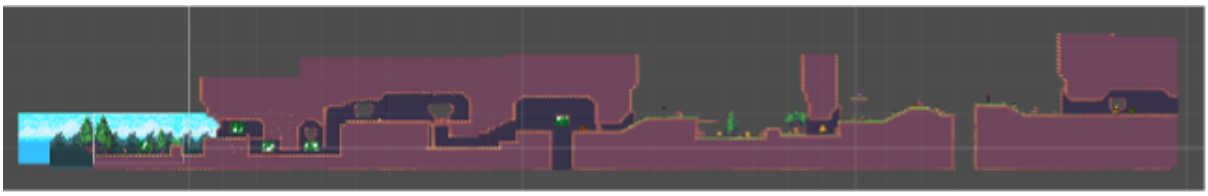


Figura 35: Nivel 0 (El tutorial)

Está diseñado para ser el nivel en el que se le enseñará al jugador todas las habilidades y el uso de ciertos objetos como los *checkpoints*, ya que también será el único al que el jugador tenga acceso en un principio. Cada habilidad es mostrada con pictogramas, y están introducidas en situaciones donde el jugador tendrá que aplicarlas o no podrá continuar. Cuando el jugador supera la zona de habilidades, se introducen a los enemigos terrestres. El resto del nivel está pensado para que pueda aprender sobre sus tamaños, qué tipos hay y cómo derrotarlos.

5.8.2. Nivel 1

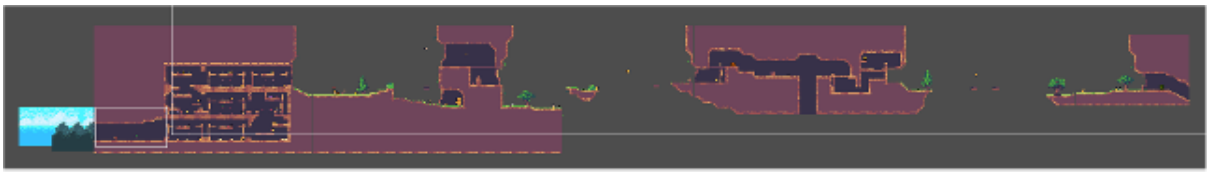


Figura 36: Nivel 1

Este ya se considera el primer nivel oficial y también es el primer nivel híbrido. Al principio del nivel, se introduce al jugador el primer laberinto de generación procedural, de 3x3 cámaras. Con la intención de presentarle este elemento de forma temprana. Cuando el jugador supera el laberinto, regresa a la superficie. En los otros apartados del nivel se introducen el resto de los elementos y enemigos que no se habían incluido en el tutorial. Concretamente, los enemigos aéreos, el enemigo torreta, las plataformas móviles y los trampolines.

5.8.3. Nivel 2 (Laberinto subterráneo)

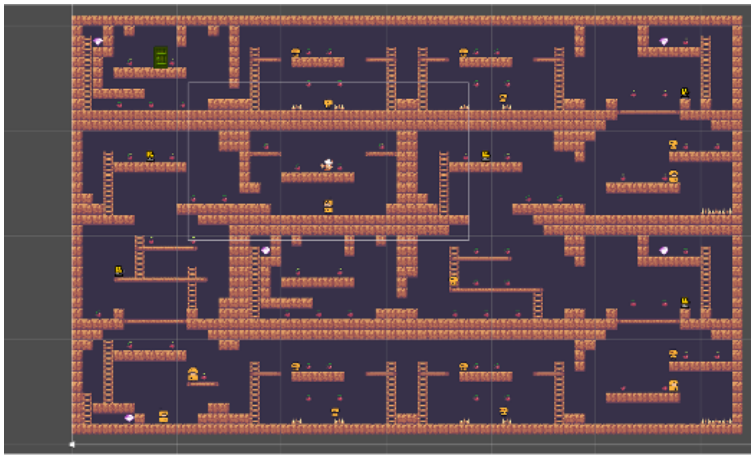


Figura 37: Nivel 2

Este es el primer nivel procedural que introduce el juego. En él se instancia un laberinto de tamaño mediano, 4x4 cámaras, que el jugador debe recorrer para superar. El tamaño relativamente pequeño o mediano de estos laberintos se ha decidido porque en la jugabilidad, a veces, se perciben como repetitivos. La intención tras estos niveles es que el jugador los perciba como un reto. Ya que, como se ha decidido que no serán muy grandes, se ha optado por aumentar un poco su dificultad haciendo que solo exista un *checkpoint* al principio.

5.8.4. Nivel 3

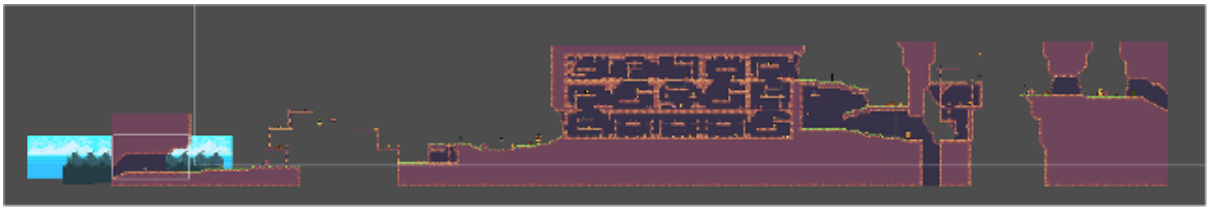


Figura 38: Nivel 3

El segundo nivel con configuración híbrida. En esta ocasión el laberinto se encuentra justo a la mitad del nivel, y cuenta con un tamaño grande (de 6x3 cámaras). Se optó por este diseño porque, sabiendo que este sería un obstáculo relativamente difícil de superar, obligaría al jugador a planear cómo superar llegar hasta ahí con la máxima cantidad de vida y recursos que pudiese.
Para las otras secciones del nivel, se ha procurado utilizar todos los tipos de enemigos cómo en el nivel 1. Pero, introduciéndolos en nuevas situaciones y circunstancias complejas para que hagan que el jugador no le sea tan fácil acabar con ellos.

5.8.5. Nivel 4 (Laberinto subterráneo)

Figura 39: Nivel 4

Se trata de un segundo nivel procedural con un laberinto de un tamaño ligeramente más grande al del nivel 2 (de 4x5 cámaras).

5.8.6. Nivel 5 (Nivel de jefe)

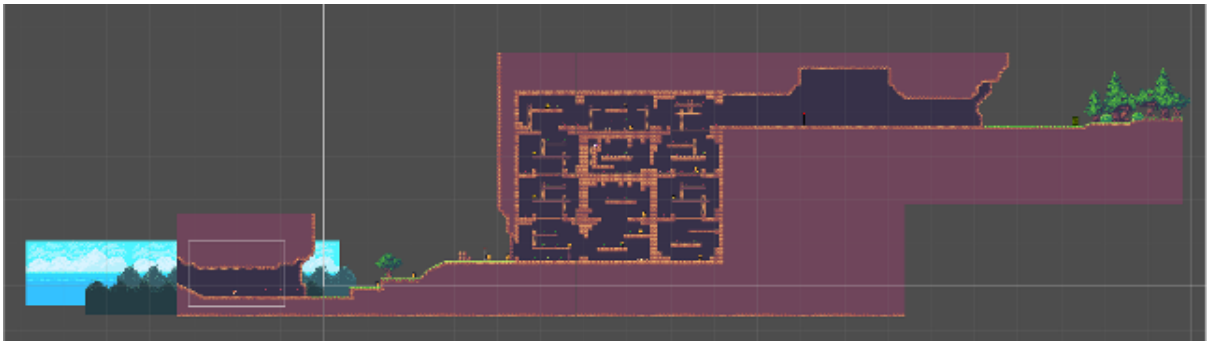


Figura 40: Nivel 5

Para el último nivel del juego se ha optado por volver a utilizar la estructura híbrida. Este nivel se divide en 3 secciones: una breve zona en la superficie, un laberinto procedural de 3x4 cámaras, y una sala de duelo en una cueva donde se dará el enfrentamiento contra el jefe. Se decidió usar esta configuración antes de la cámara de duelo para elevar un poco el nivel de dificultad. Ya que, en estos niveles, cada vez que el jugador muere, el terreno antes de llegar al jefe se suele mantener invariable (haciendo que cada vez se vuelva un poco más fácil de superar). Así, con cada nueva partida, el jugador nunca sabrá con qué configuración del escenario se encontrará y no le resultará tan cómodo llegar hasta el jefe.

6. Implementación

6.1. Requisitos de instalación

Para este producto no se requiere ninguna instalación adicional para ejecutar el juego. Ya que, este únicamente consiste en un archivo comprimido .zip que contiene el fichero ejecutable .exe del juego.

Los requisitos mínimos que debe cumplir el dispositivo para ejecutar el juego son que este posea un sistema operativo Windows_x86 (o de 32 bits). El cual deberá disponer de un espacio libre de 98.7 MB en su disco duro. En cuanto al resto de componentes, al tratarse de un plataformas en 2D, la ejecución del juego no consume una cantidad demasiado alta de recursos. Concretamente, el juego requiere un espacio de memoria de 98.4MB para ejecutarse. Por esta misma razón, el juego funcionará correctamente tanto en dispositivos con componentes de gama baja, como en los de gama alta. Finalmente, también serán necesarios un teclado y un ratón para acceder a los controles.

6.2. Instrucciones de instalación

Para instalar el juego, primero, se debe descargar el archivo comprimido en zip (que contiene la carpeta con los diferentes documentos). Entonces, bastará con descargar este archivo en el dispositivo deseado (siempre teniendo en cuenta las especificaciones mencionadas previamente). Y, finalmente, para abrir el juego se deberá buscar el ejecutable .exe, con el nombre “Underfox” entre los diferentes documentos, e iniciarlo haciendo doble clic en él.

7. Demostración

7.1. Instrucciones de uso

Los controles asociados al personaje jugador son los siguientes:

- Para mover al personaje por el escenario se utilizan las teclas “A” y “D”, (para desplazarlo a la izquierda o a la derecha respectivamente), o las flechas de izquierda o derecha del teclado.

- Para hacer correr al personaje jugador (hacer que su velocidad aumente un poco) se debe mantener pulsada la tecla “Shift” mientras este avanza en algún sentido con las teclas mencionadas previamente.

- Para hacer que el personaje jugador se agache, o atraviese plataformas delgadas, se debe pulsar la tecla “S” o la flecha hacia abajo del teclado. Solo permanecerá en este estado mientras cualquiera de estas teclas se mantenga pulsada.

- Para hacer saltar al personaje jugador se debe pulsar la tecla espacio del teclado. El jugador solo posee 2 saltos que podrá realizar de forma consecutiva.

- Los proyectiles de piña recogidos por el jugador se lanzan pulsando la tecla “T”.

- Para moverse entre los diferentes menús del juego o los niveles disponibles, el jugador deberá utilizar el ratón para interactuar con los botones de estas interfaces.

- Para parar la partida el jugador puede abrir el menú de pausa pulsando la tecla “Esc”. Este le ofrecerá diferentes opciones, entre las cuales estarían abandonar la partida accediendo al menú de niveles o al principal, o reanudarla haciendo clic sobre el botón reanudar o pulsando otra vez la tecla “Esc”.

7.2. Guía de usuario

Cuando se inicia el juego, el primer elemento con el que se encuentra el jugador es el menú principal. Este contiene las opciones de iniciar una nueva partida (con el botón “Nueva partida”) que borrará el progreso anterior del jugador en los niveles desbloqueados; continuar con la partida que estaba guardada, accediendo al menú de niveles que contiene los niveles disponibles y desbloqueados por el jugador, (con el botón “Continuar partida”); la posibilidad de acceder a las configuraciones de audio y gráficos mediante el botón “Opciones” (el volumen de la música y el de los efectos de sonido, y la opción de pantalla completa) para que el jugador pueda regularlas a su gusto, y, finalmente, la opción de cerrar la aplicación y abandonar el juego con el botón de “Salir del juego”.

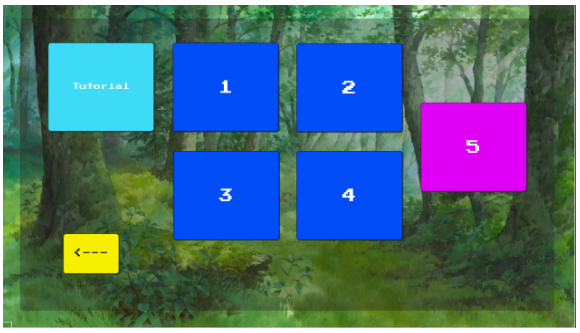


Figura 41 (Izquierda): Menú principal de Underfox

Figura 42 (Derecha): Menú de niveles de Underfox

El único nivel desbloqueado al empezar el juego es el tutorial. El jugador puede acceder a él

A partir de aquí, ya podría empezar a jugar. El personaje que el jugador maneja es un zorro. Este tiene la capacidad de saltar, correr, agacharse, rebotar en las paredes, recoger proyectiles e ítems, y disparar. Su sistema de vida consiste en una barra que se descarga al recibir daño y unas 4 vidas. Cuando la barra llega a 0, una de las 4 vidas se descuenta y desaparece. Si pierde todas las vidas se declara el fin de la partida y el jugador tiene la opción de reintentarlo o abandonar. Con el sistema de disparo el jugador puede disparar proyectiles (piñas en este caso) a diferentes elementos. Como limitación, existe un número definido de proyectiles que puede lanzar. Por último, la mecánica principal para eliminar a un enemigo es que el jugador salte encima suyo.

El objetivo de cada nivel es conseguir llegar hasta la meta, ya sea desplazándose por el escenario hacia la derecha en el caso de los niveles de superficie; o, resolviendo el laberinto de los niveles subterráneos hasta llegar a su localización en la cámara final. Al alcanzar el objetivo, al jugador se le desbloqueará la posibilidad de acceder al siguiente nivel. El juego se completará una vez el jugador llegué hasta el último nivel y derrote al jefe final. El juego incluye una cantidad de 5 niveles (sin incluir el nivel del tutorial). Para cada uno de estos se ha planeado que contengan una sección que se genere de forma procedural, o que la sección del laberinto constituya todo el nivel. Estas zonas se han diseñado para que adopten una estética de cueva, justificando así su aspecto laberíntico. De esta manera se pretende aportar cierta re-jugabilidad para el jugador, al hacer que desconozca con qué distribución del nivel se va a encontrar.
En el caso del último nivel está más justificado porque, se había concebido para que funcionara alrededor del enfrentamiento del jugador contra el jefe final. Así, agregar esta aleatoriedad para el jugador antes de enfrentarse al reto definitivo del nivel conseguiría que supusiera un pequeño desafío en función de cuánto le cueste superarlo.

El camino que conduce hasta la meta contiene obstáculos que el jugador deberá sortear para superarlo. Además, también existe un conjunto de enemigos situados a lo largo de este recorrido cuya función es la de hacer más difícil el nivel al jugador. Su comportamiento y ataques varían según la zona que cubren. Aunque, todos pueden ser derrotados de la misma

forma: saltando sobre su cabeza hasta que pierdan toda su salud, o lanzándoles proyectiles de piña para derrotarlos (a excepción del jefe final). Una vez eliminados instanciarán una recompensa antes de desaparecer.

Si el personaje jugador es herido se reproducirá una animación de daño; se restará un porcentaje de su barra de vida acorde a la magnitud del daño sufrido, y por un período de 1.7 segundos se volverá invulnerable a otros ataques (no obstante, en este estado él tampoco puede provocar daño en los enemigos). Por otro lado, si el jugador pierde toda su barra de vida, o cae directamente a una zona de vacío, perderá una de sus vidas restantes y reaparecerá en el último *checkpoint* al que llegó. En el caso de que llegará a perder todas sus vidas durante la partida, su personaje quedaría destruido y aparecería el menú de derrota por pantalla. En este se ofrecen las posibilidades de reintentar el nivel, o volver al menú principal (en el caso de que el jugador quiera abandonar el juego). Pero, si consigue superar el nivel llegando hasta el final, el juego se pausará y aparecerá el menú de victoria por pantalla. El cual mostraría las opciones de avanzar al siguiente nivel, o regresar al menú de niveles.

En caso de que necesitara detener el juego durante una partida, el jugador puede hacerlo accediendo al menú de pausa pulsando la tecla “Esc”. Además de parar el tiempo de ejecución de la partida, este menú ofrece las opciones de reanudar el juego (pulsando el botón “Reanudar”); regresar al menú de niveles (pulsando el botón “Niveles”); o, regresar al menú principal (pulsando sobre el botón “Salir al Menú principal”).

Figura 43 (Izquierda): Menú de victoria del nivel

Figura 44 (Derecha): Menú de derrota del nivel

Por último, además de elementos como las plataformas y los trampolines, se han incluido una serie de objetos a lo largo del nivel para servir de ayuda al jugador, siendo estos: los *checkpoints*, las cajas y los ítems. Los *checkpoints* son objetos que se encargan de establecer su propia posición como posición de reaparición para el jugador. Así, cuando este muere durante la partida reaparecerá en la localización del último *Checkpoint* con el que tuvo contacto. Las cajas son objetos que pueden ser destruidos de la misma manera que los enemigos (saltando encima o arrojándoles piñas), y al hacerlo hacen aparecer un ítem como recompensa. Los ítems, por otro lado, son elementos que ayudan al jugador incrementando

un aspecto que podría tener bajo (como su salud o su munición), y desaparecen de la escena tras cumplir su función. Los diferentes ítems que hay son:

- **Cereza**: incrementa una unidad el contador de cerezas del jugador cuando se recoge. Es el ítem más abundante en cada escenario.

- **Diamante**: Este ítem incrementa en una unidad el contador de diamantes del jugador cuando se recoge. Es menos abundante que las cerezas y suele encontrarse en sitios menos accesibles.

- **Tirita**: La tirita rellena un 25% de la barra de salud del jugador cuando este la recoge. Solo puede recogerse cuando la salud del jugador es inferior a la cantidad máxima (de esta forma, podría emplearla en el caso de que sufriera daño más adelante).

- **Botiquín**: El botiquín rellena la barra de salud del jugador en un 50% cuando es recogido. Posee el mismo comportamiento que la tirita (solo se recoge cuando su cantidad de salud es inferior a la máxima).

- **Corazón**: Este ítem agrega un corazón al vector de vidas restantes del jugador cuando se recoge. AL igual que los anteriores, solo puede recogerse cuando la cantidad de vidas del vector es inferior al máximo establecido.

- **Piña**: Este ítem añade un proyectil al contador de munición del jugador cuando se recoge. Los ítems de munición solo pueden recogerse si su cantidad es menor a la máxima establecida de 15.

- **Montón medio de piñas**: Agrega 5 piñas a la cantidad de munición recogida. Igual que con el ítem anterior solo puede recogerse si su cantidad es menor a la máxima establecida, 15.

- **Montón grande de piñas**: Agrega 10 piñas a la cantidad de munición recogida. Igual que en los casos previos, solo puede recogerse si su cantidad es menor a la máxima establecida, 15.

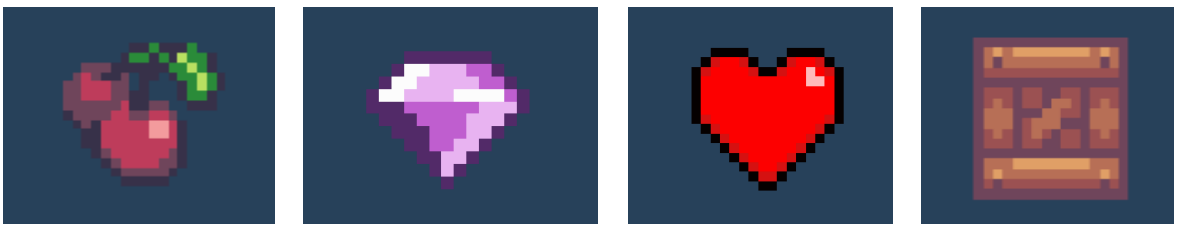


Figura 45 (Izquierda): Ítem cereza
Figura 46 (Central-Izquierda): Ítem diamante
Figura 47 (Central-Derecha): Ítem corazón
Figura 48 (Derecha): Caja pequeña

8. Conclusiones y líneas de futuro

8.1. Conclusiones

8.1.1. Lecciones aprendidas

Como se comentó en la sección de “Justificación del área” antes de este proyecto el autor sentía una gran curiosidad por cómo funcionaban los entresijos del desarrollo de un videojuego. Ya conocía una pequeña parte del procedimiento de la creación de un juego a través de algún libro que había leído, o gente que había compartido su experiencia por internet. Pero, experimentarlo de primera mano ha conseguido que ahora vea toda esta información con un ángulo diferente.

Por ejemplo, siempre había escuchado que para la realización de un juego era necesaria la colaboración de diferentes disciplinas para conseguir su lanzamiento. Siempre había entendido a qué se refería esta afirmación. Pero, ahora que ha tenido que desarrollar todo un juego completo y funcional, por fin puede entender el peso de esta. Ya que, el trabajo total para alcanzar su lanzamiento ha sido: programar todo el código del juego, asegurándome de que este era totalmente funcional; diseñar la estructura general que poseerá el producto; aprender a utilizar el entorno de desarrollo para el juego mientras este se desarrollaba; buscar los *sprites* para todos los objetos y elementos de los niveles (o en el caso de que no se encontrasen diseñarlos); hallar todas las pistas de sonido para las diferentes acciones del juego, o diseñar los fondos del menú principal o el menú de niveles entre otros. Son muchas tareas y algunas (como el apartado de audio o el diseño de ciertos gráficos) requieren cierto nivel de especialización para que su resultado se parezca al que se esperaba en un principio.

Pero, es posible que la lección más importante que el autor ha aprendido a lo largo del proceso es que: a la hora de crear un videojuego, es vital conocer su contexto de mercado y definir claramente sus características antes de empezar con su desarrollo. Esto se aprendió en la sección dedicada al estudio de mercado. El autor entendió cómo la tasa de éxito de un juego se determina prácticamente en el momento en el que se escoge su género y su plataforma de lanzamiento. De manera que, conocer cuál va a ser la oferta y la demanda que se encontrará en el mercado, o saber cuál va a ser tu competencia más directa, es tan importante como el proceso desarrollo del propio juego.

8.1.2. Objetivos

Si se hablan de los objetivos, se puede afirmar que se han conseguido cumplir la mayoría de estos. Se ha conseguido crear un sistema que crea laberintos aleatorios en determinados niveles que varían en cada partida y el jugador puede recorrer; se ha creado una IA para los diferentes tipos de enemigos entre otros, y en general se ha desarrollado una del juego sólida y sin errores del juego propuesto. No obstante, si se retrocede lo suficientemente atrás a lo que se propuso en un principio, sí se observa que existen objetivos que no se han podido cumplir debido a que el autor no conocía bien cuál iba a ser la magnitud total de este trabajo. Por ejemplo, la cantidad de niveles que se han implementado (pues se esperaban alrededor

de 10, pero han acabado siendo un total de 6); la inclusión de cinemáticas en el juego, o la exportación del producto a las plataformas de Pc y Smartphone (ya que, al final solo se ha podido realizar a la primera). Estos cambios se realizaron debido a la falta de tiempo durante el desarrollo del proyecto. No han afectado mucho al resultado final del juego, pero sí han sido un poco influyentes en aspectos como: la cantidad de personas a la que llegará el producto, su interés en obtenerlo, y su tasa de satisfacción al jugarlo (puesto que, aunque solo sean unos niveles, los clientes tienden a preferir juegos más largos).

8.1.3. Planificación y metodología seguida

En general, se ha se ha seguido al detalle la planificación del juego a lo largo del proyecto. Aunque, como ya se comentaba, la falta de experiencia del autor provocó que no pudiese medir bien cuál iba a ser la magnitud total del proyecto. Lo que ha provocado diferentes cambios a lo largo de distintos apartados, e incluso cierta falta de tiempo en la realización de algunos otros. Por ejemplo, la adición del sistema de generación de niveles procedurales fue algo a lo que se decidió añadir poco después de acabar con el plan de proyecto. Esto provocó que algunos de los obstáculos pensados para el juego no se llegasen a crear, además de retrasar parte de las otras tareas posteriores.
O también, la creación del jefe final del juego llevó más tiempo del que el autor había previsto. Porque, por falta de experiencia, le fue difícil coordinar bien la interacción entre todos sus estados de ataque y asegurarse de que estos funcionaban suficientemente bien para que supusiesen un reto para el jugador. Lo que termino por reducir el número de niveles total del juego.

Por todas estas razones es seguro admitir que la metodología escogida para el desarrollo, la ágil, ha sido la más idónea para la realización del proyecto. Ya que, esta ha proporcionado la flexibilidad suficiente para seguir adelante con estos cambios, en vez de obligar al autor a ceñirse a una planificación estricta. Y ha permitido al final entregar una versión funcional y sólida del trabajo.

8.2. Líneas de futuro

Como resultado final se ha obtenido un juego pulido y funcional con el que el autor ha quedado satisfecho, (tras ser este su primer trabajo dirigiendo un proyecto de software entero), y espera que los futuros usuarios que lo vayan a probar también lo puedan apreciar. No obstante, como ya se sabe existen diferentes características que no se han llegado a implementar durante el proceso. Y, también varias mejoras que realizar tras haber estado probando el producto final. Estas serían:

- **Elevar el nivel de dificultad**: al probar el juego tras haberlo terminado, se notó que la dificultad general de este no era demasiado alta. Se implementó de esta manera para hacerlo accesible a cualquier usuario que quisiera probarlo. Pero, se ha visto que en algunas ocasiones también hace el juego se sienta bastante seguro y no suponga un reto real para jugadores con un poco de experiencia.

- **Agregar más enemigos**: los enemigos del juego siempre suponen un obstáculo para el jugador, y sus diferentes especializaciones hacen que se puedan incluir en todos los escenarios y secciones del juego. Aún así, se opina que podría haber más variedad de estos que aportasen más retos para el jugador y para la jugabilidad.

- **Introducir más obstáculos y elementos interactivos**: los objetos como el trampolín y las plataformas móviles son muy útiles para el diseño de niveles. Ya que, funcionan como "piezas de construcción" con las que hacer un diseño de niveles más rico y variado. Si el juego se quisiera mejorar en un futuro, agregar más elementos darían más riqueza a su jugabilidad.

- **Incluir más niveles en un futuro**: a pesar de que la longitud del juego ya es suficiente con su número total de niveles para considerarlo un producto completo, todavía se cree que se podrían agregar más para alargar un poco la experiencia del usuario.

Bibliografía

[1] El anuario del videojuego, documentación (2021) [en línea]. Aevi.org.es. [Consultado el: 4 de Marzo de 2023] Página oficial de AEVI
http://www.aevi.org.es/documentacion/el-anuario-del-videojuego/

[2] La industria del videojuego en España en 2021 (pdf) (2021) [en línea]. [Consultado el: 4 de Marzo de 2023] Anuario Aevi 2021
http://www.aevi.org.es/web/wp-content/uploads/2022/04/AEVI_Anuario_2021_Final.pdf

[3] Perspectives from the global entertainment & Media outlook 2022-2026 (2022) [en línea]. pwc.com. [Consultado el: 4 de Marzo de 2023]
https://www.pwc.com/gx/en/industries/tmt/media/outlook/outlook-perspectives.html

[4] Precio kWh hoy: precio de la luz hora a hora en España (2023) [en línea]. Selectra.es. [Consultado el: 6 de Marzo de 2023]
https://selectra.es/energia/info/que-es/precio-kwh

[5] Cuanto cuesta un servicio de programador/a PHP en Zaask (2022) [en línea]. Zaask.es. [Consultado el: 6 de Marzo de 2023]
https://www.zaask.es/cuanto-cuesta/programador-php

[6] Learn about platform game: 7 Examples of platform games (2021) [en línea]. Masterclass.com. [Consultado el: 24 de Marzo de 2023]
https://www.masterclass.com/articles/platform-game-explained

[7] Platform game [en línea]. Wikipedia. [Consultado el: 24 de Marzo de 2023]
https://en.wikipedia.org/wiki/Platform_game

[8] Blast from the past: How this generation enabled platformers to crash back into the mainstream (2020). [en línea]. Games Radar [Consultado el: 24 de Marzo de 2023]
https://www.gamesradar.com/blast-from-the-past-how-this-generation-enabled-platformers-to-crash-back-into-the-mainstream/

[9] History of platform games: 9 steps of genre evolution (2017) [en línea]. Redbull.com. [Consultado el: 25 de Marzo de 2023]
https://www.redbull.com/in-en/evolution-of-platformers

[10] Five critical moments in platform game history (2014) [en línea]. VG247 [Consultado el: 25 de Marzo de 2023]
https://www.vg247.com/five-critical-moments-in-platform-game-history

[11] First platform game in true 3D (2008) [en línea]. GuinessWorldRecords.com [Consultado el: 28 de Marzo de 2023]
https://www.guinnessworldrecords.com/world-records/first-platformer-in-true-3d

[12] Platforming games 101: running, jumping & more (2011) [en línea]. RacketBoy [Consultado el: 26 de Marzo de 2023]
https://www.racketboy.com/retro/platforming-games-101-all-you-need-to-know

[13] Keys insights into Spanish Gamers (pdf) (2022) [en línea]. [Consultado el: 30 de Marzo de 2023] https://resources.newzoo.com/hubfs/Reports/Consumer%20Insights/2022_Key_Insights_Into_Spanish_Gamers_Newzoo_Consumer_Insights_Report.pdf

[14] What genres are popular in Steam in 2022 (2022) [en línea]. [Consultado el: 31 de Marzo de 2023] https://howtomarketagame.com/2022/04/18/what-genres-are-popular-on-steam-in-2022/

[15] Top Countries/Markets by game Revenues (2022) [en línea]. Newzoo.com [Consultado el: 31 de Marzo de 2023] https://newzoo.com/resources/rankings/top-10-countries-by-game-revenues

[16] Newzoo report covering Spanish gaming market shows 83% of the nation are gamers (2022) [en línea]. pocketgamer.biz [Consultado el: 1 de Abril de 2023] https://www.pocketgamer.biz/data-and-research/79452/newzoo-report-covering-spanish-gaming-market-shows-83-of-the-nation-are-gamers/#:~:text=The%20nation%20has%20contributed%20%242.3,money%20on%20games%20this%20year.

[17] Industria del videojuego en España – Datos estadísticos (2022) [en línea]. Statista.com: [Consultado el: 2 de Abril de 2023] https://es.statista.com/temas/2851/industria-del-videojuego-en-espana/#topicOverview

[18] How many gamers are there? (New 2023 Statistics) (2023) [en línea]. ExplodingTopics.com [Consultado el: 2 de Abril de 2023] https://explodingtopics.com/blog/number-of-gamers

[19] La evolución de los videojuegos para móvil (2015) [en línea]. Socialpubli.com [Consultado el: 2 de Abril de 2023] https://socialpubli.com/es/blog/la-evolucion-de-los-videojuegos-para-moviles/

[20] Mobile Revenues Acount for more than 50% of the Global Games Market as it reaches 137.8 billion in 2018 (2018) [en línea]. Newzoo.com [Consultado el: 4 de Abril de 2023] https://newzoo.com/resources/blog/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/

[21] El videojuego en España: la industria más grande del ocio audiovisual en la que ellas juegan tanto como ellos (2022) [en línea]. Maldita.es [Consultado el: 4 de Abril de 2023] https://maldita.es/malditateexplica/20220810/videojuego-espana-datos-cifras/

[22] Entertainment and media Outlook 2021-2025 España (pdf) (2021) [en línea]. [Consultado el: 4 de Abril de 2023] https://www.pwc.es/es/publicaciones/entretenimiento-y-medios/assets/gemo-espana-2021-2025.pdf

[23] 56+ Mobile Game Statistics (2023): Revenue, Market Share, Demographics (2023) [en línea]. HeadphonesAddict.com [Consultado el: 6 de Abril de 2023] https://headphonesaddict.com/mobile-gaming-statistics/

[24] Mobile gaming industry statistics and trends for 2021(2021) [en línea]. Bussinessofapps.com [Consultado el: 6 de Abril de 2023] https://www.businessofapps.com/insights/mobile-gaming-industry-statistics-and-trends-for-2021/

[25] Casual versus Core (2000) [en línea]. gamedeveloper.com [Consultado el: 6 de Abril de 2023]

https://www.gamedeveloper.com/design/casual-versus-core

[26] ¿Qué plataforma de distribución digital de juegos es Perfecta Para Mi Juego? (2018) [en línea]. gamedeveloper.com [Consultado el: 12 de Abril de 2023]
https://gamedevelopment.tutsplus.com/es/articles/which-digital-game-distribution-platform-is-the-right-for-my-game--cms-30289

[27] Report: Steam's 30% cut is actually the industry standard (2020) [en línea]. ign.com [Consultado el: 12 de Abril de 2023]
https://www.ign.com/articles/2019/10/07/report-steams-30-cut-is-actually-the-industry-standard

[28] Valve's new Steam agreement gives more money to game developers (2018) [en línea]. theverge.com [Consultado el: 12 de Abril de 2023]
https://www.theverge.com/2018/11/30/18120577/valve-steam-game-marketplace-revenue-split-new-rules-competition

[29] Understanding Google Play's service fee (2023) [en línea]. supportgoogle.com [Consultado el: 12 de Abril de 2023]
https://support.google.com/googleplay/android-developer/answer/11131145?hl=en#zippy=%2Care-all-developers-subject-to-a-service-fee

[30] GameMaker Features (2023) [en línea]. gamemaker.io [Consultado: 9 de Mayo de 2023]
https://gamemaker.io/en/gamemaker/features

[31] Unity User Manual 2021.3(LTS) (2023) [en línea]. unity3d.com [Consultado: 9 de Mayo de 2023]
https://docs.unity3d.com/Manual/index.html

[32] Godot Engine 4.0 documentation in English (2023) [en línea]. godotegine.org [Consultado el: 9 de Mayo de 2023]
https://docs.godotengine.org/en/stable/index.html

[33] Godot vs Unity: Which one is better? (2023) [en línea]. Gitnux.com [Consultado el: 10 de Mayo de 2023]
https://blog.gitnux.com/comparison/godot-vs-unity/#:~:text=Unity%20is%20better%20suited%20for,would%20be%20the%20best%20choice

[34] Unity vs GameMaker: What to choose for Game Development? (2022) [en línea]. ilogos.biz [Consultado el: 10 de Mayo de 2023]
https://ilogos.biz/unity-vs-gamemaker-what-to-choose-for-game-development/

[35] Pick a tier and make a game (2023) [en línea]. gamemaker.io [Consultado el: 14 de Mayo de 2023]
https://gamemaker.io/en/get

[36] System requirements for Unity 2021 LTS (2023) [en línea]. unity3d.com [Consultado el: 16 de Mayo de 2023]
https://docs.unity3d.com/Manual/system-requirements.html

Anexos

Anexo A: Glosario

- **Agile**. Proceso iterativo de desarrollo de software centrado en proporcionar una planificación y desarrollo más flexible que el proceso Waterfall.
- **Apple**. Compañía de tecnología multinacional americana.
- **Appstore**. Mercado en línea de aplicaciones desarrollado y mantenido por Apple para aplicaciones móviles del sistema operativo iOS.
- **Assets**. Término utilizado para cualquier elemento empleado en un videojuego.
- **8/16/32-Bits**. Adjetivo utilizado para describir la medida de los bloques de datos que podía procesar el dispositivo dependiendo de su hardware y software.
- **Casual Gamers**. Usuario jugador quien disfruta de cualquier género de videojuegos sin invertir una cantidad significativa de tiempo o dinero en ellos, jugando de manera irregular y espontánea.
- **Checkpoints**. Localización especial en el escenario del juego donde el sistema almacena la posición del jugador para establecerlo como punto de reaparición actual.
- **Consola de juegos doméstica**. Dispositivo informático diseñado especialmente para jugar a videojuegos.
- **Core gamers**. Usuario jugador con un rango de intereses amplio y que demuestra más entusiasmo por la industria del juego.
- **Cuartil inferior**. Valor estadístico de un conjunto ordenado tal que una cuarta parte de las observaciones (25%) son inferiores o iguales a él, y el resto (75%) es superior o igual.
- **Cuartil superior**. Valor estadístico de un conjunto ordenado tal que las tres cuartas partes de las observaciones (75%) son inferiores o iguales a él, y el resto (25%) es superior o igual.
- **DFS (Depth-First Search)**. Algoritmo para atravesar o buscar estructuras de datos de árboles o grafos.
- **Expansiones**. Contenido adicional descargable de un videojuego que ya ha sido publicado y distribuido.
- **Game engine (motor de juegos)**. Marco de software diseñado principalmente para el desarrollo de videojuegos que generalmente incluye bibliotecas y programas de apoyo.
- **Gamedev**. Término inglés para referirse al acto de crear videojuegos.
- **GameMaker**. Motor de juegos multiplataforma especializado en la creación de juegos en 2D empleando lenguajes como C++ y GDscript.
- **Gameplay**. Forma en la que se juega un juego.
- **Generación de consolas**. Clasificación de las consolas de juegos según sus especificaciones de Software o Hardware.
- **Generación procedural**. Método de creación de datos (como terreno u objetos) por medio del uso de algoritmos y aleatoriedad en vez de manualmente.
- **Gráfica de barras**. Estilo de gráfico financiero que representa datos numéricos con barras rectangulares cuyas alturas son proporcionales a los valores representados.
- **Gráfica de líneas**. Estilo de gráfico financiero que muestran una serie como un conjunto de puntos conectados mediante una sola línea en un informe paginado.
- **Gráfica de velas**. Estilo de gráfico financiero empleado para describir los movimientos de precios de un valor, derivado o divisa.
- **Godot**. Motor de juegos que permite crear juegos en 2D y 3D empleando lenguajes como C++ y GDscript.

- **Google Play**. Servicio en línea de distribución de videojuegos desarrollado y dirigido por Google.
- **IA**. Inteligencia demostrada por las máquinas.
- **Indie**. Referido a los productos con un desarrollo independiente o no afiliado con compañías medianas o grandes.
- **Iphone**. Línea de Smartphones producidos por Apple.
- **Items**. Objeto en un juego que puede ser recogido y/o utilizado por un jugador.
- **Loopings**. Sección del escenario en la que el jugador realiza un giro de 360 grados sobre sí mismo.
- **Mecánica**. Cualquier acción que realice el jugador que modifique la posición o el estado de los objetos o el entorno en una partida.
- **Mediana**. Valor de la variable de valor de posición central en un conjunto de datos ordenados.
- **MSI**. Corporación internacional de tecnología e información originaria de Taiwan.
- **Newbies**. Término utilizado para referirse a individuos recién llegados y/o sin experiencia.
- **Online**. Servicio o actividad disponible a través del uso de Internet.
- **Parallax**. Efecto óptico basado en el desplazamiento aparente de un objeto visto desde dos líneas de visión diferentes que no están en línea recta con el objeto.
- **PC (Personal Computer)**. Microcomputadora cuyo tamaño, capacidades y precio la hacen factible para su uso individual.
- **Plataformas de juego**. Sistema informático en el que es posible jugar a videojuegos.
- **Pixelart**. Forma de arte digital donde las imágenes son editadas al nivel de los píxeles.
- **Reddit**. Sitio web estadounidense de agregación de noticias sociales, calificación de contenido y discusión entre usuarios.
- **Script**. Los Scripts son secciones de código que se asignan a los diferentes objetos del juego para indicar como estos deben comportarse.
- **Smartphone**. Dispositivo informático portátil que combina las funciones de teléfono móvil y computadora.
- **Sprite**. Objeto gráfico en dos dimensiones que se emplea en los videojuegos para representar un objeto o personaje.
- **Steam**. Servicio en línea de distribución digital de juegos desarrollado por la compañía Valve.
- **Testroom**. Escena separada del juego original en la que se realizarán las pruebas de determinadas mecánicas antes de incluirlas en la versión oficial.
- **Tiles**. Elemento de la construcción del nivel que representa una superficie u obstáculo del escenario.
- **Tileset**. Conjunto completo de tiles para construir el escenario de un juego de plataformas.
- **Tráiler**. Anuncio comercial sobre un producto audiovisual.
- **Triple A.** Clasificación informal que se emplea para juegos producidos y distribuidos por una compañía mediana o grande, con presupuestos para el desarrollo y marketing más elevados que en otros juegos.
- **Twitter**. Red social famosa y estadounidense.
- **Underdog**. Término angloparlante usado para referirse a las víctimas de injusticia y/o marginales con muchas expectativas de que va a perder.
- **Unity**. Motor de juegos multiplataforma desarrollado por Unity Technologies.
- **Waterfall**. Proceso de desarrollo secuencial que “fluye” como una cascada a través de todas las fases del proyecto (cada fase debe terminar por completo para que pueda comenzar la siguiente).
- **Youtube**. red social estadounidense usada como plataforma para compartir vídeos en línea.