OverColor (TFP)

Autor: José María Morales Miñarro

Tutor: Jordi Duch Gavaldà

Profesor: Joan Arnedo Moreno

Máster universitario de Diseño y Programación de Videojuegos

Diseño de experiencias de juego

Fecha de entrega

06/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*OverColor*
Nombre del autor:	*José María Morales Miñarro*
Nombre del colaborador/a docente:	Jordi Duch Gavaldà
Nombre del PRA:	Joan Arnedo Moreno
Fecha de entrega (mm/aaaa):	*06/2022*
Titulación o programa:	*Máster universitario de Diseño y Programación de Videojuego*
Área del Trabajo Final:	*Diseño de experiencias de juego*
Idioma del trabajo:	*Español*
Palabras clave	*Videojuego, móvil, roguelite.*

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

El presente Trabajo de Fin de Máster trata de explicar y detallar el proceso realizado para la construcción de un videojuego dirigido hacia un público más casual desarrollado con los dispositivos móviles como principal plataforma en mente, haciendo uso de múltiples de las disciplinas aprendidas durante el periodo docente, construido principalmente con el motor Unity.

Como producto final, se obtendrá un videojuego con una curva de aprendizaje baja, en el que el jugador tendrá que vencer a hordas de enemigos haciendo uso de tres botones de colores (rojo, amarillo y azul), de forma que un enemigo será derrotado si se pulsa el botón de su color (por ejemplo, azul), si bien existirán colores más complejos (como el verde, que requerirán de que se use tanto el botón azul como el amarillo). Los enemigos vendrán por distintas líneas, por lo que el jugador deberá pensar bien su estrategia y el orden en el que pulsa los botones, creando una experiencia que mezcla la estrategia, los puzles y la acción.

El juego cuenta además con elementos típicos del género *roguelike*, como mejoras que se irán obteniendo durante la sesión de juego que serán perdidas al morir, si bien también existen otras recompensas permanentes para el jugador.

Abstract (in English, 250 words or less):

This Master's Thesis tries to explain and detail the process carried out for the construction of a video game aimed at a more casual audience developed with mobile devices as the main platform in mind, making use of multiple disciplines learned during the course, built primarily with the Unity engine.

As a final product, a video game with a low learning curve will be obtained, in which the player will have to defeat hordes of enemies using three coloured buttons (red, yellow and blue), so that an enemy will be defeated if the button of its colour (for example, blue) is pressed, although there will be more complex colours (such as green, which will require both the blue and yellow buttons to be used). The enemies will come from different lines, so the player must think carefully about their strategy and the order in which they press the buttons, creating an experience that mixes strategy, puzzles and action.

The game also incorporates typical elements of the roguelike genre, such as improvements that will be obtained during the game session that will be lost when you die, although there are also other permanent rewards for the player.

Índice

Figuras y tablas

Índice de figuras

1. Introducción

1.1. Introducción/Prefacio

No es un secreto que los videojuegos son un medio de entretenimiento en auge en nuestra sociedad actual, viniendo en todo tipo de formas y colores según el público al que tratan de entretener y agradar.

El presente Trabajo de Fin de Máster consiste en el diseño y construcción de un videojuego para dispositivos móviles centrado en ofrecer una experiencia simple y entretenida para el jugador, poniendo su punto de mira en el público más casual y dirigiendo el desarrollo hacia los dispositivos multitáctiles.

¿Por qué dispositivos móviles? Aunque a muchos pueda sorprenderle, el mercado móvil es el sector del videojuego con mayores beneficios, principalmente por un beneficio clave: la accesibilidad. En la sociedad actual, no todo el mundo tiene interés o recursos para comprarse una videoconsola, o incluso en muchos casos una computadora; sin embargo, una enorme cantidad de personas cuentan con un dispositivo móvil, el cual los acompaña a todas partes. Un usuario no demasiado interesado en los videojuegos probablemente no comprará una consola cuando vea un juego que le llame la atención, pero si tan solo tiene que dar un par de toques en la pantalla de su celular, la situación es diferente. Tal y como es sabido, el mercado del videojuego para dispositivos móviles es muy exitoso, generando una enorme cantidad de beneficios (ver Figura 1).

Sin embargo, la razón por la que se ha escogido desarrollar el proyecto con esta plataforma en mente no es la obtención de beneficios, sino específicamente el sector de la población al que va dirigido. Considerando los recursos temporales limitados para este proyecto, se decidió desde las etapas más tempranas que la mejor ruta de acción sería centrarse en desarrollar un juego simple y entretenido, con un foco de jugabilidad que fuese divertido y atractivo, pero sin profundizar en sistemas muy profundos ni escenarios/artes demasiado complejos, ya que tan solo llevaría a planificaciones no realistas y a un proyecto final a medias. Por ello, considerando que el juego ha de ser simple, es mejor dirigirlo a un público casual que pueda disfrutarlo mientras espera en el autobús, o cuando se aburre en una sala

de espera, en lugar de al jugador hardcore que espera una experiencia más profunda, y que probablemente no se vea atraído por el producto.

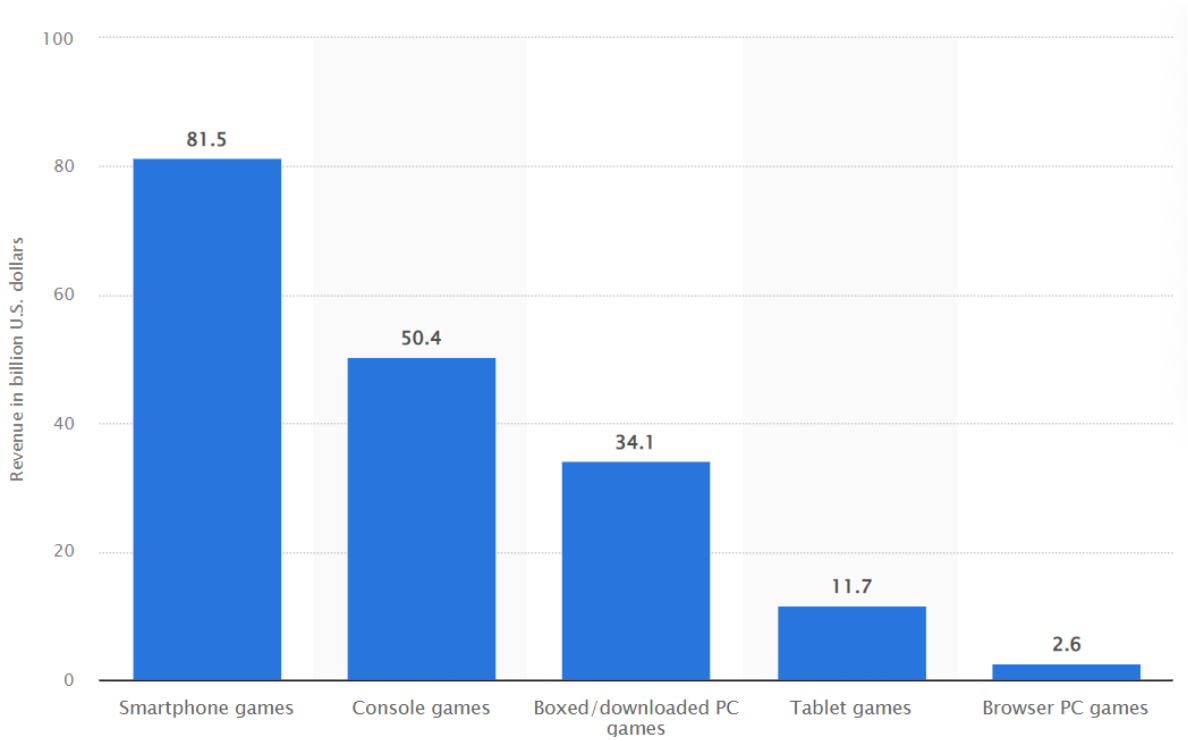


Figura 1. Beneficios obtenidos en el mercado del videojuego para diferentes plataformas en 2021 [1]

1.2. Descripción/Definición

El proyecto consistirá en el desarrollo completo de un videojuego desde cero, realizando las distintas fases de conceptualización, diseño, desarrollo y testeo. En un punto original, tan solo contamos con unas ideas básicas sobre los recursos temporales disponibles para el desarrollo y los conocimientos tratados durante los estudios, de forma que todo lo demás deberá ser desarrollado durante la realización del proyecto.

Tal y como se especificó en apartados anteriores, el videojuego desarrollado como Trabajo de Fin de Máster está dirigido al mercado de los sistemas multitáctiles, principalmente dispositivos móviles, el cual es especialmente efectivo para llegar a captar la atención de un público más casual que busque en los videojuegos un entretenimiento accesible y ligero, en lugar de la experiencia más hardcore o profunda que puedan ofrecer otras plataformas. La razón por la que tratamos de apelar a este sector de la población es debido a que con los recursos disponibles se estima que no es viable realizar un videojuego complejo, y que en su lugar es mucho más factible trabajar en algo más sencillo y centrado en sesiones de juego

más cortas y casuales en lugar de largas horas de *gameplay*, lo que encaja perfectamente con el mercado de videojuegos para dispositivos móviles.

El objetivo es la obtención de un videojuego que sea disfrutable, y que sea fácil de simplemente descargar y comenzar a jugar, sin controles muy complejos o conocimientos previos necesarios; idealmente, personas de cualquier rango de edades podrían disfrutar de él, y no requerirá de gran potencia hardware para funcionar, facilitando el acceso a un elenco de jugadores más amplio.

El videojuego desarrollado como producto final contará con la siguiente jugabilidad (boceto disponible en la Figura 2):

- En la pantalla, habrá tres líneas diferentes: la superior, la media, y la inferior. Desde cada una de esas líneas vendrán diversos enemigos, los cuales provocarán la derrota cuando alcancen al jugador.
- Por otro lado, el jugador contará con un avatar, el cual estará colocado en una de las tres líneas, si bien el propio jugador podrá mover libremente a su avatar entre cualquiera de las líneas haciendo uso de la pantalla de su dispositivo.
- El jugador contará también con tres botones de colores: uno azul, uno rojo y uno amarillo. Cuando los pulse, el avatar realizará un ataque asociado a dicho color en la línea en la que se encuentre.
- Los enemigos que vienen por las líneas tendrán un color asociado; si el jugador los golpea con un ataque de dicho color, el enemigo quedará derrotado. Algunos enemigos tendrán mecánicas o colores más complejos, que requerirán de varios golpes para derrotarlos.
- Conforme el jugador progrese y venza enemigos, recibirá recompensas aleatorias, las cuales le ofrecerán ventajas que le harán más fuerte y le permitirán enfrentarse a enemigos más fuertes. Al morir, esas ventajas se perderán.
- Existirá también una progresión global la cual no se resetea en el caso de derrota, ofreciendo pequeñas mejoras permanentes de las que el jugador siempre podrá disfrutar, de forma que, aun cuando se pierda, se sienta que la partida jugada no ha sido en vano.

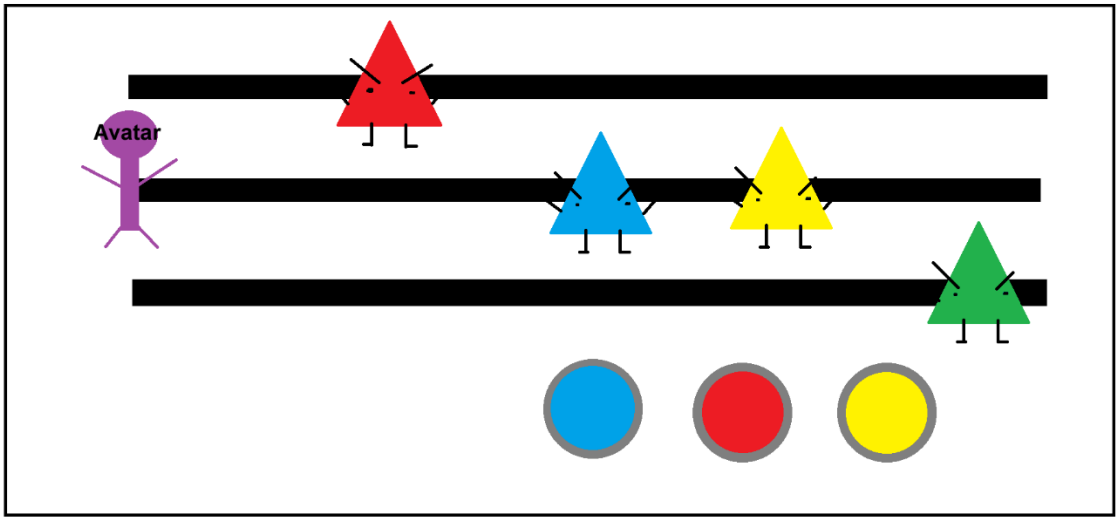


Figura 2. Boceto inicial de la pantalla de juego en OverColor

En base a lo explicado, se puede deducir que gran parte de la experiencia de juego viene condicionada por las mejoras obtenibles y los sistemas de progresión, así como los enemigos a derrotar.

En OverColor, los enemigos por lo general serán simples, ya que el desafío viene más arraigado a “vencer hordas de muchos enemigos” a “vencer a un único enemigo complejo”; sin embargo, existirán enemigos con algunas mecánicas ligeramente variadas, como enemigos que saltan de una línea a otra, o enemigos que explotan al morir.
En cuanto a las recompensas a obtener durante las partidas, se planea que haya diversas opciones según el estilo de juego. Por un lado, existirán mejoras que aumentan las estadísticas del jugador, permitiéndole disparar más rápido o recibir más impactos antes de perder; por otro lado, tendremos mejoras que añadirán mecánicas nuevas que cambiarán la forma de jugar, como máquinas que destruyan a los enemigos en pantalla cada varios segundos, clones del jugador que imitan sus acciones, o incluso mejoras que ofrezcan un efecto potente a costa de hacer a los enemigos también más peligrosos.

Para finalizar, el sistema de progresión permanente estará centrado en la obtención de tres divisas (una roja, otra azul y otra amarilla) para mejorar diversos establecimientos con el objetivo de obtener incrementos permanentes en las estadísticas del jugador, como más puntos de vida o mayor probabilidad de encontrar objetos más poderosos durante las partidas. Estas divisas se obtendrán al derrotar a enemigos, de forma que, aun cuando se pierda durante una partida, el jugador siempre se llevará algo y estará más cerca de vencer haciéndose más fuerte.

1.3. Objetivos generales

A continuación, se explicarán cuáles son los objetivos que trata de abordar el presente proyecto, tanto desde la perspectiva del videojuego como producto, como desde la perspectiva propia con respecto a los intereses asociados a trabajar este proceso de desarrollo.

1.3.1. Objetivos principales

Se comenzará explicando los objetivos de la aplicación desde un punto de vista de "videojuego como producto/servicio":

- Ofrecer una experiencia divertida y entretenida, donde los jugadores estén activamente disfrutando durante el tiempo que inviertan jugando.
- Que sea fácil de jugar, de forma que cualquiera, aun cuando no tenga experiencia pasada con los videojuegos, pueda cogerlo y empezar a disfrutar, sin sentir frustración o aversión.
- Presentar un videojuego que sea fácil de acceder, de forma que prácticamente cualquiera pueda probarlo y jugarlo, sin necesitar de una infraestructura muy específica, costosa, o alejada de lo que ya tuviese disponible.

Por otro lado, se comentarán los objetivos correspondientes al punto de vista del propio del desarrollador, y sus intereses a la hora del desarrollo del proyecto:

- Aprender e investigar sobre el desarrollo en plataformas multitáctiles y dispositivos móviles, ya que son un mercado muy potente y con un futuro aún más prometedor.
- Tener un proyecto que se pueda mostrar como parte de un portfolio.
- Desarrollar un producto que pueda ajustarse a las limitaciones de recursos temporales y de personal, sin por ello ofrecer una experiencia incompleta o insatisfactoria.

1.3.2. Objetivos secundarios

También se pueden destacar algunos objetivos adicionales, que, si bien no son el centro de atención, también afectan a la dirección y desarrollo del proyecto:

- Conseguir más experiencia en el desarrollo de videojuegos con el motor Unity.
- Poner en práctica los conocimientos obtenidos durante el periodo lectivo.

1.4. Metodología y proceso de trabajo

Para el desarrollo del proyecto, se ha considerado construir el juego desde cero, en lugar de trabajar sobre otro existente; en cierta medida, se está desarrollando un videojuego como un producto el cual (en teoría) podría ser comercializado, lo que implica que, si bien se puede buscar inspiración o referencias en otras obras, debe respetar las leyes de Copyright y ser algo propio y nuevo por sí mismo.

En cuanto a la metodología, se ha considerado que, teniendo en cuenta los límites temporales del proyecto, la metodología de gestión de proyectos o bien debía ser cascada, o bien una metodología ágil.

Por un lado, cascada es una metodología que funciona bien para proyectos pequeños, donde predecir las distintas fases no es muy difícil, y donde la corta duración de los mismos hace que las principales carencias del modelo en cascada (como la falta de flexibilidad o adaptación a situaciones inesperadas) no se vean muy potenciadas.

Por otro lado, las metodologías ágiles nos ofrecen una enorme ventaja: nos permiten ir iterando durante el proceso de desarrollo del proyecto, de forma que podemos adaptarnos mejor a cualquier imprevisto, tanto positivo como negativo, poniendo menos peso en la necesidad de una rigurosa predicción temporal, lo cual puede llegar a ser complicado y requiere de gran experiencia dentro del sector de la gestión de proyectos.

Por tanto, se decidió que la metodología a emplear sería una metodología ágil. De esta forma, en un principio se centrará el esfuerzo en pulir el núcleo de la jugabilidad y las mecánicas fundamentales, y una vez sea sólido y funcional, se añadirán nuevas mecánicas una a una, dentro de un conjunto de posibilidades. De esta forma, en caso de que acabase faltando tiempo, se podría simplemente reducir el número de mecánicas o recompensas extras a desarrollar fuera de lo estimado, así como si el proceso fuese más rápido del esperado, podrían incluirse nuevas funcionalidades o mejoras.

Sin embargo, la mayoría de las metodologías ágiles más populares están muy enfocadas en la coordinación del trabajo en equipo entre los miembros del equipo de desarrollo, mas el presente proyecto ha de ser abordado de forma individual. Por tanto, se trabajará con una versión de Kanban más simple que dejará de lado la coordinación grupal y se centrará en la clara clasificación de las distintas tareas a desarrollar según su etapa en el ciclo de vida de desarrollo.

1.5. Planificación

A continuación, describiremos la planificación temporal inicial que se estima para la realización del proyecto, teniendo en cuenta que al usar metodologías ágiles las fechas y actividades pueden verse modificadas según evolucione el proyecto.

En primer lugar, detallaremos cuáles son las actividades que se planean realizar, para después estructurarlas en el marco temporal correspondiente:

- **1. Planificación inicial y conceptualización de la propuesta**: se creará una idea inicial de proyecto, con sus actividades y objetivos, y se planificará una versión inicial para su desarrollo.
- **2. Estudio de estado del arte y mercado**: investigación sobre otros productos similares en el mercado en la actualidad.
    - **2.1. Investigación sobre otros juegos móviles, su popularidad y género**: proceso dedicado a investigar sobre el mercado móvil y los juegos de renombre del sector.
    - **2.2. Investigación de otros juegos del género *roguelite* o *roguelike***: proceso dedicado a estudiar los sistemas y técnicas que caracterizan a los juegos de estos géneros.
    - **2.3. Investigación sobre motores de desarrollo y software de dibujo**: proceso dedicado a investigar sobre alternativas y potenciales opciones de entorno y herramientas de desarrollo a utilizar.
- **3. Construcción de prototipo con mecánicas fundamentales**: desarrollo de un primer proyecto funcional que presente cómo será el juego.
    - **3.1. Implementación de la pantalla con tres líneas**: construcción de la pantalla principal de juego, con las tres líneas por las que vendrán enemigos.

    - **3.2. Implementación de enemigos básicos y botones**: construcción de los enemigos más básicos de forma funcional y los botones para vencerlos.
    - **3.3. Implementación básica del sistema de progresión**: construcción de una versión básica del sistema de avance permanente.
    - **3.4. Implementación básica de las mejoras obtenibles**: construcción de una versión básica de las mejoras obtenibles durante una partida.
    - **3.5. Implementación básica de otras pantallas (menús, fin de partida…)**: construcción de una versión preliminar de las otras pantallas.
- **4. Pulir mecánicas fundamentales**: afinamiento del prototipo inicial con el objetivo de que el foco de la jugabilidad sea fluido y no tenga bugs mayores.
- **5. Implementación de una versión base del juego**: creación de un segmento del juego más completo que muestre un fragmento del juego en una fase más avanzada.
    - **5.1. Implementación completa de los enemigos**: implementación de la versión final y sólida de los enemigos del juego.
    - **5.2. Implementación completa de las mejoras**: implementación de la versión final mecánicamente de las mejoras obtenibles.
    - **5.3. Implementación completa del sistema de progresión**: implementación definitiva de forma mecánica del sistema de progresión permanente del juego.
- **6. Balanceo de los sistemas del juego**: revisión de los distintos elementos del juego como velocidad de aparición de enemigos, nivel de poder de las ventajas y del sistema de progresión, y correspondiente ajuste en pos de la mejor experiencia posible.
- **7. Desarrollo del apartado artístico**: construcción o adquisición de recursos artísticos tanto visuales como sonoros para el producto final.
- **8. Redacción de la memoria**: construcción del documento trabajado durante el desarrollo del proyecto con toda la información del mismo.

Cabe destacar que, aun cuando las fechas de las actividades son variables, los entregables en forma de PEC en la planificación han de mantenerse en las fechas establecidas con el propósito de que los tutores puedan hacer un seguimiento correcto del proceso de desarrollo.

En la figura 3 se podrá observar una tabla con las fechas estimadas acompañada de su correspondiente diagrama de Gantt, ambas desarrolladas con el software Microsoft Project; en la figura 4, se tendrá el diagrama de PERT mostrando la interrelación de las actividades y las dependencias entre ellas.

Nombre de tarea	Duración	Comienzo	Fin
Planificación inicial y conceptualización de la propuesta	14 días	16/02/22	06/03/22
Entrega PEC 1	0 días	06/03/22	06/03/22
Estudio de estado del arte y mercado	**11 días?**	**07/03/22**	**20/03/22**
Investigación sobre otros juegos móviles, su popularidad y	6 días	07/03/22	14/03/22
Investigación de otros juegos del género roguelite o roguelike	4 días	15/03/22	18/03/22
Investigación de motores de desarrollo y software de dibujo	2 días	19/03/22	20/03/22
Construcción de prototipo con mecánicas fundamentales	**11 días?**	**21/03/22**	**03/04/22**
Implementación de la pantalla con tres líneas	2 días	21/03/22	22/03/22
Implementación de enemigos básicos y botones	4 días	23/03/22	26/03/22
Implementación básica del sistema de progresión	2 días	26/03/22	28/03/22
Implementación básica de las mejoras obtenibles	4 días	29/03/22	01/04/22
Implementación básica de otras pantallas (menús, fin	2 días	02/04/22	03/04/22
Entrega PEC 2	0 días	03/04/22	03/04/22
Pulir mecánicas fundamentales	11 días	04/04/22	18/04/22
Implementación de una versión base del juego	**15 días?**	**19/04/22**	**08/05/22**
Implementación completa de los enemigos	6 días	19/04/22	26/04/22
Implementación completa de las	5 días	27/04/22	03/05/22
Implementación completa del sistema de progresión	4 días	04/05/22	08/05/22
Balanceo de los sistemas del juego	11 días	09/05/22	22/05/22
Entrega PEC 3	0 días	08/05/22	08/05/22
Desarrollo del apartado artístico	11 días	23/05/22	05/06/22
Redacción de la memoria	79 días	16/02/22	05/06/22
Entrega PEC 4	0 días	05/06/22	05/06/22

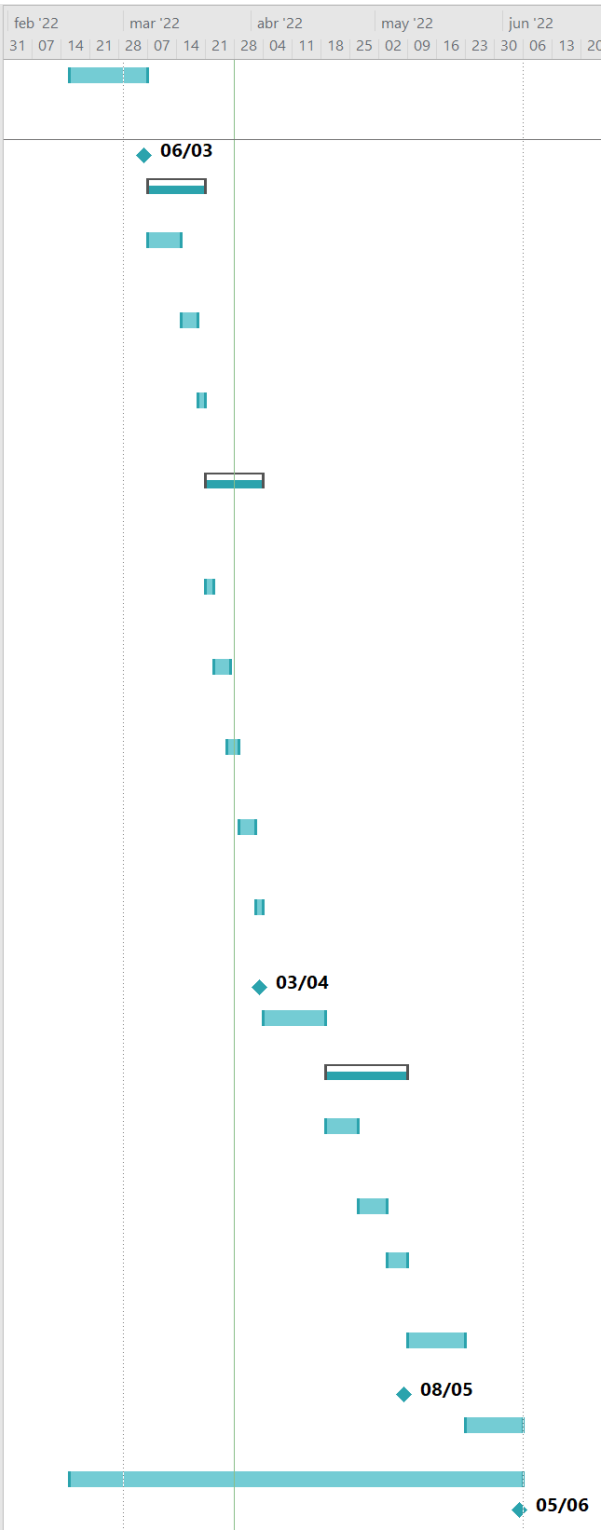


Figura 3. Tabla con flechas y diagrama de Gantt del proyecto

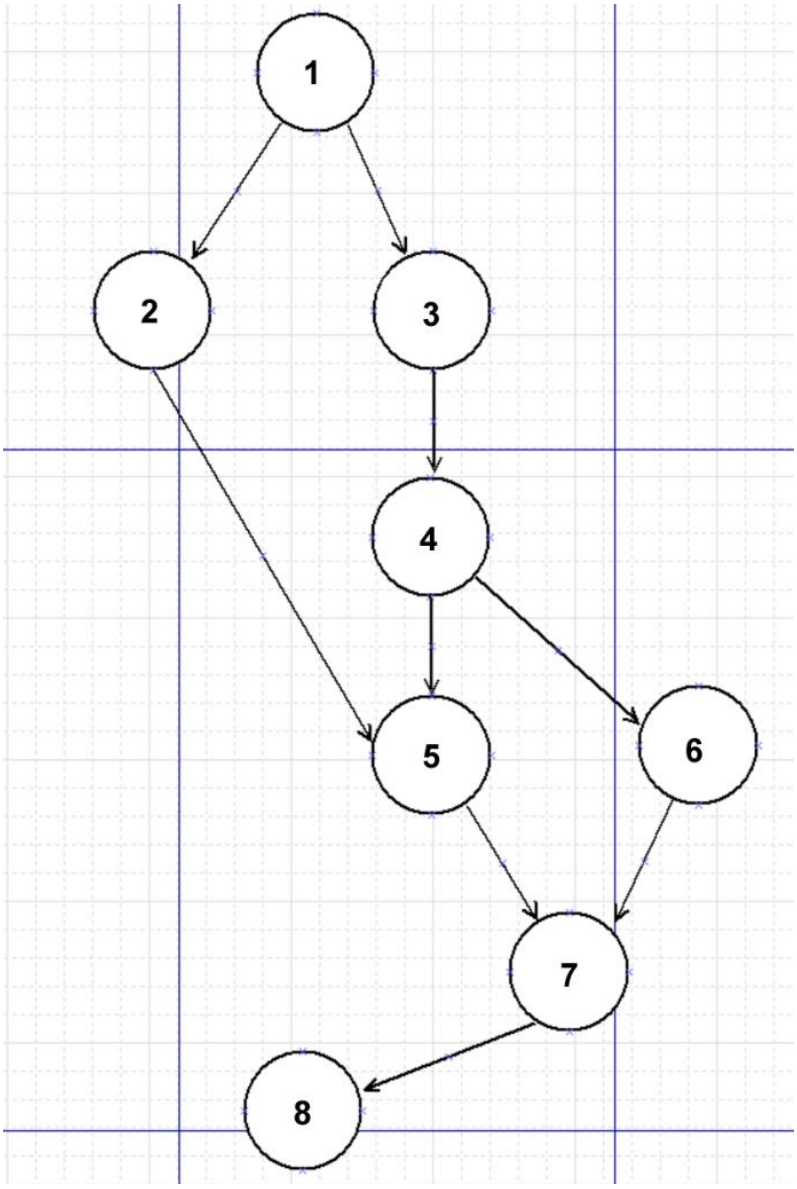


Figura 4. Diagrama de PERT de las actividades del proyecto

1.6. Estructura del resto del documento

A continuación, se explicarán los puntos que conformarán el presente documento con un ligero resumen de su contenido:

1. **Introducción**: en este apartado se explicarán las bases del proyecto, la idea a implementar y el porqué, así como los objetivos a conseguir y la planificación temporal.
2. **Estado** del arte: este punto estará centrado en profundizar sobre las diferencias entre *roguelites* y *roguelikes*, así como una presentación del estado actual del mercado tanto de videojuegos móviles como del mercado *roguelite*. Se hablará también sobre los motores y entornos más prometedores de la actualidad y su viabilidad.
3. **Propuesta**: en este apartado se solidificará la propuesta del videojuego a desarrollar, explicando sus bases, sus mecánicas, su historia, los elementos del *gameplay* y los sistemas de progresión.
4. **Diseño**: en este punto se justificarán las decisiones de diseño tomadas, listando las herramientas utilizadas para construir el videojuego, así como se presentará un inventario de los *assets* empleados. Se presentará también la interfaz de usuario.
5. **Implementación**: este apartado estará centrado tanto en listar los scripts que respaldan la lógica del juego como en guiar al usuario en el proceso de instalación.
6. **Demostración**: este punto explicará las pruebas realizadas para comprobar el correcto funcionamiento de OverColor.
7. **Conclusiones y líneas de futuro:** en el apartado final se hablará sobre las implicaciones de la realización del proyecto, sobre el cumplimiento de los objetivos iniciales y de la planificación, así como se presentarán propuestas para mejoras y expansiones a implementar en el futuro para el videojuego.

2. Estado del arte

En este apartado se presentará el estado del arte del ámbito del videojuego a construir, profundizando en el proceso en el género del juego, referentes y éxitos del mercado, así como se hablará de las tecnologías a utilizar y la audiencia objetivo del producto en cuestión.

2.1. Revisión del género

Tal y como se presentó en la introducción, nuestro juego mezcla ligeramente varios géneros. A efectos prácticos, OverColor es un juego de acción del tipo arcade, en donde el peso reside en una jugabilidad de repetición, así como hay ligeros toques del género puzles, ya que el jugador deberá combinar correctamente los colores para tener éxito. Sin embargo, eso no es todo; el corazón de OverColor ha de ser catalogado bajo el género *roguelite*.

2.1.1. Roguelite vs Roguelike

Hace unos instantes se hizo referencia al término *roguelite*, pero antes de continuar es conveniente aclarar el significado de dicho término que se utilizará en el presente proyecto para describir a OverColor; especialmente, en contraposición al género *roguelike*, el cual está fuertemente vinculado al *roguelite*, pero con diferencias fundamentales.

El origen de ambos géneros viene del videojuego Rogue (1980) [2] en el que el jugador debía planificar su aventura por una mazmorra llena de diversas salas y peligros generados de forma procedural, con un sistema de juego RPG por turnos, y un gran grado de dificultad. No obstante, lo más característico de esta obra fue su sistema de muerte permanente o *permadeath*: cuando el jugador caía derrotado por cualquiera de los peligros de las mazmorras, tendría que empezar desde 0, en una nueva mazmorra completamente nueva generada aleatoriamente, y sin tener en posesión ninguna de las mejoras o ventajas que pudo haber obtenido en su intento anterior. A efectos prácticos, toda la progresión desaparecía completamente, salvando, claro está, la propia habilidad del jugador y experiencia real obtenida.

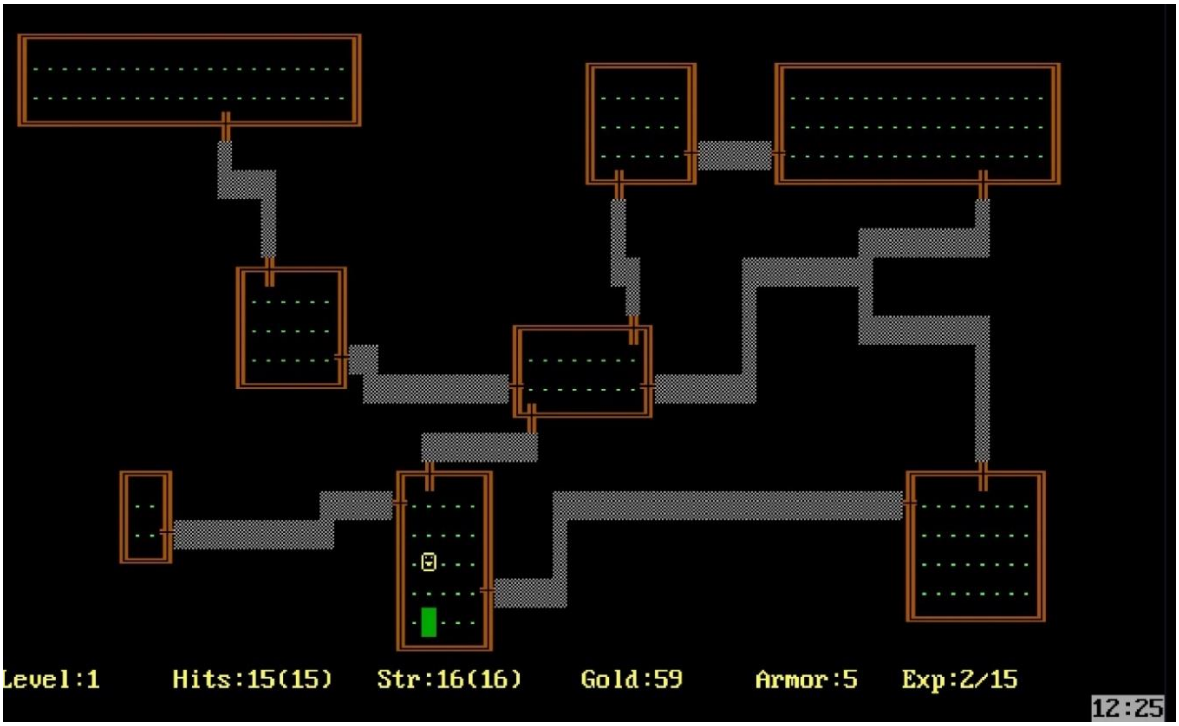


Figura 5. Rogue (1980), el primer roguelike

En los años siguientes al nacimiento de Rogue, aparecerían otras obras con ese mismo esqueleto y mecánica de *permadeath*, formalizando, en cierto modo, un nuevo género de nicho: los *roguelike*. Viajando más a los tiempos presentes, el género volvería a redescubrirse, con todo tipo de obras que se abrirían paso al mercado más general y *mainstream*, heredando parte de la esencia de Rogue. Sin embargo, no todas se mantendría completamente fieles a la estructura de juego seguida por Rogue y sus predecesores; muchas de estas obras no harían uso de combates RPG por turnos, otras mantendrían parte del progreso entre partidas o *runs*. Encontrando dificultad para saber hasta qué punto un juego es un *roguelike* o no, empezó a acuñarse el término *roguelite*.

En internet, inundando los foros, comenzaron a aparecer todo tipo de debates y opiniones sobre qué características eran necesarias para poder llamarse “un roguelike”, y discusiones sobre si X juego debía ser catalogado como *roguelike* o *roguelite* eran algo común. En el artículo “The Roguelike Debate – Roguelikes vs Roguelites” (2019) [3] de Josh Bycer, se trata de dar una respuesta a dicha cuestión, especialmente desde el punto de vista de los objetivos que se deberían alcanzar como desarrollador de videojuegos en caso de tener intención de crear juegos de un tipo u otro. El criterio final que propone Josh Bycer es el siguiente: el peso de un juego *roguelike* reside en “el camino” en sí, en las decisiones tomadas durante la *run*; por otro lado, el peso de un *roguelite* está en “el final” de la *run*, en las mejoras y progresión

obtenidas que podrán ser trasladadas al siguiente intento. A efectos prácticos, un *roguelike* es un *roguelike* porque tiene *permadeath*, obligando al jugador a mejorar su habilidad si quiere obtener resultados, mientras que un *roguelite* no obliga a empezar de cero al jugador, dándole más importancia a que el jugador avance en los sistemas de progresión del propio juego para hacerse más fuerte de forma virtual. Bajo esta descripción, se hizo la catalogación de OverColor como *roguelite*.

2.1.2. Roguelite en el mercado actual

En base a la descripción y comparación del apartado anterior, se determinará que OverColor trata de ser un videojuego amparado en el género *roguelite*: durante el juego, parte de las mejoras se guardarán de forma permanente, haciendo que las estadísticas y números del jugador, como pueden ser los puntos de vida, sean más altas al inicio de una *run* para un jugador con varias horas de juego que para uno que acaba de empezar.

Por otro lado, OverColor tampoco comparte otros elementos del juego Rogue como son el combate por turnos, mas tenemos numerosos ejemplos de juegos, catalogados tanto como *roguelike* como *roguelite*, que también cambia la fórmula inicial.

Uno de los ejemplos más llamativos de juegos exitosos del género *roguelite* es, sin lugar a duda, el videojuego Hades (2018) [4], de Supergiant Games. En Hades, el jugador deberá controlar a Zagreo, el hijo de Hades, en su intento de escapar del infierno, recibiendo en el camino diversas bendiciones impuestas por los dioses del Olimpo. Al finalizar una *run*, ya sea tras conseguir escapar exitosamente, o tras ser derrotado, el jugador perderá las bendiciones de los dioses, pero mantendrá algunas divisas, como la Oscuridad. Dichas divisas podrán ser intercambiadas por mejoras permanentes para Zagreo, como más vida o más munición, las cuales se conservarán para siempre en el archivo de guardado.

Figura 6. Hades (2018) es uno de los roguelite más populares del mercado

Sin embargo, se podría formular la pregunta: ¿cuál es la razón por la que se ha decidido desarrollar un juego del tipo *roguelite*, en lugar de *roguelike*? Para obtener la respuesta, es necesario recordar cuál es la audiencia objetivo del juego en desarrollo, así como cuál es la plataforma en la que se planea trabajar.

2.1.3. Mercado móvil, público casual y progresión

Tal y como se comentó en el apartado introductorio, OverColor, debido a las limitaciones temporales, se planificó como un juego con una complejidad no muy alta, fácil de entender y de comenzar a jugar, perfecto para un público más casual, y menos centrado en superar grandes desafíos que requieran de mucho tiempo, dedicación y esfuerzo; y, para apelar a ese público, se consideró que lo ideal era desarrollar el videojuego para las plataformas móviles.

Si damos un paseo por los juegos más populares de la Google Play Store [5], no será necesario investigar mucho para localizar un elemento común: la mayoría de juegos, o bien tienen un precio extremadamente bajo (inferior a los 5 euros), o bien siguen un modelo de negocio basado en las micro transacciones (en múltiples formas, como gacha, *lootboxes*, compra de divisas), anuncios, o una mezcla de varias. Además, no suelen hacer uso de increíbles gráficos capaces de competir con PC o consolas, ni sistemas profundos como los de los grandes juegos AAA; la experiencia está más orientada a jugadores que usan su móvil para jugar mientras esperan al bus, o quieren simplemente matar el rato.

Entrando más en profundidad en el tema entre manos, el género *roguelite* también está presente en el ámbito de los videojuegos para móviles; sin embargo, es más difícil encontrar juegos del género *roguelike* más puro. Y no es algo sorprendente, ya que la *permadeath* de los *roguelikes* puros es una mecánica más propia de los juegos más *hardcore* o de nicho, que esperan que el jugador ponga toda su alma en ellos para superar su gran dificultad mejorando su habilidad en cada partida; para un jugador casual, perder una *run* y sentir que ha sido fútil, y que no ha ganado nada, puede ser razón suficiente para no querer volver a jugar. Y es aquí donde entran los *roguelite*.

En un videojuego *roguelite*, aun cuando el jugador pierda, idealmente no sentirá que fue una "pérdida de tiempo"; verá como sus divisas han aumentado, y como puede mejorar sus estadísticas, a sabiendas de que la próxima vez será más fuerte. Y, aun cuando en esa próxima vez volviese a caer derrotado, esa derrota le ha acercado de forma tangible a la victoria, obteniendo más y más mejoras que mantendrá para siempre. Un ejemplo es el videojuego Candies 'n Curses (2018) [6], el primer juego de Tako Boy Studios.

Candies 'n Curses es un juego arcado del tipo roguelite, con una estética pixel-art y en 2D, en donde el jugador controlará a una joven llamada Molli Pop, quien se aventurará en una mansión encantada de la que debe escapar. La mansión estará repleta de fantasmas y otros monstruos, pero Molli los podrá derrotar iluminándolos con su linterna, avanzando, durante cada *run*, por los distintos pisos de la mansión, hasta enfrentar al jefe final.

De forma básica, en Candies 'n Curses el jugador cuenta con dos divisas diferentes: por un lado, las Almas, las cuales permiten desbloquear nuevas mejoras para Molli, como son los amuletos (mejoras permanentes del estilo "+10 puntos de vida" o "los coleccionables en un radio de X distancia se recogen solos", así como otras linternas diferentes con formas diferentes de juego (al estilo de las diferentes armas en Hades o los personajes de The Binding of Isaac (2011) [7], que si bien no cambian el juego, cambian la forma de jugarlo); por otro lado, la segunda divisa es el Plasma, la cual sirve para mejorar dichos amuletos o linternas que previamente obtuvimos con Almas. Ambas divisas se consiguen durante las *runs*, así como completando misiones del tipo "derrotar a 10 fantasmas", cuyo progreso también es guardado entre partidas. Una vez más, si bien al perder el jugador ha de comenzar su intento de huida de la mansión desde el comienzo, con cada *run* vendrá mejor

equipado con más y mejores amuletos, haciendo que sea mucho más fácil alcanzar pisos más avanzados.

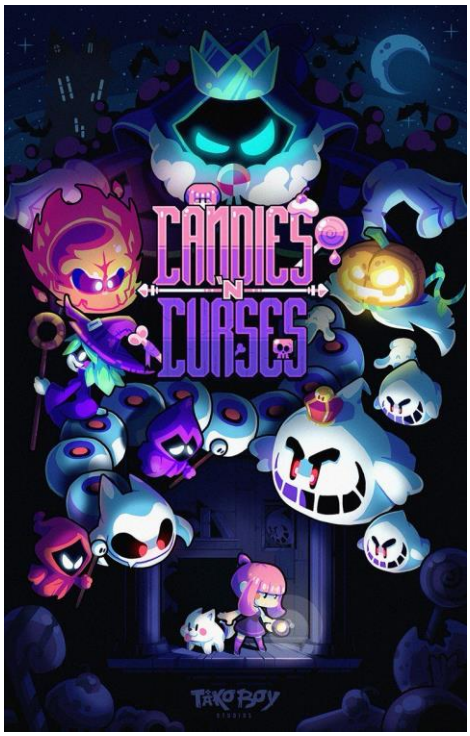
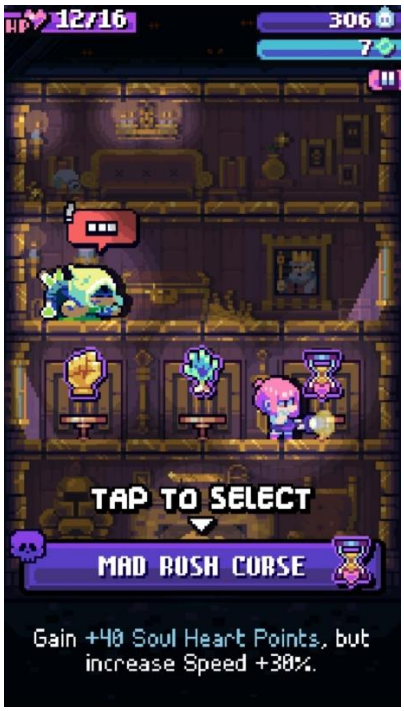


Figura 7. Candies 'n Curses (2018), un juego roguelite para dispositivos móviles

Figuras 8 y 9. Menú y progresión en Candies ‘n Curses a la izquierda, *gameplay* a la derecha

En cuanto al *gameplay* durante las *runs*, el objetivo es muy simple y fácil de entender: dentro de cada sala el jugador podrá mover a Molli de una planta a otra tocando la pantalla de arriba abajo o viceversa, así como cambiar la dirección en la que Molli camina arrastrando el dedo por la pantalla de forma horizontal. Con la linterna básica, Molli vencerá automáticamente a los enemigos que se encuentren justo delante de ella, pero es vulnerable a ataques de otras direcciones. Algunos enemigos soltarán Almas y Plasma, los cuales han de recogerse para progresar en los sistemas permanentes del juego. Tras vencer a un determinado número de enemigos en una sala, aparecerá un jefe con patrones de ataques más complejos, y que requerirá de múltiples golpes para ser derrotado. Tras cada sala de jefe, el jugador podrá escoger una recompensa de entre tres opciones, y avanzará al siguiente piso. Estas recompensas pueden ser caramelos (con efectos positivos) o maldiciones (con efectos positivos muy poderosos, pero también con efectos negativos), y se perderán por completo al finalizar la *run*.



Figuras 10 y 11. A la izquierda, una batalla contra un jefe en Candies ‘n Curses. A la derecha, una pantalla para obtener una mejora para la *run*.

Si bien Candies ‘n Curses no tiene la popularidad de otros títulos para dispositivos móviles como el juego de cartas en tiempo real Clash Royale (2016) [8], o el juego por turnos famoso por su sistema gacha y saga tras de sí Fate/GO (2015) [9], es un buen ejemplo de un juego de

estilo arcade con mecánicas *roguelite* implementado a dispositivos móviles para un público más casual, desarrollado por un equipo sin el gran renombre de otras grandes compañías. Por tanto, será una interesante fuente de inspiración para el videojuego en desarrollo, OverColor, desde un ámbito más cercano que el de otros *roguelite* como el ya mencionado Hades.

2.2. Tecnología y software de desarrollo

Para el motor del juego, desde un primer momento se supo que el desarrollo se realizaría en Unity [10]. Si bien existen otras opciones interesantes, como pueden set Unreal Engine [11] o GameMaker [12], Unity cuenta con una ventaja enorme que no comparten las otras alternativas: durante el desarrollo del máster en el que se cursa el presente Trabajo de Fin de Máster, se ha utilizado primordialmente Unity como plataforma de desarrollo, por lo no se requeriría de un proceso de aprendizaje previo profundo de la plataforma, así como se podría traducir lo aprendido en el itinerario académico de forma mucho más rápida y eficaz al proceso de construcción de OverColor.

Unity es un motor desarrollado por la compañía Unity Technologies que lleva funcionando desde 2005; originalmente estaba centrado en el desarrollo de videojuegos para entornos tridimensionales, pero progresivamente ha ido recibiendo grandes actualizaciones y mejoras, por lo que en la actualidad también es capaz de dar vida a juegos 2D. Bajo este motor se han creado algunos de los juegos más populares y característicos de los últimos años como Cuphead (2017) [13], Hollow Knight (2017) [14], Hearthstone (2014) [15], Ori and the Blind Forest (2015) [16], y una enorme y larga lista más de juegos de todos los géneros imaginables. Los lenguajes de programación que soporta son principalmente C++ y C#, y gracias a su creciente y activa comunidad de usuarios, es posible descargar desde su Unity Asset Store [17] todo tipo de módulos con funcionalidades avanzadas y *assets* con entornos, fuentes, objetos, sonidos, y muchos elementos más de forma rápida y, en muchos casos, gratuita, facilitando el proceso de construcción de un producto a todo tipo de desarrolladores. Además, el propio motor Unity puede ser descargado de forma gratuita para uso personal o académico.

Figura 12. Logotipo de Unity

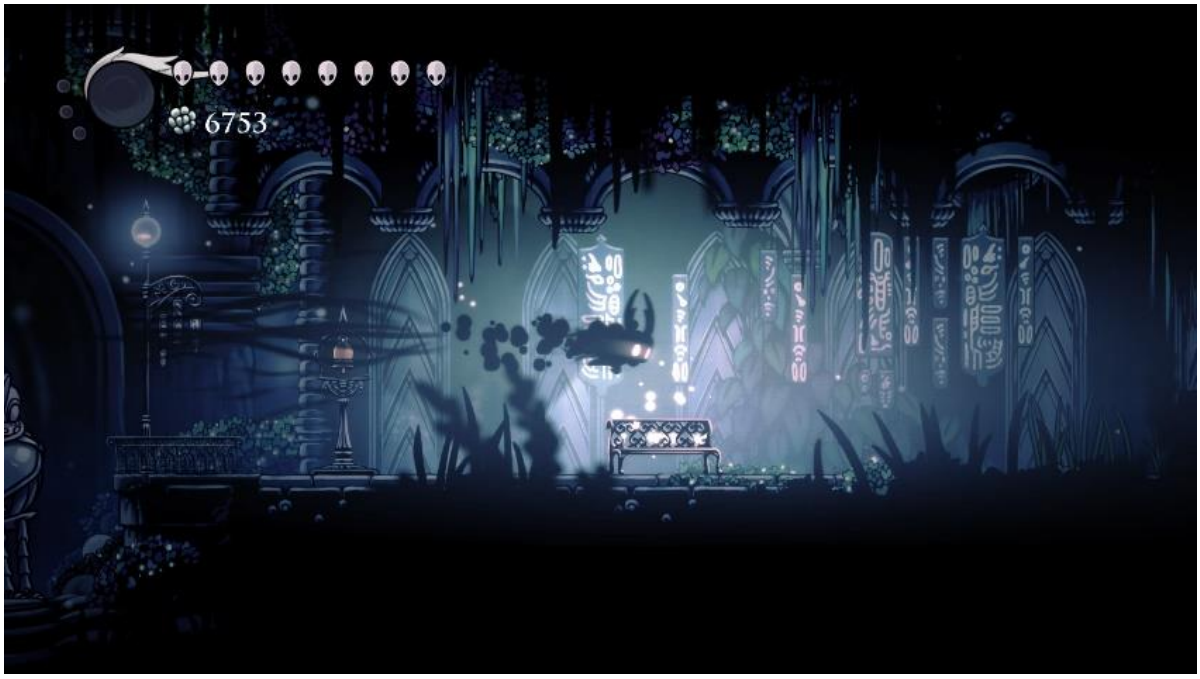


Figura 13. Hollow Knight (2017) fue desarrollado con Unity

Su principal competidor es probablemente Unreal Engine el cual, si bien no será utilizado en el proyecto, sigue siendo un entorno extremadamente potente y popular. Unreal Engine fue lanzado al mercado en 1998, y se ha mantenido como uno de los motores más notorios en el ámbito de los videojuegos. Desarrollado por la famosa Epic Games, ha dado lugar, a lo largo de sus múltiples iteraciones, a todo tipo de videojuegos de renombre, como la saga Bioshock [18], la saga Borderlands [19] o incluso Fortnite [20], quien se abrió paso en los últimos años de forma desmesurada. Usando como lenguaje principal C++, Unreal Engine se ofrecerá de forma gratuita a todos aquellos que quieran usarlo como motor de sus videojuegos, si bien Epic Games se quedará con un porcentaje de las ganancias en caso de que el producto genere grandes beneficios.

UNREAL
ENGINE

Figura 14. Logotipo de Unreal Engine

Figura 15. Fortnite (2017) fue desarrollado con Unreal Engine

Para finalizar, el software a emplear para la construcción del arte del videojuego será Aseprite [21], una herramienta de dibujo estilo píxel-art de gran popularidad debido a su facilidad de uso y alto margen de posibilidades, así como su conjunto de herramientas centradas en el proceso de creación de animaciones. Aseprite no es un software gratuito; no obstante, puede comprarse desde su sitio web oficial o desde Steam por menos de 20 euros. Si bien existen gran multitud de herramientas en el mercado que permiten la construcción de arte pixel-art, así como herramientas generales de arte como GIMP [22], Aseprite es perfecto cuando se tiene en mente el desarrollo de un videojuego.

Figura 16. Logotipo de Aseprite

3. Propuesta

En el presente apartado se formalizará la propuesta del videojuego a construir, explicando elementos importantes como son la historia, la *setting*, los enemigos, los sistemas de progresión, así como otros elementos del *gameplay* tal que las mejoras que se van obteniendo durante las partidas.

3.1. Descripción del videojuego

OverColor es un videojuego diseñado para dispositivos móviles del género *roguelite* con un *gameplay* caracterizado por acción arcade con toques de puzle. El jugador tomará el control de la protagonista, Roloc, mientras asciende por un arcoíris tratando de devolverle el color al mundo, enfrentando enemigos en el proceso en un *gameplay* basado en intentos o *runs*, consiguiendo mejoras que le ayuden a llegar más lejos. Cuando Roloc sea derrotada, se caerá del arcoíris y tendrá que volver al principio, perdiendo las mejoras obtenidas. Sin embargo, sí se quedará con el color que obtuvo durante el camino, que podrá utilizar para volver a colorear a su villa, otorgándole mejoras permanentes que harán que cada intento le acerque más a la victoria.

Durante su ascenso, Roloc irá armada con el OverColor, una invención suya que le permitirá derrotar a los enemigos que encontrará en el arcoíris, absorbiendo su color. Dentro del *gameplay*, el arcoíris estará dividido en tres líneas entre las que el jugador podrá intercambiar la posición de Roloc arrastrando el dedo por la pantalla, desde las cuales diversas criaturas del arcoíris cargarán agresivamente contra Roloc. Además, el jugador contará con tres botones: uno rojo, uno amarillo, y uno azul. Cuando el jugador pulse uno de los botones, todos los enemigos de la línea en la que Roloc se encuentra verán su color absorbido y serán derrotados, mas los botones tienen un enfriamiento, por lo que no es suficiente con pulsar muy rápido; además, algunos enemigos son más complejos, requiriendo de varios ataques diferentes de múltiples colores para ser derrotados.

El objetivo final del juego será alcanzar el final del arcoíris llegando a las nubes, las cuales se formaron por la evaporación de los colores del mundo, para poder así restaurar el color a nivel mundial. Para poder alcanzar el final del arcoíris, será necesario cruzar por los distintos colores del arcoíris, que marcarán las diversas zonas del juego.

3.2. Historia y ambientación

La historia en OverColor es un elemento secundario que existe para apoyar a la jugabilidad, y no es el foco de la experiencia; es una explicación o justificación de los elementos del juego.

Hace un tiempo, un enorme diluvio azotó al mundo. Tal fue la potencia del mismo, que el agua se llevó el color de todas las cosas, volviendo al mundo un lugar gris y aburrido. Sin embargo, toda tormenta siempre acaba, y al finalizar el gran diluvio, se formó un arcoíris de entre las nubes; además, el agua que se había llevado el color ahora se ha evaporado, y ha formado coloridas nubes en el cielo.

A la protagonista, Roloc, no le gusta la situación actual, por lo que ha construido una máquina, el OverColor, capaz de absorber el color de las cosas. Con esto, Roloc se propone subir por el arcoíris, el cual está lleno de peligros y enemigos, para llegar a las nubes y devolver al mundo su color haciendo uso del OverColor.

El juego comenzaría en este punto; las *runs* representarán intentos de Roloc de alcanzar las nubes. Cada vez que fracase, se caerá del arcoíris y volverá a caer al suelo, en su villa, por lo que deberá volver a comenzar desde el principio. Sin embargo, el color que ha absorbido por el camino le permitirá ir recoloreando su villa, hasta que finalmente sea capaz de llegar a las nubes. Una vez alcance las nubes a efectos prácticos será el final de la historia, con el mundo recuperando su color, si bien el juego podrá seguir siendo jugado.

La ambientación o *setting* es una bastante sencilla y alegre, en un mundo no muy definido, el cual se asume es en cierto grado similar al nuestro, pero con elementos fantásticos como el arcoíris o tecnología inexistente como el OverColor.

3.3. Personajes y enemigos

En OverColor, de forma general, tan solo existe un único personaje, Roloc (la protagonista), por lo que todos los diálogos serán a sí misma y no habrá interacciones entre personajes. El objetivo de OverColor es, tal y como fue explicado, centrar el foco de atención en el *gameplay* y no en la historia.

Por otro lado, se encuentran los diversos enemigos. Los enemigos no serán catalogados como personajes ya que no contarán con diálogos ni roles narrativos, si bien serán variados con el objetivo de desafiar al jugador mientras lo enfrenta. Cabe destacar que su diseño trata de no ser extremadamente complejo, ya que la dificultad de OverColor no radica en derrotar a enemigos específicamente complejos y profundos, sino en vences hordas de múltiples enemigos que vienen constantemente desde múltiples direcciones a los que hay que evitar a toda costa.

Los enemigos se catalogarán en diferentes tipos. En primer lugar, es posible catalogarlos según su color – método básico para ser derrotados:

- Enemigos simples: enemigos de uno de los tres colores básicos (rojo, amarillo, azul), los cuales serán inmediatamente derrotados cuando se les golpee con su color.
- Enemigos de color doble: enemigos con un color que combine otros dos (verde = azul + amarillo, naranja = rojo + amarillo, morado = rojo + azul). Cuando son golpeados por un ataque de uno de sus dos colores, se transformarán en un enemigo simple con el color restante.
- Enemigos de colores oscuros: enemigos con colores básicos más oscuros los cuales, cuando reciben un golpe de su color, se transforman en enemigos simples de su mismo color.

En segundo lugar, se podrán catalogar según sus posibles mecánicas adicionales:

- Enemigos con escudo: enemigos con algún tipo de barrera de un color específico (puede ser simple o doble). Para poder dañarlos, es necesario romper antes la barrera golpeándola con su color correspondiente.
- Enemigos explosivos: enemigos de cualquier otro tipo que, cuando son derrotados, comienzan una pequeña cuenta atrás que tras alcanzarse provoca una explosión en la línea que fueron derrotados, dañando al jugador si se encuentra en la línea en el momento de la explosión.
- Enemigos tiradores: enemigos de cualquier otro tipo que en lugar de avanzar hacia al jugador, se mantienen a una distancia media lanzando de vez en cuando proyectiles, los cuales dañan al jugador en caso de impacto.

- Enemigos excavadores: enemigos de cualquier otro tipo que se desplazan bajo tierra hasta que se encuentran a una distancia media del jugador (mitad de la línea). Cuando están bajo tierra, no pueden ser golpeados por los ataques del jugador.
- Enemigos saltarines: enemigos de cualquier otro tipo que saltan de una línea a otra de forma aleatoria cada varios instantes

3.4. Recursos

En este apartado se explicarán los distintos recursos que el jugador deberá gestionar durante su experiencia en OverColor. Los recursos engloban una gran cantidad de elementos diferentes del juego, por lo que se dividirán en subcategorías.

3.4.1. Puntos de vida

Tal y como es común en el mundo de los videojuegos, los puntos de vida son un contador que representa cuantas veces puede el jugador recibir golpes de sus enemigos hasta que se considere que la partida ha acabado.

En OverColor, los puntos de vida se representarán de forma numérica; cuando un enemigo dañe a Roloc (la mayoría de las veces debido a que ha cruzado la línea) el número bajará; la cantidad de puntos de vida restantes del jugador será visible en todo momento en el HUD.

Los puntos de vida, una vez perdidos, pueden volver a recuperarse, ya sea recogiendo corazones dejados por los enemigos, o bien mediante algunas de las mejoras que pueden escogerse conforme se avanza en una *run*; no obstante, los puntos de vida no podrán subir por encima de su cantidad máxima.

Por otro lado, la cantidad de puntos de vida máxima del jugador puede aumentar durante la *run* con mejoras, o de forma permanente mediante el sistema de progresión del juego basado en colorear la villa, tal y como se verá más adelante.

3.4.2. Recursos del sistema de progresión

El sistema de progresión permanente de OverColor queda representado mediante una villa a la que hay que devolverle el color poco a poco; por tanto, el recurso para representar la divisa a emplear es el propio “color” que Roloc ha recogido durante sus intentos de ascenso.

De esta forma, habrá tres divisas: color rojo, color amarillo, y color azul. Los distintos edificios de la villa requerirán que se invierta cierta cantidad de o bien uno de estos recursos o una combinación de varios para desbloquearse y posteriormente mejorarse. Al hacer dichas mejoras, el jugador será recompensado con diversos beneficios, como mayores puntos de vida máxima, menor tiempo de enfriamiento de sus botones, la posibilidad de comenzar con alguna mejora o incluso oportunidades extra para “revivir” en caso de una derrota.

Las tres divisas se obtendrán al derrotar enemigos; los enemigos otorgarán divisa al jugador correspondiente a su color, así como los enemigos de colores dobles darán divisa de ambos tipos. Los enemigos más fuertes darán más divisa que los enemigos más simples. Por otro lado, algunas mejoras otorgarán directamente al jugador alguna de estas tres divisas, o bien multiplicadores que aumenten la cantidad de divisa obtenida en esa *run*.

Los edificios de la villa que pueden colorearse haciendo uso de estas divisas son:

- Taller de Roloc (recurso amarillo): conforme se coloree el taller de Roloc, se reducirá ligeramente el tiempo de enfriamiento de los botones.
- Casa de Roloc (recurso rojo): conforme se coloree la casa de Roloc, aumentará la vida máxima de Roloc.
- Carreteras (recurso azul): conforme se coloreen las carreteras, se desbloquearán nuevas localizaciones que podrán ser coloreadas (el bosque, el supermercado y el aeropuerto).
- Bosque (recurso amarillo + recurso azul): conforme se coloree el bosque, aumentará las probabilidades de encontrar mejoras más poderosas durante el ascenso por el arcoíris.
- Supermercado (recurso amarillo + recurso rojo): conforme se coloree el supermercado, Roloc podrá comenzar los ascensos con alguna o algunas mejoras extra de forma inicial.
- Aeropuerto (recurso azul + recurso rojo): conforme se coloree el aeropuerto, Roloc podrá obtener vidas extras que le permitan “revivir” en caso de caer derrotada.

3.4.3. Mejoras

Las mejoras son objetos que el jugador irá obteniendo durante los intentos de ascenso que le irán otorgando más poder o nuevas posibilidades. Cada vez que el jugador complete una zona, se le ofrecerá a elegir entre tres mejoras elegidas de forma aleatoria de una *pool* de

opciones; importante destacar que no todas las mejoras tienen un nivel de poder similar, aumentando el grado de variación entre partidas.

La lista de mejoras de OverColor es la siguiente (los números finales no están definidos ya que han de probarse antes):

- Amor: aumenta en una cantidad ligera los puntos de vida máximos.
- Botiquín: cura al completo los puntos de vida de Roloc.
- Máquina de pulsos rota: cada varios segundos emite un pulso de un color aleatorio que daña a los enemigos.
- Máquina de pulsos (disponible en rojo, azul y amarillo): cada varios segundos emite un pulso de su color asociado que daña a los enemigos de ese color.
- Super Máquina de pulsos (disponible en verde, naranja y morado): cada varios segundos emite un pulso de su color asociado que daña a los enemigos de ese color o de sus subcolores.
- Baterías nuevas: reduce el tiempo de enfriamiento de los botones y mejoras.
- Ticket de avión: ofrece una vida extra al jugador cuando pierde, permitiéndole revivir una vez.
- Bomba de color: mejora que hará que en la pantalla del jugador aparezca un nuevo icono. Al pulsarlo, todos los enemigos que estén actualmente en pantalla serán inmediatamente derrotados. La mejora tiene un enfriamiento para volver a usarse largo.
- Multiplicador de color (disponible en rojo, azul y amarillo): multiplica en cierta medida las ganancias de la divisa del color correspondiente al color de la mejora.
- Torreta portátil (disponible en rojo, azul y amarillo): torreta que sigue a Roloc y va disparando de vez en cuando un rayo de su color asociado.
- Super Torreta portátil (disponible en verde, naranja y morado): torreta que sigue a Roloc y va disparando de vez en cuando un rayo de su color asociado.
- Ultra Amor: aumenta de forma considerable los puntos de vida máximos.
- Ultra Baterías nuevas: reduce considerablemente el tiempo de enfriamiento de los botones y mejoras.
- La Caja: al obtenerse, otorga al jugador una mejora aleatoria de entre todas las posibles del juego.

- Espejo: de vez en cuando, generará una réplica de Roloc en una línea aleatoria. Durante unos segundos, la copia se mantendrá en la línea replicando todos los ataques que haga Roloc. La copia es inmune a cualquier tipo de ataque.
- Trébol de cuatro hojas: aumenta las probabilidades de recibir mejoras de mayor calidad durante el resto del intento de ascenso.
- Turbo!: aumenta la velocidad a la que vienen los enemigos, pero también reduce drásticamente el enfriamiento de los botones, aumenta la vida máxima y cura a Roloc.
- Buscacorazones: aumenta la probabilidad de que los enemigos suelten corazones al ser derrotados.

Tal y como ya se comentó con anterioridad, hay mejoras más efectivas que otras, e incluso diversos sistemas del juego lo aprovechan aumentando la probabilidad de obtener mejoras más poderosas. Las mejoras que catalogaremos como “poderosas” son: Super Amor, Super baterías nuevas, Super Máquina de pulsos (cualquiera), Gran Escudo (cualquiera), Bomba de Color, Super Torreta Portátil (cualquiera), Espejo mágico, Turbo!. Normalmente, tendrán una probabilidad mucho más reducida de aparecer que las otras mejoras.

3.5. Objetivos planteados al jugador

Para finalizar, se hablará de los objetivos que tendrá el jugador durante su tiempo de juego en OverColor.

Como juego *roguelite* que es, OverColor da gran importancia a su sistema de progresión. De esta forma, si bien los primeros intentos deberían terminar en derrotas, conforme el jugador dedique más tiempo y avance en el coloreado de la villa, Roloc se volverá más fuerte y al jugador le resultará más sencillo llegar más lejos. Por tanto, el objetivo principal del juego es progresar en el sistema de coloreado de la villa.

Dentro de los intentos de ascenso en sí, el jugador tendrá el objetivo simultáneo de preservar sus puntos de vida y derrotar a los enemigos; la mayoría de enemigos hacen perder vida a Roloc cuando la alcanzan, por lo que eliminarlos es la forma más efectiva de mantener la vida intacta. Cuando la vida baje a cero, el intento de ascenso finalizará, y se regresará a la villa para preparar un nuevo intento.

A nivel narrativo, el objetivo del juego es alcanzar las nubes completando exitosamente todo el ascenso. Sin embargo, aun cuando se logre, el juego podrá seguir siendo jugado y los sistemas de progresión podrán seguir completándose.

4. Diseño

En este apartado se explicarán las distintas decisiones de diseño tomadas como parte del desarrollo del proyecto, como son las herramientas y sistemas utilizados de forma final, los recursos empleados, así como la arquitectura del juego.

4.1. Entorno de desarrollo final

Tal y como se explicó en el apartado *2.2. Tecnología y software de desarrollo*, el motor empleado para el desarrollo de OverColor fue Unity, debido a que es la herramienta que ha sido empleada mayoritariamente durante el periodo lectivo del máster correspondiente al presente Trabajo de Fin de Máster, con la cual se ha obtenido además mayor experiencia y conocimiento. Se valoraron otras opciones, pero de forma final Unity era una opción demasiado sólida como para ser ignorada.

De forma específica, la versión de Unity con la que ha sido desarrollado el proyecto es la versión 2020.3.28f1 [23], la cual ya fue utilizada con anterioridad durante el periodo lectivo.

Los requisitos mínimos hardware para el uso de la herramienta Unity en un equipo son los siguientes [24]:

- **Windows**:
  - Sistema Operativo: Windows 7 (SP1+) y Windows 10, tan solo versiones 64-bit.
  - CPU: Arquitectura X64 con soporte para SSE2.
  - API Gráfica: GPUs capaces de usar DX10, DX11 y DX12.

- **macOs**:
  - Sistema Operativo: High Sierra 10.13+.
  - CPU: Arquitectura X64 con soporte para SSE2.
  - API Gráfica: GPUs Intel y AMD con soporte para Metal.

- **Linux**:
  - Sistema Operativo: Ubunto 16.04, Ubuntu 18.04 y CentOS 7.
  - CPU: Arquitectura X64 con soporte para SSE2.

- API Gráfica: OpenGL 3.2+ o GPUs Nvidia y AMD con soporte para Vulkan.

De forma específica, para el desarrollo de OverColor se hizo uso de un equipo con las siguientes características:

- Sistema Operativo: Windows 10, versión 64-bit.
- Procesador: Intel(R) Core(TM) i5-8400
- Memoria RAM: 16GB
- Tarjeta Gráfica: NViDIA GeForce GTX 1660 Ti

4.2. Otras herramientas utilizadas

A parte del motor Unity, para la realización del proyecto se ha hecho uso de diversas herramientas:

- **Visual Studio 2019** [25]: Visual Studio es un IDE centrado en el desarrollo en .NET y C++ desarrollado por Microsoft. Fue utilizado como IDE para la edición de los scripts desarrollados en el proyecto en Unity, los cuales estarán escritos en C#.

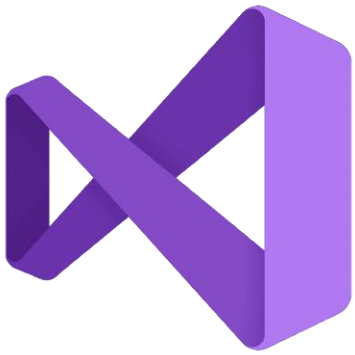


Figura 17. Logo de Visual Studio

- **GitLab** [26]: GitLab es una herramienta centrada en el mantenimiento de repositorios Git y de control de versiones y del ciclo de vida del proyecto; gracias a su funcionalidad, es posible gestionar con facilidad y rapidez cambios en el código fuente de un proyecto, así como permite la automatización de muchos procesos.

  GitLab tuvo una doble funcionalidad en el proyecto: por un lado, fue empleada para el control del proyecto, permitiendo, tal y como se mencionó anteriormente, controlar su evolución y versiones, permitiendo devolver al proyecto a una versión anterior en caso

de error crítico; por otro lado, GitLab también sirve como puente entre el alumnado y el profesorado, permitiendo que el tutor del presente Trabajo de Fin de Máster revisase y controlase el progreso en el proyecto de forma rápida y efectiva.

Figura 18. Logo de GitLab

- **SourceTree** [27]: SourceTree es una herramienta centrada en la simplificación de la gestión y control de repositorios Git, ofreciendo una interfaz visual amigable y fácil de usar que permite manejar múltiples repositorios sin tener que hacer uso de comandos por consola, reduciendo enormemente la dificultad de uso y acelerando el proceso de desarrollo.

  Gracias a SourceTree, se controló con mayor facilidad y rapidez el repositorio levantado en GitLab.

Figura 19. SourceTree

- **Aseprite** [21]: Tal y como se explicó en el apartado *2.2. Tecnología y software de desarrollo*, para la construcción de los assets píxel-art se hizo uso de la popular herramienta Aseprite, la cual está equipada con toda la funcionalidad necesaria para

crear *sprites* píxel-art bidimensionales y sus respectivas animaciones, con la posibilidad de exportar los resultados en diversos formatos que faciliten su uso dentro del proyecto.

- **Trello** [28]: Trello es una herramienta desarrollada con el objetivo de ofrecer a sus usuarios la posibilidad de gestionar tareas mediante la creación de tableros virtuales colaborativos en los que crear diversas tarjetas a modo de tareas clasificadas en listas, siguiendo la ideología principal propuesta por la metodología Kanban.

  En el proyecto la herramienta fue utilizada para la gestión de las tareas del proyecto, la cual, tal y como se explicó en el primer apartado del presente documento, hace uso de la metodología Kanban adaptada al trabajo individual.

Figura 20. Logo de Trello

4.3. Inventario de assets utilizados

A continuación, en este apartado se detallarán los *assets* utilizados para el apartado artístico (visual y auditivo) del proyecto, así como su origen y localización en el proyecto. Con el objetivo de presentar la información de forma ordenada, el apartado se dividirá en diferentes subsecciones.

4.3.1. Assets visuales

Dentro del apartado de los *assets* visuales, nos referiremos principalmente a los *sprites* de los distintos elementos del juego, así como de los elementos del HUD. Cabe destacar que la totalidad de los *sprites* y elementos visuales fueron desarrollados manualmente con el propósito de este Trabajo de Fin de Máster, por lo que no fueron obtenidos de fuentes

externas. Tal y como se explicó en apartados anteriores, todos ellos fueron construidos con el software Aseprite.

Como elemento general, la estética general del juego es estilo píxel-art, por lo que la mayoría de los *sprites* descansarían sobre esta categoría artística.

Tras esa aclaración, se comenzará hablando de Roloc, el avatar del jugador. Su apariencia es muy simple; se trata de una joven de cabello castaño, la cual va pilotando una extraña máquina similar a una moto, el OverColor. El OverColor cuenta, además, con un cañón, desde el que saldrán los ataques, si bien éstos son elementos independientes. Dentro del juego, Roloc cuenta con una animación por la que las ruedas de la moto se mueven, a la vez que la bufanda/capa de Roloc se va moviendo debido al viento y la velocidad del movimiento. Existe una segunda animación en la que Roloc pestañea, la cual se activa cuando se realiza un ataque. Todos los *frames* de las animaciones se construyeron uno a uno y colocaron en consecución.

Figura 21. Sprite de Roloc, el avatar del jugador

Por otro lado, existen los ataques. Cuando el jugador pulsa uno de los botones, o cuando alguna mejora como las torretas se activan, aparecerá por pantalla un rayo de energía en la línea en la que se haya realizado dicho ataque, simbolizándolo. Estos ataques serán rayos alargados del color correspondientemente utilizado, y contarán con una animación por la que irán generándose hacia delante con el movimiento de sus partículas internas. Existe una versión de estos *sprites* y animaciones para cada color posible de ataque, incluyendo los de las torretas mejoradas.

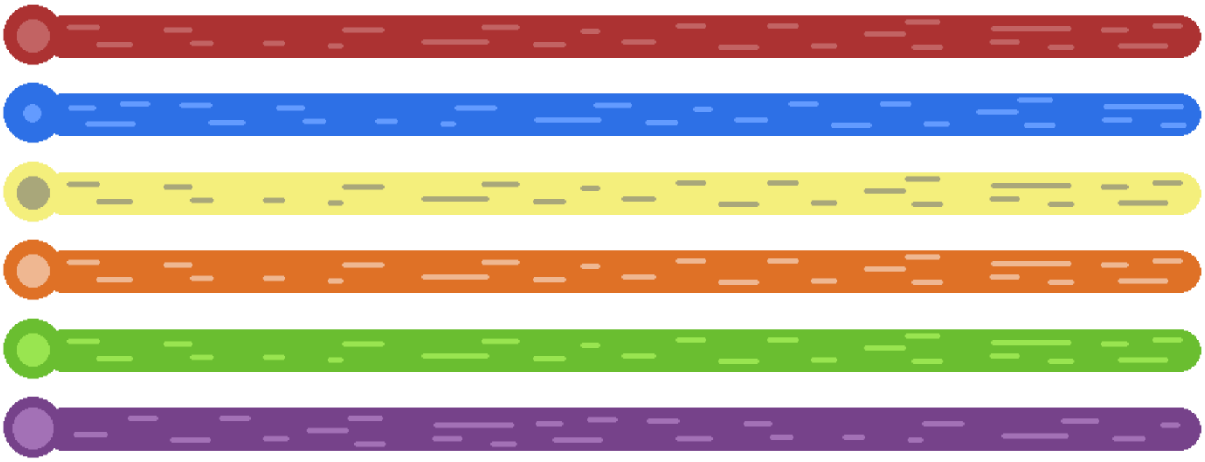


Figura 22. Sprites de los ataques

Otro elemento importante son los enemigos, los cuales también contarán con sus *sprites* y animaciones. Según el color del mismo, cada enemigo tendrá un Sprite único y diferente, si bien son mayoritariamente parecidos unos con otros; comparten, además, una animación muy similar de movimiento.

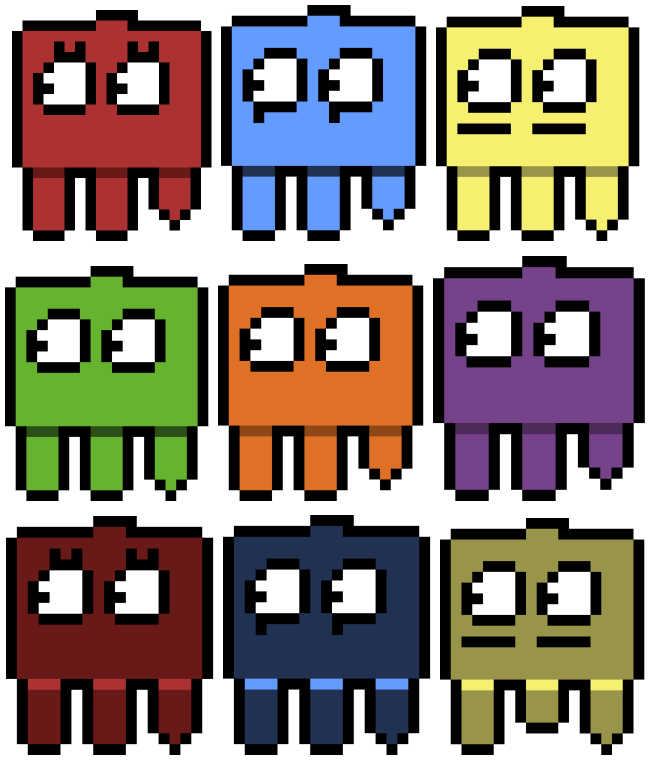


Figura 23. Sprites de los enemigos de los diferentes colores

Sin embargo, los enemigos no están tan solo caracterizados por su color; algunos enemigos cuentan con habilidades especiales que usarán para tratar de ponérselo difícil al jugador.

Estas habilidades usan *sprites* diferentes, los cuales son superpuestos a los *sprites* ya existentes de los enemigos. Algunas habilidades no cuentan con ningún Sprite asociado, así como otras requieren que se les aplique un coloreado dentro de Unity específico.

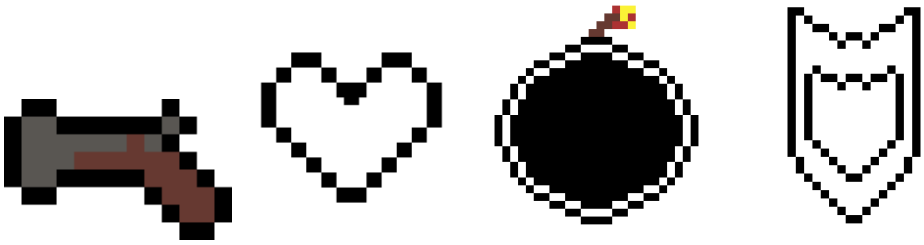


Figura 24. Sprites de los power-ups de los enemigos

Otro elemento visual importante son los edificios de la villa, los cuales pueden ser mejorados. Para representarlos, en lugar de realizar dibujos de edificios como tal, se optó por la creación de iconos que representasen al edificio en cuestión, coloreados del color o combinación de colores necesarios para poder mejorar el mismo. Existe un *sprite* diferente por cada uno de los seis edificios.

Figura 25. Sprites de los edificios de la villa

También han de destacarse los *sprites* de las mejoras que se obtienen durante las *runs*. Para diferenciarlas unas de otras, todas cuentan con su *sprite* propio y único, si bien aquellas que

están relacionadas (por ejemplo, máquinas de pulsos o torretas de colores diferentes) tienen diseños muy parecidos. Adicionalmente, aquellas mejoras de calidad rara tendrán un bordeado de color amarillo para diferenciarlas del resto.

Figura 26. Sprites de las mejoras

Para finalizar, cabe destacar que los *sprites* mostrados no representan la totalidad de los elementos visuales del juego, si bien son los más destacables. También hay múltiples objetos con otras funcionalidades, como para construir las interfaces de usuario o para conformar los iconos o sistemas de partículas del juego. Entre estos destacamos, por ejemplo, el icono del juego que se mostrará en el dispositivo móvil una vez esté instalado.

Figura 27. Logo del juego

4.3.2. Assets de audio

Una vez revisados los *sprites* del proyecto y sus animaciones, se procederá a explicar el siguiente tipo de elementos: los *assets* de audio. Dentro de este grupo se englobarán tanto las canciones como los propios efectos de sonido.

En cuanto a las canciones, en OverColor se hace uso de cinco canciones diferentes, que se reproducen en lugares específicos del juego. A diferencia de los *sprites* y *assets* visuales, que fueron construidos específicamente para este proyecto, las canciones fueron obtenidas desde el sitio web especializado en música gratuita de uso libre Free Stock Music [29]. Las canciones en cuestión serían:

- En la pantalla de la villa suena la canción “Tavern Loop One”, realizada por Alexander Nakarada. Dicha canción requiere de la siguiente referenciación:

  Tavern Loop One by Alexander Nakarada |
  https://www.serpentsoundstudios.com
  Music promoted by https://www.free-stock-music.com
  Attribution 4.0 International (CC BY 4.0)
  https://creativecommons.org/licenses/by/4.0/

- En una run, durante las tres primeras zonas sonará la canción “Pixel Story” de Roa Music. Su referenciación es la siguiente:

  Pixel Story by Roa Music | https://soundcloud.com/roa_music1031
  Music promoted by https://www.free-stock-music.com
  Creative Commons Attribution 3.0 Unported License
  https://creativecommons.org/licenses/by/3.0/deed.en_US

- De igual forma, durante las tres últimas zonas del *gameplay*, la canción que estará sonando es en su lugar “Mountain Trials”, de Joshua McLean. La referencia sería la siguiente:

  Mountain Trials by Joshua McLean | http://mrjoshuamclean.com
  Music promoted by https://www.free-stock-music.com
  Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)
  https://creativecommons.org/licenses/by-sa/4.0/

- La cuarta canción, correspondiente a la pantalla de derrota/fin del ascenso, es titulada “Elevator Pitch” de PYC Music, con la siguiente referencia:

  Elevator Pitch by PYC Music | https://soundcloud.com/pycmusic
  Music promoted by https://www.free-stock-music.com
  Attribution-NoDerivs 3.0 Unported (CC BY-ND 3.0)
  https://creativecommons.org/licenses/by-nd/3.0/

- Para finalizar, la quinta y última canción, la cual suena durante la pantalla de victoria tras una *run* exitosa, es “Victory”, de MaxKoMusic. Su referencia es la siguiente:

  Victory by MaxKoMusic | https://maxkomusic.com/
  Music promoted by https://www.free-stock-music.com
  Creative Commons Attribution-ShareAlike 3.0 Unported
  https://creativecommons.org/licenses/by-sa/3.0/deed.en_US

Sin embargo, tal y como se adelantó con anterioridad, las canciones no son los únicos *assets* sonoros empleados; también es necesario considerar los efectos de sonido. En total se hizo uso de X archivos de audio dedicados a efectos sonoros. Todos los archivos fueron obtenidos del sitio web Mixkit [30], el cual ofrece una gran librería de recursos de uso abierto. La lista de los efectos es la siguiente:

- Arcade-retro-jump-233, empleado para el sonido de los rayos lanzados por el OverColor.
- Cartoon-falling-whistle-395, para la caída de Roloc en la pantalla de derrota.

- Chor-bless-bell-656, para cuando se le da la opción al jugador de escoger una mejora.
- Explosion-2759, para las explosiones de las bombas dejadas por algunos enemigos, o para la mejora específica Bomba de color.
- Failure-arcade-alert-notification-240, para cuando se recibe daño.
- Fairy-arcade-sparkle-866, para cuando se recibe una curación.
- Fall-on-foam-splash-754, para cuando se derrota a un enemigo.
- Hand-saw-tool-on-wood-827, para cuando se mejora un edificio.
- Magic-festive-melody-2986, que suena en la pantalla de título.
- Metal-hammer-hit-833, para cuando se abre el menú de un edificio.
- Metal-tool-drop-835, para cuando se le rompe el escudo a un enemigo acorazado.
- Video-game-mistery-alert-234, para cuando se abre el menú de pausa.

4.3.3. Otros recursos

Para finalizar, se hizo uso de un recurso externo adicional: la fuente del texto empleada en la mayoría de los cuadros de texto del juego, la cual es estilo píxel-art, fue obtenida de la Unity Asset Store bajo el nombre de “Free Pixel Font – Thaleah” [31].

Figura 28. Thaleah Fat Free Pixel Font

4.4. Diseño de niveles

Por su naturaleza, los niveles en OverColor son algo particular ya que la pantalla en la que se desarrolla todo el juego es siempre la misma, si bien lo que cambia entre niveles son otros elementos, como el ratio de aparición de enemigos o sus cualidades, así como el color del fondo. De forma narrativa, los niveles en OverColor representan “colores del arco iris” por los que la protagonista irá ascendiendo.

A efectos prácticos, existen once parámetros principales que definen a un nivel en OverColor. Dichos parámetros son los siguientes:

- **Zone Color:** Representa el color del nivel; servirá para colorear el fondo del mismo, de forma que represente uno de los colores del arco iris.
- **Zone Number:** Representa el número del nivel, el cual será mostrado al jugador al comienzo del mismo para que pueda saber por qué fase se encuentra.
- **Spawn Rate:** Representa el ratio de aparición de enemigos en el nivel. A más alto, se generarán más enemigos, por lo que los primeros niveles lo tendrán muy bajo, e ira subiendo progresivamente en las fases más avanzadas.
- **Speed:** Velocidad a la que se mueven los enemigos en el nivel. A más alta, más rápidos serán, aumentando considerablemente la dificultad.
- **Duration:** Duración en segundos del nivel. En OverColor, los niveles no están definidos por una cantidad fija de enemigos a derrotar o una meta, sino a una duración temporal.
- **Double Color Chance:** Valor entre 0 y 100 que representa la probabilidad de que los enemigos generados tengan un color doble. Un color doble es compatible con las otras cualidades posibles, pero no con un color oscuro.
- **Dark Color Chance:** Valor entre 0 y 100 que representa la probabilidad de que los enemigos generados tengan un color oscuro. Un color oscuro es compatible con las otras cualidades posibles, pero no con un color doble.
- **Shielded Chance:** Valor entre 0 y 100 que representa la probabilidad de que los enemigos cuenten con un escudo. Si un enemigo tiene un escudo, no tendrá otras cualidades especiales.
- **Explosive Chance:** Valor entre 0 y 100 que representa la probabilidad de que los enemigos sean explosivos y exploten al morir. Si un enemigo tiene está habilidad, no tendrá otras cualidades especiales.
- **Shooter Chance:** Valor entre 0 y 100 que representa la probabilidad de que los enemigos sean capaces de disparar proyectiles. Si un enemigo tiene está habilidad, no tendrá otras cualidades especiales.
- **Grounded Chance:** Valor entre 0 y 100 que representa la probabilidad de que los enemigos aparezcan escondidos bajo tierra siendo invulnerables temporalmente. Si un enemigo tiene está habilidad, no tendrá otras cualidades especiales.

- **Jumper Chance:** Valor entre 0 y 100 que representa la probabilidad de que los enemigos sean capaces de saltar de una línea a otra del mapa. Si un enemigo tiene está habilidad, no tendrá otras cualidades especiales.

Por tanto, mediante el balanceo de estos parámetros, podemos crear de forma rápida y fácil distintos niveles, ya que son prácticamente la única diferencia entre las diversas fases del juego.

Sin embargo, si bien los parámetros presentados anteriormente permiten diferenciar a los niveles, todos estos cuentan con un par de elementos comunes; más específicamente, todos los niveles cuentan con la misma estructura:

- **Fase inicial:** período de unos segundos en los que se muestra por pantalla el nombre del nivel. Durante esta fase, no aparecen enemigos.
- **Fase de juego:** periodo en el que ocurre el *gameplay*. Su duración depende del parámetro Duration visto anteriormente. Aquí, irán generándose enemigos que el jugador deberá derrotar hasta alcanzar el final de la fase.
- **Fase final:** periodo final del nivel. Durante esta fase no se generarán enemigos. Unos instantes antes de acabar, se le ofrecerá al jugador una mejora como recompensa por haber completado la fase.

Una vez vista la estructura de una zona, se pasará a explicar la curva de dificultad a lo largo de una run completa. Actualmente, una *run* de OverColor cuenta con seis zonas diferentes, cada cual más difícil que la anterior. De forma más específica, la progresión a lo largo de una run sería de la siguiente forma:

- **Zona 1:** la primera zona, de color azul, es la más sencilla de todas. No aparecerán demasiados enemigos, y estos no serán demasiado rápidos. Habrá una pequeña probabilidad de que algunos de los enemigos tengan un color doble o posean una bomba, presentando algunas mecánicas.
- **Zona 2:** la segunda, de color verde, sigue siendo una zona moderadamente sencilla. El número de enemigos será el mismo que en la zona anterior, si bien serán ligeramente más rápidos. La dificultad añadida estará en que los enemigos dobles serán más comunes, y empezarán a aparecer enemigos disparadores.

- **Zona 3:** la tercera zona empezará a mostrar un aumento de dificultad un poco más notable. Los enemigos serán un poco más rápidos y se generarán más. Además, comenzarán a aparecer enemigos saltarines con una probabilidad moderadamente alta.
- **Zona 4:** a partir de este punto, que marca la mitad de una *run*, la dificultad aumentará notoriamente, al mismo tiempo que cambia la música, mostrando como la cosa "se pone seria". Los enemigos serán más rápidos, y al mismo tiempo que se mantendrán los enemigos ya presentados, empezarán a generarse además enemigos con todas las cualidades restantes; enemigos con escudos y enemigos escondidos bajo tierra.
- **Zona 5:** si bien la Zona 4 fue el gran aumento de dificultad, la Zona 5 sigue manteniendo el nivel. Aumentará el ratio de aparición de enemigos, y se comenzarán a generar enemigos del último de los tipos: los enemigos oscuros.
- **Zona 6:** para finalizar, el último nivel, la Zona 6, será el desafío final, con todo lo que tienen por ofrecer los enemigos. Éstos serán más rápidos, y prácticamente todos los enemigos tendrán algún tipo de capacidad especial, teniendo el jugador que vencer oleadas y oleadas de veloces y potentes enemigos; además, es el nivel más largo. Tras superarlo, el jugador será recompensado con la pantalla de victoria.

4.5. Diseño de la interfaz de usuario

En este apartado, hablaremos del diseño tras la interfaz de usuario de OverColor; específicamente, nos centraremos tanto en cómo se comunica el jugador con el videojuego, así como en el Head-Up Display (HUD).

4.5.1. Comunicación jugador-videojuego

OverColor es un videojuego desarrollado principalmente para dispositivos móviles multitáctiles, por lo que todo lo que el jugador quiera transmitir al juego deberá hacerse a través de la pantalla táctil de su dispositivo.

Para el movimiento de la protagonista entre las líneas, se deberá tocar la zona izquierda de la pantalla; el avatar del jugador se colocará en la línea más cercana al toque en todo momento, permitiendo que el cambio entre líneas sea lo más rápido y fluido posible.

Para los ataques, tal y como ya fue explicado, se hará uso de los botones de la pantalla. Cuando estos estén en enfriamiento, aparecerán grisáceos y desactivados, sugiriendo que el jugador debe esperar hasta poder volver a usarlos.

La navegación entre menús también se hará pulsando botones, lo que incluye la mejora de edificios.

A efectos prácticos, los controles del juego son muy simples y accesibles, y la dificultad del juego radica en la gestión de enfriamientos y capacidad de tomar decisiones bajo presión.

4.5.2. Head-Up Display

Durante el *gameplay*, el jugador dispondrá de un HUD con diversa información útil que le ayudará a conocer su estado en la partida y le permitirá tomar las decisiones correctas. En el HUD se muestra la siguiente información:

- Puntos de vida del jugador, de forma que sepa en todo momento cuantos golpes más puede recibir antes de morir.
- Vidas extra del jugador, para controlar cuantas veces más se puede morir antes de perder la run.
- Tiempo de juego en forma de contador, que indica cuantos segundos han pasado desde el comienzo de la run.
- Cantidad de divisas que posee el jugador (roja, azul y amarilla), las cuales se actualizan en tiempo real cada vez que se vence a un enemigo.
- Botones de ataque (rojo, azul y amarillo), los cuales se desactivan mientras están en enfriamiento.

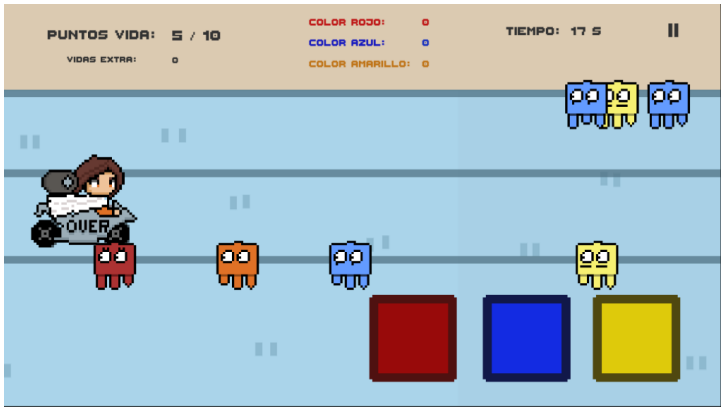


Figura 29. El HUD durante el gameplay

Por otro lado, el HUD será diferente cuando nos encontremos en la villa, ya que allí elementos como la vida o el tiempo no tienen relevancia. En la villa principalmente tendremos las divisas, los edificios, y el botón para comenzar a jugar. Además, dentro de los propios edificios, las interfaces y menús tratarán de ser claras, mostrando los beneficios que ofrecen los edificios y sus costes jugando con los colores al mismo tiempo que son explicados mediante texto.

Figura 30. El HUD en la villa

Figura 31. Menú de mejora de edificio

4.6. Esquema de arquitectura del juego

A continuación, se presenta un diagrama con el flujo de las escenas en el juego. Desde la escena de inicio se irá a la villa, que hará de menú principal del juego en donde mejorar los edificios o comenzar a jugar.

Mientras se esté desarrollando el *gameplay*, toda la acción estará en la misma escena, ya que los cambios se provocan en la generación de enemigos. Según si el jugador es derrotado o consigue completar el ascenso, irá a una escena diferente, desde la que podrá o volver a jugar o regresar a la Villa para gastar su divisa obtenida.

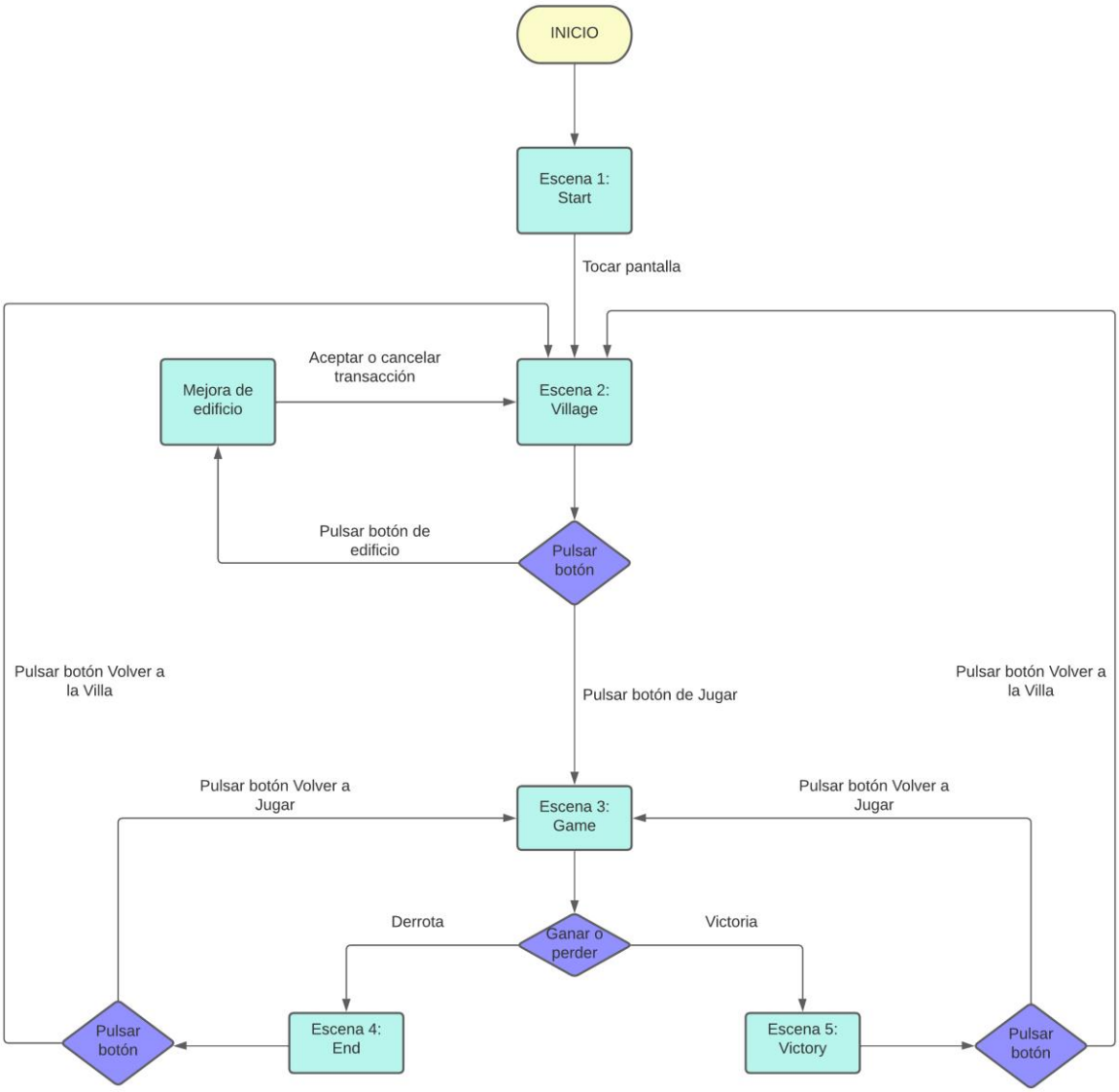


Figura 32. Diagrama de flujo

5. Implementación

En este apartado se hará una breve explicación sobre los diferentes scripts de código que conforman el proyecto y su funcionalidad, así como una breve guía sobre como instalarlo y comenzar a jugar.

5.1. Scripts implementados

Tal y como se explicó con anterioridad, OverColor fue desarrollado en el motor Unity, y los scripts construidos están en el lenguaje C#. Según su funcionalidad, serán clasificados en categorías.

En primer lugar, se listarán los scripts relacionados con los edificios de la villa y su funcionalidad:

- Aeropuerto: clase que hereda de Building e inicializa el edificio con los parámetros y funcionalidad asociados a la mejora del edificio Aeropuerto (color morado, otorga vidas extra).
- Bosque: clase que hereda de Building e inicializa el edificio con los parámetros y funcionalidad asociados a la mejora del edificio Bosque (color verde, otorga mayor probabilidad de encontrar objetos raros).
- Building: clase que ejerce de base para los edificios de la villa. Todos los edificios heredan de ella y cuenta con diversos métodos virtuales que han de ser sobrescritos por las clases de los edificios específicos.
- BuildingUnlocker: clase que gestiona el proceso de desbloquear nuevos edificios y mostrárselos al jugador. Fuertemente arraigada a la clase Carretera.
- BuildMenuButtonController: clase que controla el comportamiento de los botones de la villa; estos son, tanto los botones para mejorar un edificio o cancelar el proceso de mejora, así como el propio botón de jugar para comenzar una run.
- Carretera: clase que hereda de Building e inicializa el edificio con los parámetros y funcionalidad asociados a la mejora del edificio Carretera (color azul, desbloquea nuevos edificios).
- Casa: clase que hereda de Building e inicializa el edificio con los parámetros y funcionalidad asociados a la mejora del edificio Casa (color rojo, otorga puntos de vida extras).

- Mercado: clase que hereda de Building e inicializa el edificio con los parámetros y funcionalidad asociados a la mejora del edificio Mercado (color naranja, otorga mejoras iniciales extra).
- Taller: clase que hereda de Building e inicializa el edificio con los parámetros y funcionalidad asociados a la mejora del edificio Taller (color amarillo, otorga reducción de enfriamiento de los botones extra).

Tras esto, serán listados los scripts asociados al funcionamiento de los diferentes tipos de enemigos, cualidades específicas de los mismos, y su generación:

- BulletController: clase que controla el movimiento, funcionamiento e impacto de las balas disparadas por los enemigos tiradores.
- DoubleColoredEnemy: clase que hereda de EnemyController y sobrescribe varios de los funcionamientos básicos de los enemigos para adaptarse al funcionamiento de un enemigo de color doble (verde, morado, naranja) o de un enemigo de color oscuro; especialmente, gestiona la generación de un enemigo del subtipo correspondiente tras la muerte de este.
- EnemyController: clase principal de los enemigos que gestiona toda su funcionalidad, desde su movimiento e impacto contra el jugador, hasta su supervivencia y muerte a manos del mismo, pasando por sus *power-ups*.
- EnemyExplosion: clase que presentan los enemigos explosivos que controla el proceso de explosión tras la muerte de estos.
- EnemyGenerator: clase que controla la generación y aparición de enemigos en la escena. Cuenta además con las funciones necesarias para dotar a dichos enemigos de sus cualidades especiales, como *power-ups* o colores dobles.
- JumperEnemy: clase que presentan únicamente los enemigos saltarines y que gestiona la temporización y el proceso de saltar de una línea a otra.
- ShooterEnemy: clase exclusiva de los enemigos tiradores que controla el proceso de disparar balas y de mantener las distancias con el jugador.

Otro subgrupo a diferenciar es el del par de clases encargadas de la permanencia y guardado de datos del juego y progreso del jugador en los sistemas de la villa y obtención de divisa:

- GameData: clase serializable en la que existe una variable serializable por cada parámetro del juego que se quiera almacenar de forma permanente en el sistema. Estos parámetros serán:

    - Vida máxima.
    - Reducción de enfriamiento.
    - Probabilidad de obtener objetos raros.
    - Número de objetos iniciales.
    - Vidas extra.
    - Número de edificios desbloqueados.
    - Número de victorias.
    - Nivel del taller.
    - Nivel de la casa.
    - Nivel de las carreteras.
    - Nivel del bosque.
    - Nivel del mercado.
    - Nivel del aeropuerto.
    - Cantidad de divisa roja.
    - Cantidad de divisa azul.
    - Cantidad de divisa amarilla.

- SaveSystem: clase que gestiona tanto la carga como guardado de datos haciendo uso de instancias de la clase GameData. Para ello, se utilizarán archivos del tipo binario llamados “overcolor.data”.

En cuarto lugar, se hablará sobre las clases estáticas que usa el proyecto para mantener información en instancias únicas que permanecen entre escenas:

- BuildingLevels: clase en la que se almacenan los niveles a los que se encuentra cada edificio de la villa.
- Currencies: clase que almacena la cantidad de divisa de cada tipo que posee el jugador.
- EndValues: clase que almacena el tiempo que sobrevivió el jugador durante la run de forma previa a mostrarlo en la pantalla de derrota.

- PlayerStats: clase en la que se almacenan los valores del personaje como son su vida máxima o reducción de enfriamiento de los botones entre otros.

En quinto lugar, se listarán los scripts relacionados con las Mejoras obtenibles durante una run y su control y generación. En esta categoría encontraremos la mayor cantidad de scripts, ya que cada Mejora posee su propia clase:

- BlueMultiplierUpgrade: clase que controla el funcionamiento del Multiplicador Azul.
- RedMultiplierUpgrade: clase que controla el funcionamiento del Multiplicador Rojo.
- YellowMultiplierUpgrade: clase que controla el funcionamiento del Multiplicador Amarillo.
- BluePulseUpgrade: clase que gestiona el comportamiento de la Máquina de pulsos azules.
- BrokenPulseUpgrade: clase que gestiona el comportamiento de la Máquina de pulsos rota.
- GreenPulseUpgrade: clase que gestiona el comportamiento de la Máquina de pulsos verdes.
- OrangePulseUpgrade: clase que gestiona el comportamiento de la Máquina de pulsos naranjas.
- PurplePulseUpgrade: clase que gestiona el comportamiento de la Máquina de pulsos morados.
- RedPulseUpgrade: clase que gestiona el comportamiento de la Máquina de pulsos rojos.
- YellowPulseUpgrade: clase que gestiona el comportamiento de la Máquina de pulsos amarillos.
- BlueTurretUpgrade: clase que maneja el funcionamiento de la Torreta Azul.
- GreenTurretUpgrade: clase que maneja el funcionamiento de la Torreta Verde.
- OrangeTurretUpgrade: clase que maneja el funcionamiento de la Torreta Naranja.
- PurpleTurretUpgrade: clase que maneja el funcionamiento de la Torreta Morada.
- RedTurretUpgrade: clase que maneja el funcionamiento de la Torreta Roja.
- TurretController: clase controladora del comportamiento genérico de las torretas. Aun cuando se consigan varias Mejoras del tipo torreta, no aparecerán varias torretas, sino que habrá una única que se irá haciendo más poderosa. Esto último es gestionado por esta clase.

- YellowTurretUpgrade: clase que maneja el funcionamiento de la Torreta Amarilla.
- BatteryUpgrade: clase que controla el funcionamiento de las Baterías nuevas.
- BotiquinUpgrade: clase que controla el funcionamiento del Botiquin.
- BoxUpgrade: clase que controla el funcionamiento de La Caja.
- BuscacorazonesUpgrade: clase que controla el funcionamiento del Buscacorazones.
- ColorBombUpgrade: clase que controla el funcionamiento de la Bomba de Color.
- LoveUpgrade: clase que controla el funcionamiento de Amor.
- BatteryUpgrade: clase que controla el funcionamiento de las Baterías nuevas.
- MirrorUpgrade: clase que controla el funcionamiento del Espejo.
- TrebolUpgrade: clase que controla el funcionamiento del Trébol de Cuatro Hojas.
- TurboUpgrade: clase que controla el funcionamiento de Turbo.
- UltraBatteryUpgrade: clase que controla el funcionamiento de las Ultra Baterías nuevas.
- UltraLoveUpgrade: clase que controla el funcionamiento de Ultra Amor.
- Upgrade: clase que ejerce como base para todas las Mejoras. Cualquier clase que represente una mejora hereda de Upgrade.
- UpgradeButtonController: clase que gestiona el comportamiento de los botones del menú que aparece cuando es momento de escoger una mejora, asegurándose de que la mejora escogida sea la única que se aplique.
- UpgradeGenerator: clase encargada de controlar la generación de Mejoras para que el jugador escoja, así como de eliminar de la *pool* de Mejoras aquellas que ya fueron escogidas.

En sexto lugar, existirá una pequeña categoría en la que tan solo se agruparán dos scripts relacionados con las zonas que constituyen a una run:

- Zone: clase en la que se almacenan los parámetros fundamentales que definen una zona.
- ZoneManager: clase que controla la duración de una zona, la gestión de sus fases, y la transición a las siguientes zonas, manteniendo siempre control sobre cual es la zona actual y cuales son sus ratios y cualidades.

Para finalizar, en séptimo y último lugar, agruparemos los scripts que no encontraron un lugar en ninguna de las clasificaciones anteriores:

- AudioDestroyer: clase que se encarga de destruir al objeto en la que esté instanciada al terminar de reproducir un audio.
- BackgroundController: clase que gestiona el movimiento de los fondos que mueven una textura indefinidamente.
- BombController: clase que gestiona el funcionamiento del botón extra que aparece al obtener la Mejora Bomba de Color.
- ColorButtonController: clase que controla la activación y enfriamiento de los botones de colores.
- EndButtonsController: clase con la funcionalidad de los botones de la pantalla de derrota.
- EndManager: clase que controla el comportamiento de la escena de derrota, asegurándose de que todo se reproduzca de forma temporalmente correcta.
- FadingController: clase encargada de destruir las imágenes translúcidas que dejan los enemigos saltarines.
- GameMusicManager: clase encargada de cambiar de una canción a otra durante el gameplay y de pausar o reanudar las canciones cuando sea necesario.
- GameplayManager: clase de gran importancia que gestiona el flujo del juego. Aquí se guardarán diversas variables con las estadísticas del jugador, si bien teniendo además en cuenta los cambios provocados por la obtención de Mejoras. Además, desde aquí se controlan los enemigos que hay en cada línea y su destrucción mediante ataques a líneas específicas o ataques de pulso. Esta clase también controla las situaciones en las que el jugador recibe daño o una curación, así como determina si el jugador ha sido derrotado en base a su vida.
- HeartController: clase que controla el movimiento e impacto de los corazones dejados por enemigos.
- MirrorController: clase que controla el comportamiento del clon de Roloc que aparece tras obtener la Mejora Espejo.
- PauseController: clase que controla el proceso de pausar el juego y el funcionamiento de los botones del menú de pausa.
- RayController: clase que controla la autodestrucción de los *sprites* de los rayos dejados por los ataques del jugador.
- RolocController: clase que controla el cambio entre líneas del avatar del jugador al tocar en la pantalla.

- SelfDestructController: clase que controla la autodestrucción de un objeto cualquiera en base a un tiempo predefinido de vida.
- StartScreenManager: clase que gestiona la funcionalidad de la pantalla de inicio.
- VictoryManager: clase que gestiona la funcionalidad y los botones de la pantalla de victoria.

5.2. Requisitos e instrucciones de instalación

OverColor es un juego desarrollado para dispositivos móviles multitáctiles, específicamente hablando, dispositivos Android. Cualquier dispositivo móvil Android con tecnología multitáctil debería ser capaz de poder correr OverColor sin problemas.

En cuanto a las resoluciones de pantalla aceptadas, se ha realizado el mayor esfuerzo posible para que OverColor pueda adaptarse prácticamente a cualquier pantalla. Sin embargo, los recursos disponibles para poder probar el resultado final en múltiples dispositivos son muy limitados, por lo que tan solo se han realizado pruebas en algunos dispositivos.

Dentro del entorno Android, la instalación de OverColor es muy sencilla y similar a la instalación de cualquier otra aplicación proveniente de un archivo del tipo Android Application Package (APK). Es importante tener en cuenta que la mayoría de dispositivos móviles no permiten la instalación de archivos APK de forma predeterminada para aumentar la seguridad y evitar la instalación accidental de software malicioso por parte de usuarios más inexpertos, por lo que será necesario asegurar que el dispositivo móvil en cuestión en el que se quiera instalar OverColor esté configurado para poder instalar archivos APK. Este proceso de configuración varía entre dispositivo y dispositivo, por lo que se recomienda revisar en la red sobre más información acerca del dispositivo específico que vaya a usarse.

Por otro lado, si lo que se desea no es instalar OverColor en sí, sino el proyecto de Unity, el proceso a tener en cuenta es diferente. La versión de Unity utilizada es la 2020.3.28f1, por lo que será necesario disponer de un dispositivo compatible con dicha versión (para más información, se recomienda revisar el apartado *4.1. Entorno de desarrollo final* de este documento).

Una vez se disponga de un equipo con Unity instalado en la versión correspondiente, será necesario descargar el proyecto desde el repositorio, ya sea exportándolo como un archivo comprimido, o clonando el repositorio en el equipo en cuestión. Tras esto, podrá abrirse con Unity como cualquier otro proyecto.

De forma adicional, si se quiere ejecutar el proyecto dentro de Unity, será necesario conectar un dispositivo móvil Android al equipo mediante una conexión USB, y se deberá descargar en el dispositivo Android la aplicación oficial Unity Remote 5 [32], lo que permitirá visualizar la ejecución del juego directamente en el dispositivo móvil. Considerando que el control del juego es completamente táctil, sería imposible jugar sin un dispositivo móvil.

6. Demostración

OverColor es un juego simple y orientado a un público amplio, por lo que no cuenta con sistemas demasiado complejos, ni con manuales de instrucciones ni tutoriales.

El juego desarrollado como parte del presente Trabajo de Fin de Grado está completo y funcional, por lo que no serán adjuntados prototipos adicionales, ya que el propio repositorio cuenta con la versión completa.

En cuanto a la realización de pruebas, se escogió a un conjunto de personas dispuestas a probar el proyecto y se les permitió jugar a OverColor, o bien en sus propios dispositivos móviles, o bien en uno previamente preparado.

La mitad de los participantes probaron el juego tras recibir una explicación previa del mismo: partían con el conocimiento de que para vencer a los enemigos debían golpear a los enemigos con su propio color, que los enemigos de colores combinados como el verde debían ser golpeados por diferentes ataques, el funcionamiento de las divisas y los sistemas de progresión, y la existencia de las Mejoras. Al segundo grupo se le asignó jugar al juego sin ningún tipo de explicación previa, si bien seguían bajo supervisión durante su tiempo de juego.

Dentro de ambos grupos se contaba tanto con público muy casual y poco adherido al mundo de los videojuegos, como con jugadores consistentes con gran experiencia previa en este medio. En general, el juego parecía entenderse fácil aun sin explicaciones previas, si bien es cierto que algunos jugadores del segundo grupo no entendían bien cómo vencer a los enemigos en primera instancia hasta hacer un poco de prueba y error.

Desgraciadamente, debido a la limitación de recursos del proyecto, el número de personas con las que se realizaron las pruebas fue relativamente pequeño, por lo que es posible que los resultados obtenidos requiriesen de un entorno más grande.

7. Conclusiones y líneas de futuro

Para finalizar el documento, se presentarán las conclusiones obtenidas de la realización del proyecto, y se hablará sobre posibles mejoras futuras a implementar.

7.1. Conclusiones

La realización del presente Trabajo de Fin de Máster fue una experiencia interesante, didáctica y enriquecedora. Durante el desarrollo del proyecto, se ha aprendido a decidir las características a implementar, de forma realista, y a planificar y posteriormente construir dichas mecánicas ideadas, investigando sobre cómo fue el acercamiento al género a tratar (*roguelite*) por parte de otros juegos exitosos que viniesen antes, y se trató de darle un enfoque personal.

Se considera que todos los objetivos ideados al comienzo del proyecto (reflejados en el apartado *1.3. Objetivos generales* de este documento) fueron mayoritariamente cumplidos.

Desde el punto de vista de los objetivos primarios viendo el proyecto como “producto o servicio”, se tiene esperanza en que el juego desarrollado sea divertido para el jugador, y en un principio es fácil de jugar, requiriendo de no demasiada habilidad mecánica. Debido a que fue desarrollado para dispositivos móviles, debería poder ser accesible sin requerir de una consola específica o de un ordenador de altas capacidades.

Desde el punto de vista de los objetivos primarios desde el propio beneficio del desarrollador, gracias al proyecto se aprendió mucho sobre el desarrollo para juegos móviles, sobre sus limitaciones y beneficios, sobre sus controles e interfaz de usuario, así como sobre cómo probarlos y configurarlos. Se considera que el proyecto, si bien aun tiene un gran margen de mejora y posibilidades de futuro, es un proyecto lo suficientemente sólido como para mostrar en un portfolio, y se considera que, aun cuando los recursos temporales eran limitados, se construyó una experiencia satisfactoria y completa.

Desde el punto de vista de los objetivos secundarios, se considera que el trabajo en el entorno Unity ayudó aumentar la maestría y experiencia con dicho motor, así como se pudieron poner en práctica los conocimientos adquiridos en las diferentes asignaturas cursadas.

Por último, se hablará de la planificación y su seguimiento a lo largo del desarrollo del proyecto. En una primera instancia se creó una planificación inicial, y se decidió hacer el seguimiento mediante el uso de metodologías ágiles, especialmente haciendo uso de Kanban. Si bien al principio se consideró que probablemente conforme el proyecto avanzase la planificación variaría enormemente, no fue el caso, y verdaderamente se mantuvo robusta y sólida durante la mayor parte del proyecto. En general, el uso de Kanban fue extremadamente exitoso, permitiendo controlar y gestionar fácilmente qué tareas quedaban por realizar, cuales requerían de revisión, que bugs habían aparecido y requerían de una solución, etc. Sin embargo, si bien las fechas reales y las iniciales no coinciden exactamente al cien por ciento, por lo general son mayormente parecidas, especialmente reflejado en las diferentes PECs a entregar como parte del desarrollo del proyecto. En ese sentido, la propia "planificación base" propuesta por el profesorado a modo de entregables guio la propia planificación creada para el proyecto, y gracias a esta se debe el éxito del mismo.

Para cerrar el apartado, diré que estoy altamente satisfecho con el proyecto realizado, y extremadamente agradecido por haber tenido la oportunidad de trabajar en este entorno construyendo mi propio producto en base a mi propia visión de este.

7.2. Líneas de futuro

Si bien se considera que OverColor cumple con lo esperado, existe un amplio margen de mejora, y existen ya un gran conjunto de ideas y planes de mejora para ampliar el contenido del mismo y hacerlo mucho más interesante, divertido y rejugable.

De forma principal, una de las ideas a implementar es la existencia de jefes. Actualmente, el cambio entre niveles es marcado por la obtención de una Mejora, pero sería mucho más interesante si además hubiese que enfrentar a un jefe. Dichos jefes podrían seleccionarse aleatoriamente de una pool, y deberían tener mecánicas interesantes, haciendo que vencerlos fuese casi como un puzle. Por ejemplo, un jefe podría iluminarse siguiendo un patrón de colores, y el jugador debería memorizarlo y atacarle siguiendo dicho orden para poder hacerle daño.

Además, sería ideal añadir aún más Mejoras a la *pool* del juego. Actualmente OverColor cuenta con alrededor de treinta, pero estás son las que hacen con diferencia que las *runs* se sientan diferentes unas de otras, por lo que sería muy interesante expandir el catálogo de

opciones. De igual forma, sería altamente interesante añadir sinergias entre Mejoras. Por ejemplo, en su versión actual, obtener la Mejora Turbo tras haber obtenido mejoras de reducción de enfriamiento aumenta la vida de Roloc de forma adicional. Añadir más sinergias y compatibilidades como la anterior podría sorprender a los jugadores, los cuales se verían recompensados por experimentar con nuevas opciones.

El apartado artístico también podría mejorarse en el futuro. Se podrían crear *sprites* y animaciones más pulidos y complejos, así como podrían componerse bandas sonoras originales para el juego, en lugar de utilizar canciones preexistentes de uso libre.

Por último, también sería extremadamente interesante ofrecer una versión del juego para dispositivos iOS multitáctiles, permitiendo al máximo de gente posible disfrutar de OverColor.

Bibliografía

[1] Video game market revenue worldwide in 2021, by segment(in billion U.S. dollars). Statista [En línea: https://www.statista.com/statistics/292751/mobile-gaming-revenue-worldwide-device/] (último acceso: 28 de febrero de 2022).

[2] Glenn Wichman (1980). *Rogue* [Video game].

[3] Josh Bycer (2019). *The Roguelike Debate – Roguelikes vs Roguelites, Game Developer* [En línea: https://www.gamedeveloper.com/design/the-roguelike-debate----roguelikes-vs-roguelites] (último acceso: 28 de marzo de 2022).

[4] Supergiants Games (2018). *Hades* [Video game].

[5] Google Play Store [En línea: https://play.google.com/store/apps?gl=ES] (último acceso: 29 de marzo de 2022).

[6] Tako Boy Studios (2018). *Candies 'n Curses* [Video game].

[7] Edmun McMillen (2011). *The Binding of Isaac* [Video game].

[8] Supercell (2016*). Clash Royale* [Video game].

[9] Delightworks, Lasengle (2015). *Fate/Grand Order* [Video game].

[10] Unity Technologies. Unity [En línea: https://unity.com/es] (último acceso: 28 de marzo de 2022).

[11] Epic Games. Unreal Engine [En línea: https://www.unrealengine.com/en-US/] (último acceso: 28 de marzo de 2022).

[12] Mark Overmars, YoYo Games. GameMaker Studio [En línea: https://www.yoyogames.com/es/gamemaker] (último acceso: 28 de marzo de 2022).

[13] Studio MDHR (2017). *Cuphead* [Video game].

[14] Team Cherry (2017). *Hollow Knight* [Video game].

[15] Blizzard Entertainment (2014). *Hearthstone* [Video game].

[16] Moon Studios (2015). *Ori and the Blind Forest* [Video game].

[17] Unity Technologies. Unity Asset Store [En línea: https://assetstore.unity.com/] (último acceso: 28 de marzo de 2022).

[18] 2K Games, Irrational Games (2007). *Bioshock* [Video game].

[19] Gearbox Software (2009). *Borderlands* [Video game].

[20] Epic Games (2017). *Fortnite* [Video game].

[21] David Capello. Aseprite [En línea: https://www.aseprite.org/] (último acceso: 28 de marzo de 2022).

[22] GIMP [En línea: https://www.gimp.org/] (último acceso: 28 de marzo de 2022).
[23] Unity Technologies. Unity 2020.3.28 What's new [En línea: https://unity3d.com/es/unity/whats-new/2020.3.28] (último acceso: 7 de mayo de 2022).
[24] Unity Technologies. System requirements for Unity 2020.1 [En línea: https://docs.unity3d.com/2020.1/Documentation/Manual/system-requirements.html] (último acceso: 7 de mayo de 2022).
[25] Visual Studio [En línea: https://visualstudio.microsoft.com/es/] (último acceso: 7 de mayo de 2022).
[26] GitLab [En línea: https://about.gitlab.com/] (último acceso: 7 de mayo de 2022).
[27] SourceTree [En línea: https://www.sourcetreeapp.com/] (último acceso: 7 de mayo de 2022).
[28] Trello [En línea: https://trello.com/es] (último acceso: 7 de mayo de 2022).
[29] Free Stock Music [En línea: https://www.free-stock-music.com/] (último acceso: 21 de mayo de 2022).
[30] Mixkit [En línea: https://mixkit.co/] (último acceso: 21 de mayo de 2022).
[31] Unity Asset Store. Free Pixel Font – Thaleah [En línea: https://assetstore.unity.com/packages/2d/fonts/free-pixel-font-thaleah-140059#description] (último acceso: 21 de mayo de 2022).
[32] Google Play. Unity Remote 5 [En línea: https://play.google.com/store/apps/details?id=com.unity3d.mobileremote&hl=es_419&gl=US] (último acceso: 21 de mayo de 2022).