UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Gandia

PIUM: Producción de un videojuego multijugador basado en niveles generados de forma procedural con gestión de puntuaciones en red.

Trabajo Fin de Grado

Grado en Tecnologías Interactivas

AUTOR/A: Maldonado Llambies, Adrian

Tutor/a: Pérez López, David Clemente

CURSO ACADÉMICO: 2022/2023

Resumen:

Este trabajo presenta la producción de un videojuego multijugador desde la fase de diseño, planificación, preproducción, implementación hasta las pruebas. La temática se centra en PIUM, una patrulla intergaláctica de unidad y misterio cuyo objetivo es investigar un planeta desconocido. Se trata de un juego *roguelite* de acción en perspectiva isométrica donde se controla una nave. Dicho videojuego consta de un sistema de niveles generados de forma procedural donde se garantiza la posibilidad de acceder a todas las partes del mapa. Además, se trata de un juego multijugador donde se puede visualizar e interactuar en tiempo real con el aliado, mediante un sistema cooperativo en red. Asimismo, se implementa una gestión de puntuaciones en remoto, con un servicio web para almacenamiento de resultados en base de datos, así como una página web para mostrar un ranking con las mejores puntuaciones de los jugadores.

Resumen (inglés):

This work presents the production of a multiplayer video game from the design, planning, pre-production, implementation, and testing phases. The theme focuses on PIUM, an intergalactic patrol of unity and mystery whose objective is to investigate an unknown planet. It is a *roguelite* action game in an isometric perspective where a spaceship is controlled. The video game consists of a system of procedurally generated levels where access to all parts of the map is guaranteed. Additionally, it is a multiplayer game where players can visualize and interact with allies in real-time through a cooperative network system. Furthermore, a remote scoring management system is implemented, with a web service for result storage in a database, as well as a website to display a ranking with the best player scores.

Palabras clave: Videojuego, procedural, multijugador, ranking, web

Palabras clave (inglés): videogame, procedural, multiplayer, ranking, web

Índice

1. Introducción

En los últimos años la industria de los videojuegos ha experimentado un gran crecimiento (Orús, 2023). Como consecuencia cada vez hay más desarrolladores aventurándose en la creación de cada vez más novedosos y arriesgados títulos. En este contexto, el presente trabajo de fin de grado tiene como objetivo la producción de un videojuego multijugador basado en niveles generados de forma procedural con gestión de puntuaciones en red.

El videojuego, llamado PIUM (Patrulla Intergaláctica de Unidad y Misterio), se enmarca dentro del género *roguelite* de acción, en perspectiva isométrica, donde el jugador controla una nave de patrulla intergaláctica en una misión de exploración en un planeta desconocido.

Los *roguelite* son un género de los videojuegos que derivan de los *roguelike*. El término *roguelike* nace de Rogue (1980), uno de los primeros videojuegos que dieron vida a este sistema de juego pese a que no fuera el primero (Craddock, 2021). La característica principal de estos videojuegos es que sus mapas se van creando de forma aleatoria, es decir, siguiendo un algoritmo (procedural) que va creando habitaciones de manera que nunca sabemos qué enemigos van a estar presentes ni podemos seguir un destino marcado. Simplemente, cada partida es una nueva aventura a superar, conscientes de los grandes riesgos y objetivos, pero sin saber lo que podemos esperar. Además, en este género, los juegos tienen que estar divididos por mazmorras con sistema de casillas. Por otra parte, deben contar con un sistema por turnos, tener un control de recursos e incluso presentar la llamada “permamuerte”, de tal forma que el jugador, si su personaje fallece, tendrá que dar comienzo a su aventura de nuevo.

El género *roguelite* deriva de *roguelike*, donde lite vendría a traducirse como “ligero”, es decir, engloba a los videojuegos que cuentan con las principales características de *roguelike* pero sin llegar a cumplir con todas. Aunque pueda compartir elementos como las mazmorras creadas de forma aleatoria, existen diferencias como una muerte que, aunque permanente, haga iniciar el camino con un personaje que ya incluye ciertas mejoras, no como si se reiniciara la aventura de cero.

En el presente trabajo se describe la fase de producción, explicando la parte de diseño, planificación e implementación hasta la fase de pruebas. También, se expondrán y explicarán las metodologías utilizadas, los desafíos a los que se les ha hecho frente y los resultados finales.

1.1. Preámbulo

Según el informe de Newzoo sobre la industria de los videojuegos de 2021, la industria de los videojuegos estaba creciendo exponencialmente en los últimos años. Se estimó que el mercado mundial de videojuegos generaría ingresos de más de 180 mil millones de dólares en 2021 y que el número de jugadores a nivel global superaría los 3 mil millones (Wijman, 2021). Para 2023, se estima que el valor del mercado supere los 250 mil millones de dólares (Orús, 2023).

El juego objeto de este trabajo se enfoca en un elemento importante en la industria de los videojuegos: la comunidad. La inclusión de un sistema multijugador y de gestión de puntuaciones en red permite que los jugadores puedan competir y comparar sus habilidades con otros jugadores de todo el mundo.

En este sentido, el desarrollo del videojuego se enmarca dentro de una industria en constante evolución y crecimiento, y busca contribuir a la misma mediante la producción de un producto innovador y atractivo para los usuarios.

Por otra parte, el desarrollo de este trabajo es una oportunidad para conocer sistemas de desarrollo de videojuegos multijugador, desde los mecanismos y técnicas necesarias para crear experiencias de juego compartidas entre múltiples jugadores, la sincronización de acciones, hasta la gestión de la

conectividad. También, el trabajo ofrece la posibilidad de investigar en el campo de los algoritmos de generación de niveles de juego de forma dinámica y aleatoria. Además, es una forma de explorar sistemas web de gestión y almacenamiento de puntuaciones, donde los jugadores registran y comparten sus estadísticas en línea, fomentando la competencia y la interacción social. Todo ello, sin perder de vista la integración conjunta de todos estos elementos, bajo el paraguas de un entorno de desarrollo de videojuegos.

Por último, también es una oportunidad de descubrir algunos de los videojuegos históricos y actuales más relevantes, relacionados con la temática del trabajo.

1.2. Objetivos

El objetivo principal de este trabajo es producir un videojuego multijugador basado en niveles generados de forma procedimental con gestión de puntuaciones en red.

Para alcanzar este objetivo principal, se plantean los siguientes objetivos específicos:

- Realizar una revisión teórica sobre plataformas para la creación de videojuegos, herramientas disponibles para crear videojuegos multijugador, así como sistemas para almacenaje y gestión de puntuaciones en sistemas colaborativos.
- Estudiar algoritmos que permitan generar niveles de forma automática y aleatoria.
- Analizar videojuegos existentes para obtener conocimientos y perspectivas que ayuden a mejorar nuestro propio diseño y desarrollo del juego, identificando fortalezas y áreas de oportunidad para ofrecer una experiencia de juego más enriquecedora y atractiva para los jugadores.
- Diseñar y planificar las fases de desarrollo del videojuego, desde la definición de la temática, la mecánica de juego hasta la identificación de los recursos necesarios.
- Implementar tanto el videojuego como el sistema de gestión de puntuaciones en red, haciendo un uso adecuado de las herramientas, conceptos y tecnologías estudiadas.
- Evaluar el videojuego mediante pruebas con usuarios para valorar su jugabilidad, nivel de rejuego y experiencia de juego.

Con estos objetivos, se busca no solo producir un videojuego de alta calidad, sino también adquirir habilidades y conocimientos sobre el proceso de producción de un videojuego multijugador, desde la planificación y diseño, hasta la implementación y evaluación del mismo. Además, se pretende contribuir al campo de los videojuegos mediante la producción de un juego innovador y atractivo para los usuarios.

1.3. Metodología

Para la realización del presente TFG se ha empleado la metodología iterativa o incremental, la cual centra el desarrollo en la creación y mejora progresiva del videojuego a través de ciclos repetitivos de trabajo. En lugar de intentar desarrollar todo el juego de una sola vez, la metodología iterativa permite construir el juego en etapas, donde cada etapa se basa en la retroalimentación y los aprendizajes obtenidos de la etapa anterior.

El proceso generalmente comienza con una versión básica o prototipo mínimo viable (MVP) del juego, que incluye los elementos esenciales y características principales (Ries, 2011). A medida que se obtiene retroalimentación de los jugadores o del equipo de desarrollo, se realizan iteraciones adicionales para mejorar y expandir el juego. Cada iteración puede incluir la adición de nuevas características, mejoras en la jugabilidad, ajustes en el equilibrio del juego, correcciones de errores y optimizaciones de

rendimiento. Después de cada iteración, se realizan pruebas exhaustivas para evaluar el juego y recopilar comentarios adicionales.

Este enfoque iterativo permite adaptarse a medida que el juego evoluciona, corrigiendo problemas y aprovechando nuevas oportunidades a lo largo del proceso de desarrollo. También facilita la entrega de un producto funcional en etapas tempranas, lo que puede ser beneficioso para obtener financiamiento, involucrar a la comunidad de jugadores y mantener un flujo constante de retroalimentación para mejorar el juego.

Asimismo, la metodología de desarrollo de videojuegos iterativa permite una producción más ágil y flexible del videojuego, ya que permite la adaptación a los cambios y ajustes necesarios durante el proceso de producción. Además, facilita la identificación de problemas y la toma de decisiones, al permitir la evaluación continua del juego en desarrollo.

Las iteraciones estaban divididas en cuatro grandes bloques: Juego base, multijugador, sistema de clasificación y sonido.

Por otra parte, se ha empleado Trello como herramienta para gestionar las tareas a realizar en el proyecto y Git como sistema para gestionar el código que se ha generado.

Trello ha sido utilizado como una plataforma de gestión de proyectos basada en tableros, que nos ha permitido organizar y supervisar de manera eficiente las diferentes etapas del desarrollo del videojuego. Mediante la creación de tarjetas y listas, hemos podido asignar tareas, establecer fechas límite y realizar seguimiento del progreso del desarrollo.

Además, hemos utilizado Git como sistema de control de versiones para gestionar el código fuente del videojuego. Gracias a Git, hemos podido mantener un historial completo de las modificaciones realizadas en el código, asegurando la integridad y disponibilidad de las diferentes versiones del proyecto.

La combinación de Trello y Git ha demostrado ser una solución eficaz para gestionar tanto las tareas del proyecto como el control del código generado. Estas herramientas nos han brindado una visión clara del progreso del proyecto, facilitando la organización en el desarrollo, y garantizando la integridad del código en todo momento.

1.4. Fases

El desarrollo del videojuego se ha dividido en las siguientes fases:

1. Fase de investigación: En esta fase se realiza una revisión bibliográfica exhaustiva sobre los conceptos y tecnologías necesarias para la producción del videojuego. Además, se lleva a cabo un análisis de los videojuegos similares y las tendencias actuales en la industria de los videojuegos.
2. Fase de diseño y planificación: En esta fase se define la temática y mecánicas de juego del videojuego. Se identifican los recursos necesarios, tales como herramientas y tecnologías para su implementación, y se planifican las diferentes iteraciones necesarias para su desarrollo (juego base, multijugador, sistema de clasificación y sonido).
3. Fase de desarrollo e implementación: En esta fase se desarrolla e implementa el videojuego. Se llevan a cabo diferentes iteraciones, cada una con objetivos específicos y metas claras, hasta llegar a una versión final del videojuego.
4. Fase de evaluación y revisión: En esta fase se evalúa el videojuego mediante pruebas con usuarios para valorar su jugabilidad, el valor de rejuego y experiencia de juego. Se identifican

los problemas y se llevan a cabo las revisiones necesarias para mejorar la calidad del videojuego.

1.5. Estructura del documento

El presente documento se ha dividido en cinco capítulos: Introducción, Materiales y métodos, Conclusiones y trabajos futuros, Bibliografía y Anexos.

El primero presenta la temática que aborda este trabajo, los objetivos propuestos, la metodología utilizada, las fases del desarrollo y por último la estructura del documento.

En el segundo se describe el cuerpo del trabajo, desde la contextualización teórica hasta el propio desarrollo del videojuego. Concretamente, se trata: los videojuegos referentes, las plataformas de creación de videojuegos, de plataformas multijugador y de sistemas de ranking web, el documento de diseño, la implementación y testeo con los resultados de los mismos.

El tercero relata las conclusiones obtenidas sobre el trabajo realizado, así como las posibles líneas futuras de desarrollo.

En el cuarto se listan las referencias bibliográficas que han sido consultadas para la realización del trabajo.

En el quinto y último es una sección complementaria donde se incluyen materiales adicionales como la relación con los Objetivos de Desarrollo Sostenible de la Agenda 2030, un enlace al ejecutable resultante, la web con las clasificaciones, el código desarrollado, así como un enlace a un video con la ejecución de una partida.

2. Materiales y métodos

2.1. Introducción

En este capítulo se describirán los materiales y métodos utilizados para el desarrollo del proyecto. Se detalla la selección de los videojuegos referentes, el análisis de las diferentes herramientas para la creación de videojuegos, el análisis de las plataformas de desarrollo de videojuegos multijugador, el estudio de las plataformas de ranking web, así como los requerimientos y el proceso de implementación y testeo.

La idea para la creación de este videojuego, proviene de la pasión de quien escribe por los juegos de acción y la emoción de los *roguelites*. Se buscó desarrollar un juego que ofreciera a los jugadores una experiencia desafiante y adictiva, donde cada partida fuera única y llena de sorpresas. Añadiendo funciones y comportamientos que no suelen estar disponibles en este tipo de juegos. La inspiración se encontró en la combinación de mecánicas de combate fluidas, niveles generados de forma procedural y una progresión significativa a lo largo del juego. El objetivo era crear un *roguelite* de acción que mantuviera a los jugadores en constante tensión, enfrentándose a hordas de enemigos y desafiantes jefes finales. Asimismo, se buscó proporcionar una variedad de habilidades y mejoras para que los jugadores pudieran personalizar su estilo de juego y adaptarse a diferentes situaciones.

2.2. Videojuegos referentes

En el mundo de los videojuegos, existen obras que han dejado una huella imborrable en la historia del entretenimiento digital. Estos son los videojuegos referentes, aquellos títulos que han marcado hitos, han establecido nuevos estándares y han dejado una profunda influencia en la industria y en los jugadores. En este capítulo, se exploran algunos de los videojuegos que han servido como referencia e inspiración para la creación del presente trabajo.

1. **Rogue**:

Rogue, lanzado originalmente en 1980, es ampliamente reconocido como uno de los videojuegos más influyentes y pioneros en el género de los *roguelike* (Craddock, 2021). Desarrollado por Michael Toy y Glenn Wichman, Rogue se destacó por su jugabilidad basada en la generación de niveles de forma procedural, usando caracteres ASCII, como se ve en la figura 1, muerte permanente y un alto grado de dificultad (Toy, M. & Wichman, G. ,1980).

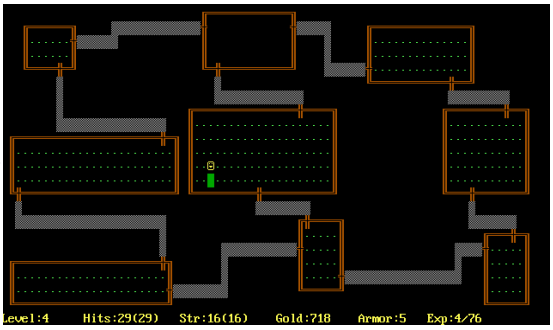


FIGURA 1: CAPTURA DEL VIDEOJUEGO ROGUE, FUENTE: WIKIPEDIA

El juego sentó las bases del género de los *roguelike* y sus elementos característicos, como la generación procedural de niveles, la muerte permanente y la dificultad desafiante, han influido en numerosos videojuegos posteriores (Craddock, 2021). Rogue ha sido ampliamente reconocido como el padre de los videojuegos *roguelike* y ha dejado un legado duradero en la industria de los videojuegos (Zapata, 2005).

2. **Vampire Survivor**

Vampire Survivor es un videojuego de acción y aventura en 2D desarrollado por Doublequote Studio (Galante, 2022). Se lanzó en 2022 y se centra en la lucha contra vampiros en un mundo postapocalíptico.
Los jugadores controlan a un personaje que puede equiparse con armas y habilidades especiales para luchar contra los enemigos. El juego también presenta un sistema de progresión de personajes, lo que permite al jugador mejorar su personaje a medida que avanza en el juego.

FIGURA 2: CAPTURA DE PANTALLA DE VAMPIERE SURVIVOR, FUENTE: STEAM

Este juego se enmarca dentro de la categoría de los denominados “lluvia de balas” (Dormans, J.,2012) donde es el jugador el causante de la misma, como se ve en la figura 2.

Los enemigos aparecen de forma continua alrededor del jugador y el mismo tendrá que ir derrotándolos para poder progresar.

Para PIUM se ha tomado de inspiración ese aspecto de la “lluvia de balas” aunque en este caso, tanto enemigos como jugadores son los causantes de la misma.

3. The Binding of Isaac

The Binding of Isaac es un juego *roguelike* desarrollado por Edmund McMillen y Florian Himsl en 2011 (McMillen & Himsl, 2011). El juego se centra en un niño llamado Isaac que huye a través de su sótano, enfrentándose a enemigos y jefes mientras busca objetos para mejorar sus habilidades, como se muestra en la figura 3. Pertenece al género *roguelite* por el sistema de niveles generados proceduralmente y la muerte permanente.
Una característica interesante del juego son las mejoras o ítems que Isaac puede encontrar durante su aventura. Estas mejoras otorgan habilidades especiales al personaje, como aumento de daño, velocidad o nuevas formas de ataque.

FIGURA 3: CAPTURA DE PANTALLA DE THE BINDING OF ISAAC, FUENTE: STEAM

Cada mejora que obtiene el jugador se refleja visualmente en el personaje de Isaac, lo que le permite apreciar de manera visual cómo estas mejoras afectan su aspecto y habilidades.

El sistema de recompensas de este videojuego sirve de inspiración para el desarrollo de PIUM.

4. Risk of Rain 2

Risk of Rain 2 es un *roguelite* multijugador desarrollado por Hopoo Games y publicado por Gearbox Publishing (Hopoo Games, 2020). En este juego, los jugadores se enfrentan a hordas de enemigos en un entorno 3D generado de forma procedural. La característica principal de Risk of Rain 2 es su acción frenética y su enfoque en la cooperación entre jugadores. Los jugadores deben explorar el mapa, recolectar objetos y mejorar sus habilidades para enfrentarse a enemigos cada vez más desafiantes, se puede apreciar en la figura 4 un fotograma del combate.

FIGURA 4: CAPTURA DE PANTALLA DE RISK OF RAIN 2, FUENTE: STEAM

El juego ofrece una amplia variedad de personajes jugables, cada uno con habilidades únicas y estilos de juego diferentes.

Ese enfoque frenético y de colaboración entre diferentes jugadores también ha servido de inspiración para PIUM.

5. **Hades**:

Hades es un *roguelite* multijugador desarrollado por Supergiant Games. En este juego, los jugadores asumen el papel de Zagreus, el príncipe del Inframundo, en su intento de escapar de su hogar. Hades combina elementos de *roguelite* con una narrativa rica y emocionante. A medida que los jugadores se adentran en las profundidades del Inframundo, descubren nuevos poderes, interactúan con los dioses del Olimpo y desentrañan la historia de Zagreus y su familia.

FIGURA 5: CAPTURA DE PANTALLA DE HADES. FUENTE: STEAM

Además de la jugabilidad desafiante, Hades, destaca por su excelente narrativa, personajes bien desarrollados y un sistema de combate satisfactorio (Supergiant Games, 2020). En la figura 5 se puede apreciar un fotograma del combate.

Es muy interesante el sistema de recompensas de este videojuego, por lo que también ha servido de inspiración para PIUM.

Todos los videojuegos presentados presentan ideas interesantes para PIUM, sin embargo, todos pueden ser mejorados en algún aspecto. PIUM intenta unir en un único videojuego las características más atractivas, y además añade una característica diferenciadora, la clasificación web, incentivando la competitividad e interacción social de los jugadores.

2.3. Análisis plataformas de creación de videojuegos

Para desarrollar un videojuego, existen diferentes plataformas de creación, también conocidas como motores o engines. Un motor de juego es un software que proporciona una serie de herramientas, bibliotecas y funcionalidades predefinidas que facilitan la creación y desarrollo de videojuegos.

Estos motores de juego ofrecen una amplia gama de características que incluyen gráficos en 2D o 3D, físicas, detección de colisiones, sonido, inteligencia artificial, animación, administración de recursos,

gestión de escenas, entre otros. Al utilizar un motor de juego, los desarrolladores se benefician de una base sólida y optimizada que les permite concentrarse en la creación de contenido y la implementación de la lógica del juego, en lugar de tener que construir desde cero cada aspecto técnico.

En el desarrollo de este trabajo, se realizó un análisis exhaustivo de diversas plataformas de creación de videojuegos con el objetivo de seleccionar la más adecuada para el proyecto. A continuación, se presenta un análisis de algunas de las plataformas más populares consideradas:

1. Unity: Una de las plataformas más utilizadas en la industria del desarrollo de videojuegos. Posee un amplio conjunto de herramientas y funcionalidades, incluyendo soporte multiplataforma, capacidad de desarrollo en 2D y 3D, y una comunidad activa de desarrolladores. Unity proporciona una interfaz intuitiva y potentes capacidades de scripting, lo que permite la implementación de mecánicas de juego complejas. Además, cuenta con una amplia gama de plugins y activos disponibles en su Asset Store (plataforma en línea de recursos para Unity), lo que facilita el desarrollo y agiliza el proceso de producción.

Tiene una versión gratuita con funcionalidades básicas y una versión de pago, Unity Pro, que incluye características avanzadas y soporte premium. La versión gratuita es ideal para desarrolladores independientes o equipos pequeños con un presupuesto limitado. Sin embargo, algunas características avanzadas y servicios adicionales, como Unity Cloud Build o el soporte para algunas plataformas específicas, pueden requerir una suscripción o pago adicional (Unity, 2023).

Unity ofrece la posibilidad de publicar un juego de forma gratuita hasta que el juego ingrese una cantidad máxima de dinero, 100 mil dólares, a partir de ese punto, Unity reclama un pago anual por usuario con los planes Unity Plus, si el ingreso es inferior a 200 mil dólares, y Unity pro, sin límites de ingresos. Respectivamente, cuestan 369 $/año y 1877 $/año.

2. Unreal Engine: Otra plataforma líder en el desarrollo de videojuegos, especialmente conocida por su potente motor gráfico y capacidad para crear experiencias visuales impresionantes. Tiene un sistema de desarrollo en tiempo real y proporciona una gran cantidad de herramientas y recursos para el desarrollo de juegos en 2D y 3D. Cuenta con una comunidad activa y una amplia gama de documentación y tutoriales disponibles (Epic Games, 2004).

Posee una versión gratuita, Unreal Engine 5, que permite a los desarrolladores utilizar todas las características del motor sin coste inicial. Sin embargo, Unreal Engine tiene un modelo de negocio basado en regalías, lo que significa que los desarrolladores deben pagar un porcentaje de los ingresos generados por el juego una vez que supere un umbral determinado. En este caso se debe pagar un 5% de los ingresos brutos tras generar un millón de dólares en ingresos brutos (Epic Games, 2023).

Este modelo puede ser beneficioso para los desarrolladores con recursos limitados, ya que no requiere un pago inicial, pero es importante tener en cuenta las implicaciones económicas a largo plazo si el juego es comercialmente exitoso.

3. GameMaker Studio: Plataforma de desarrollo de videojuegos centrada en la creación de juegos en 2D (Overmars, 1999). Conocida por su interfaz amigable y su facilidad de uso. GameMaker Studio proporciona una serie de herramientas y recursos para la creación rápida de juegos, incluyendo un lenguaje de scripting basado en JavaScript, C++ y C#, GameMaker Language (GML).

Ofrece una versión gratuita y varias ediciones de pago con características adicionales. La versión gratuita es adecuada para desarrolladores principiantes o aquellos que deseen probar el software antes de comprometerse con una edición de pago, en esta versión solo se puede exportar para GX Games, la plataforma de videojuegos en el navegador de Opera GX (Opera Norways, 2023). Las ediciones de pago brindan acceso a características avanzadas y permiten a los desarrolladores exportar el juego a múltiples plataformas. Existen tres ediciones con pagos mensuales de 4,99$, 9,99$ y 79,99$ o bien pagos anuales de 49,99$, 99,99$ y 799,99$.

4. Cocos2d: Framework de código abierto ampliamente utilizado para el desarrollo de juegos en 2D. Es conocido por su enfoque en la simplicidad y la eficiencia, permitiendo a los desarrolladores crear juegos rápidamente con un rendimiento óptimo. Actualmente es el motor más utilizado para la creación de los minijuegos más populares en WeChat (Cocos, 2019).

Soporta múltiples plataformas y proporciona un conjunto sólido de herramientas para el desarrollo de juegos 2D. Sin embargo, al ser más específico para juegos en 2D, puede tener limitaciones en términos de funcionalidades 3D y complejidad de desarrollo.

Es una plataforma gratuita, lo que la convierte en una opción económica para desarrolladores independientes o aquellos con un presupuesto limitado. No hay coste de licencia asociado al uso de Cocos2d. Sin embargo, es importante tener en cuenta que puede requerir una mayor curva de aprendizaje en comparación con otras plataformas, y la disponibilidad de recursos y activos puede ser más limitada en comparación con las opciones comerciales.

5. Godot: Motor de juegos de código abierto que ha ganado popularidad en los últimos años (Google Trends, 2023).

Es conocido por su ligereza, versatilidad y facilidad de uso. Godot permite el desarrollo en 2D y 3D, y proporciona un editor intuitivo y un lenguaje de scripting propio, GDScript, similar a C# (Linietsky & Manzur, 2007).

También cuenta con una comunidad activa y una amplia documentación (Linietsky & Manzur, 2014). Godot es completamente gratuito. Esto significa que no hay coste de licencia asociado y los desarrolladores pueden utilizar todas las características del motor sin restricciones.

Además, Godot cuenta con una comunidad activa que comparte recursos, tutoriales y soporte, lo que brinda a los desarrolladores acceso a una amplia gama de recursos sin coste adicional.

En el caso de PIUM, se seleccionó Unity como la plataforma de desarrollo principal. Los motivos para seleccionar Unity frente a las otras opciones son:

Unreal: Éste necesita unos mayores requerimientos de hardware y tiene una comunidad de desarrolladores en constante crecimiento pero que no es comparable a la de Unity. Además, su modelo de negocio está basado en regalías, que, si bien inicialmente no existen, pueden ser un problema ante el posible éxito del producto.

GameMaker Studio: Como ya se ha comentado, está orientado a la creación de juegos 2D, y PIUM debe ser un juego 3D con cierta complejidad.

Cocos 2d: Aunque potencialmente permita la publicación en WeChat, los inconvenientes superan a esa posible ventaja, entre ellos, el más importante, la limitación para crear entornos 3D, al igual que GameMaker Studio.

Godot: Siendo una muy buena opción debido a que es un motor de código abierto completamente gratuito, presenta una gran desventaja, su escasa documentación, sobre todo en comparación a la ofrecida por otros motores, haría que la curva de aprendizaje se dificultase demasiado.

En conclusión, optar por una plataforma diferente a Unity, habría implicado superar una curva de aprendizaje que ya se ha superado con él. Además, la elevadísima cantidad de recursos adicionales que se pueden obtener de forma gratuita, supera por mucho la oferta de cualquiera de las otras plataformas. Y finalmente, si el producto fuera un éxito en ventas, el problema de las regalías se solventa con la adquisición de una licencia muy asequible.

2.3. Análisis plataformas multijugador

FIGURA 6: LOGOS DE LAS PLATAFORMAS MULTIJUGADOR. FUENTE: PROPIA

Para la implementación del modo multijugador, se consideraron diferentes opciones de plataformas y servicios de red disponibles en el mercado, teniendo en cuenta, que tenían que funcionar sobre Unity, ya que este fue el motor de videojuegos seleccionado para el desarrollo. Se analizaron las características y funcionalidades de cada plataforma, así como los requisitos y limitaciones de nuestro proyecto. Se consideraron plataformas como Unity Networking, Mirror, Photon entre otras, en la figura 6 se muestran los logos de las más destacadas.

1. **Unity Networking**

Unity Networking (UNet) es una solución de red clásica integrada en el motor de juego Unity, la cual permitía a los desarrolladores crear juegos multijugador. UNet ofrecía funcionalidades como la sincronización de objetos en red, la creación de *lobbies*, la gestión de conexiones y la comunicación entre clientes y servidores. Sin embargo, a partir de la versión 2019.3 de Unity, UNet se considera obsoleto y ya no se está desarrollando ni se proporciona soporte oficial (House, 2018). Es por ello que se descarta su uso en el desarrollo de PIUM.

2. **Netcode for GameObjects**

Para sustituir a UNet, Unity ha introducido una nueva solución llamada Netcode for GameObjects (Multiplayer Networking). Ésta es una tecnología de red completamente nueva y diseñada para ofrecer una experiencia de juego multijugador más escalable y robusta (Dacuba, 2020). Netcode for GameObjects se centra en mejorar la sincronización y la comunicación entre los objetos del juego en red, lo que resulta en una mayor fluidez y precisión en las interacciones de los jugadores.

La nueva versión de Netcode for GameObjects presenta una arquitectura de red basada en el modelo de autoridad compartida (Shared Authority) (Rémy, P., & Marais, P.,2007), lo que permite una mayor

tolerancia al retraso y a la pérdida de paquetes. Esto significa que los clientes pueden realizar ciertas acciones de forma inmediata y localmente, mientras que otras acciones que requieren autoridad del servidor se sincronizan de manera eficiente.

Además, Netcode for GameObjects ofrece un mayor nivel de control y personalización para los desarrolladores. Proporciona herramientas y API flexibles que permiten ajustar el comportamiento de la red de acuerdo con las necesidades específicas del juego. Esto incluye opciones para la predicción de movimiento, la interpolación y la compensación de latencia.

3. Mirror

Por otra parte, Mirror, es un middleware de red, de código abierto, diseñado específicamente para el motor de juego Unity. Proporciona una solución de red de alto rendimiento y fácil de usar para desarrollar juegos multijugador en Unity (Mischa, 2014).

Mirror se basa en la arquitectura de autoridad compartida (Shared Authority) de forma similar a Netcode for GameObjects de Unity. Esto significa que algunos aspectos de la lógica del juego se ejecutan localmente en los clientes, mientras que otros aspectos que requieren autoridad del servidor se sincronizan de manera eficiente.

Una de las principales ventajas de Mirror es su enfoque en la simplicidad y facilidad de uso. El middleware está diseñado para ser fácil de aprender y utilizar, incluso para desarrolladores con poca experiencia en redes. Proporciona una API limpia y bien documentada, lo que facilita la implementación de la funcionalidad de red en los juegos.

Mirror también ofrece un alto rendimiento gracias a su enfoque en la optimización. Está diseñado para minimizar la latencia y el ancho de banda requeridos, lo que resulta en una experiencia multijugador más fluida y receptiva. Además, admite la compresión de datos y la predicción de movimiento para reducir aún más los problemas de latencia.

Otra característica destacada de Mirror es su comunidad activa y de apoyo. Los creadores de Mirror mantienen una presencia constante en los foros y canales de comunicación, respondiendo preguntas y proporcionando asistencia técnica. Además, la comunidad de Mirror es muy activa y contribuye con extensiones, complementos y mejoras adicionales para el middleware.

4. Photon

Photon es una plataforma de red en tiempo real que ofrece una amplia gama de herramientas y servicios para el desarrollo de videojuegos multijugador. Con Photon, los desarrolladores pueden crear juegos multijugador en línea de alta calidad con una configuración y mantenimiento mínimos. Ofrece una amplia gama de opciones de personalización, desde la creación de partidas y el emparejamiento de jugadores, hasta la gestión de las sesiones y la sincronización de datos en tiempo real. Además, Photon cuenta con una documentación completa y una amplia comunidad de usuarios que pueden proporcionar soporte y ayuda en caso de problemas o dudas durante el desarrollo del proyecto (Photon Team, Sitio Web Oficial, Documentación).

Photon ofrece tres servicios diferentes para juegos multijugador:

- **Photon Cloud** es un servicio en la nube de Photon que proporciona un alojamiento para servidores dedicados y un SDK fácil de usar para desarrolladores. Este servicio ofrece una buena escalabilidad y un alto rendimiento, lo que es ideal para juegos multijugador con una gran cantidad de jugadores.

- **Photon Server**, alternativamente, es una solución de alojamiento de servidores locales que permite una mayor personalización y control sobre el servidor. Esto puede ser beneficioso para juegos multijugador que requieren un mayor control del servidor, pero también requiere una mayor inversión en hardware, alojamiento y mantenimiento.
- **Photon Realtime** es un servicio que ofrece una solución de red en tiempo real, lo que es ideal para juegos de acción y juegos en tiempo real.

5. PUN

PUN (Photon Unity Networking) es una solución de Photon diseñada específicamente para desarrolladores que utilizan Unity (Photon Team, 2023). Combina Photon Cloud y Photon Realtime en una solución fácil de usar para desarrolladores de Unity, lo que permite una rápida implementación de la funcionalidad multijugador en el juego.

En cuanto a los costes se ofrecen diferentes planes los cuales están divididos según los CCU (Usuarios conectados al mismo tiempo). En la versión gratuita se tiene un límite de 20 CCU. PUN permite hasta 50.000 CCU con un precio de $0,29 por CCU.

En resumen, solo tres de las plataformas presentadas son adecuadas para el desarrollo de PIUM, y sus características principales son:

Netcode for GameObjects:

Arquitectura de autoridad compartida: Basado en una arquitectura de autoridad compartida, lo que permite una mayor tolerancia al retraso y a la pérdida de paquetes. Esto proporciona una experiencia de juego más suave y responsiva para los jugadores.

Personalización y control: Ofrece un alto nivel de control y personalización para los desarrolladores. Proporciona herramientas y API flexibles que permiten ajustar el comportamiento de la red según las necesidades específicas del juego.

Mirror:

Arquitectura de autoridad compartida: Permite una mayor tolerancia al retraso y a la pérdida de paquetes.

Simplicidad y facilidad de uso: Se destaca por su enfoque en la simplicidad y la facilidad de uso. Su API bien documentada y su enfoque minimalista hacen que sea más fácil para los desarrolladores implementar la funcionalidad de red en sus juegos.

Enfoque en el rendimiento: Diseñado para ofrecer un alto rendimiento y baja latencia en las comunicaciones de red. Proporciona herramientas y técnicas para optimizar el tráfico de red y garantizar una experiencia de juego fluida.

PUN:

Escalabilidad y confiabilidad: Se basa en una infraestructura de servidores global distribuida, lo que significa que los juegos pueden escalar fácilmente a grandes cantidades de jugadores y regiones geográficas. Tiene una buena reputación en términos de confiabilidad y estabilidad de su red.

Fácil de usar: API intuitiva y bien documentada que facilita la implementación de la funcionalidad de red en los juegos de Unity. También proporciona un conjunto de herramientas y servicios en la nube que facilitan la gestión y configuración de la red.

Amplia compatibilidad: Compatible con múltiples plataformas, lo que permite a los desarrolladores crear juegos multijugador que se pueden jugar en diferentes dispositivos, como PC, consolas y dispositivos móviles.

Finalmente, se decidió utilizar PUN como la plataforma del modo multijugador. Los motivos son:

Netcode for GameObjects: Actualmente es una opción poco utilizada por su reciente creación. Además, tampoco posee servidores propios.

Mirror: Opción menos conocida y utilizada, además tiene una documentación limitada y ausencia de servidores dedicados. Además, puede que tenga una falta de soporte a largo plazo.

PUN: El único con un servidor remoto que aloja la partida en lugar de ser los jugadores quienes lo hagan, como sucede con Mirror y NetCode, hace que sea mejor en aspectos de seguridad, así como, estar a prueba de trampas por parte de los jugadores. Al ser un juego centrado en el aspecto competitivo se quiere que los jugadores tengan las mismas posibilidades. En cuanto a costes, la versión gratuita es suficiente para abarcar los requisitos de este TFG.

2.4. Análisis sistemas ranking web

FIGURA 7: LOGOS DE LAS SISTEMAS DE RANKING WEB. FUENTE: PROPIA

Con el fin de implementar un sistema de ranking web para el videojuego PIUM, se han analizado diferentes sistemas de gestión de puntuaciones en línea, tales como Unisave, LootLocker y Google Play Games Services. En la figura 7 se muestran los logos de los mismos.

1. **Unisave**

Servicio en la nube desarrollado para Unity que proporciona diversas funcionalidades, incluido un sistema de ranking integrado (Mayer, 2020). El sistema de ranking de Unisave permite a los desarrolladores de juegos implementar un sistema de puntuaciones y clasificaciones en línea para sus juegos.

Asimismo, Unisave se basa en una arquitectura cliente-servidor, donde los datos de puntuación de los jugadores se almacenan y administran en servidores remotos. Esto permite que los jugadores envíen

sus puntuaciones al servidor y las comparen con otros jugadores para establecer clasificaciones y rankings.

Una de las principales ventajas del sistema de ranking de Unisave es su facilidad de uso e integración con juegos de Unity. Proporciona una API sencilla y bien documentada que los desarrolladores pueden utilizar para enviar y recibir datos de puntuación desde sus juegos. Igualmente, Unisave se encarga de toda la infraestructura subyacente, como el almacenamiento de datos y la gestión de la base de datos, lo que facilita la implementación y el mantenimiento del sistema de ranking.

Además de la funcionalidad básica de clasificación y puntuación, Unisave también ofrece características adicionales para enriquecer el sistema de ranking. Esto incluye la posibilidad de establecer rangos o divisiones en las clasificaciones, sistemas de temporadas para restablecer las puntuaciones periódicamente, tablas de clasificación filtradas por criterios específicos y la opción de mostrar datos estadísticos adicionales sobre las puntuaciones de los jugadores.

Unisave cuenta con varios planes de pago y una versión gratuita. La versión gratuita permite hacer 20.000 peticiones diarias a la base de datos. En las versiones de pago de $8, $25 y $35 se tienen respectivamente 30.000, 100.000 y 200.000 peticiones.

2. Google Play Games Services

Google Play Games Services es una plataforma de servicios proporcionada por Google que permite a los desarrolladores integrar funcionalidades multijugador, logros, marcadores y otros servicios relacionados en juegos desarrollados para dispositivos Android (Google, 2023).

Dentro de Google Play Games Services, uno de los servicios destacados es el sistema de ranking, también conocido como Leaderboards. Este servicio permite a los desarrolladores implementar tablas de clasificación en sus juegos, donde los jugadores pueden comparar sus puntuaciones y logros con otros jugadores.

La mayor desventaja es que es un sistema propio de Android y no funciona en Windows.

Su uso no implica más que el coste de una fianza de 25$ para crear una cuenta de desarrollador de Google y verificarla.

3. LootLocker

LootLocker ofrece un sistema de ranking que permite a los desarrolladores implementar y gestionar tablas de clasificación en sus juegos (Berg, 2021). Estas tablas de clasificación permiten a los jugadores comparar sus puntuaciones y logros con otros jugadores, fomentando la competencia y el espíritu competitivo.

Una de las ventajas del sistema de ranking de LootLocker es su integración sin problemas con otros aspectos del juego, como los logros y la progresión del jugador. Los desarrolladores pueden vincular los logros desbloqueados por los jugadores con las tablas de clasificación, lo que brinda un enfoque integral para la competencia y el reconocimiento de los jugadores más destacados.

Otra ventaja del sistema de ranking de LootLocker es su escalabilidad y rendimiento. La plataforma está diseñada para manejar grandes volúmenes de datos de clasificación y mantener un rendimiento óptimo, incluso con una gran cantidad de jugadores y puntuaciones. Esto asegura una experiencia fluida y sin interrupciones para los jugadores, sin importar cuántos participantes haya en las tablas de clasificación.

Además de las características básicas de clasificación, LootLocker también ofrece herramientas de análisis y seguimiento de datos. Los desarrolladores pueden acceder a métricas y estadísticas detalladas sobre el rendimiento de las tablas de clasificación, como el número de jugadores activos, la frecuencia de actualización de las puntuaciones y la participación de los jugadores. Esto proporciona información valiosa para ajustar y optimizar las estrategias de clasificación y mejorar la participación de los jugadores (LootLocker, 2020).

En cuanto al precio, depende de la cantidad de usuarios mensuales que se tengan. La opción gratuita permite 10.000 usuarios mensuales y los precios varían según el rango de jugadores que se tengan.

A partir de los 10.000 serían $0 más $7.25 por cada 1.000 de más. A partir de los 50,000 $250 más $5 por cada 1.000 de más (LootLocker, 2020). Existen más planes, pero los más relevantes serían esos.

Para el desarrollo de PIUM se ha elegido LootLocker. Los motivos de porque se ha elegido esta frente a las otras opciones son:

> **Unisave**: Aunque es una opción viable tanto por coste como por su naturaleza multiplataforma, está limitado en términos de comunidad de desarrolladores y recursos en línea.
>
> **Google Play Games Services**: Su principal desventaja es estar limitado al ecosistema de Google Play y dispositivos Android. Sin perder de vista los posibles cambios en las políticas de Google, lo cual se escapa del alcance de este TFG.
>
> **LootLocker**: destaca por la amplia compatibilidad con diferentes motores de juegos y plataformas. También la amplia documentación asociada y la posibilidad de personalización y adaptación a las necesidades específicas del juego. Incluso, permite la obtención de estadísticas y explotación de los datos vía el panel del desarrollador o su api, que es un aspecto interesante pero que se escapa del ámbito de este trabajo. Además, en su versión gratuita cubre las necesidades de este TFG. Por último, es el único compatible con una API en JavaScript para poder mostrar los valores en una página web propia, que era una característica que se quería implementar en este trabajo.

2.5. Documento de diseño

El documento de diseño de videojuego (GDD por sus siglas en inglés) se utiliza como una valiosa herramienta de trabajo durante el desarrollo del proyecto. Este documento es una guía detallada que describe y especifica todos los aspectos importantes del videojuego, desde la mecánica de juego y la narrativa hasta los personajes, el arte y el sonido. Además, a lo largo del proceso de desarrollo, el documento de diseño de videojuego se actualiza y se ajusta según sea necesario. A medida que se toman decisiones y se realizan cambios en el diseño, se reflejan en el documento para mantenerlo actualizado y relevante.

2.5.1. Objetivos del juego

***Roguelite* de acción en perspectiva isométrica**: Combina la perspectiva isométrica con la mecánica de un juego *roguelite*, ofreciendo una experiencia de juego diferente. Los jugadores controlan una nave espacial en intensas batallas llenas de acción contra varios enemigos.
Multijugador cooperativo: Presenta la funcionalidad multijugador cooperativo, lo que permite a los jugadores formar equipos con sus amigos y enfrentar desafíos juntos. La interacción en tiempo real y la coordinación con los aliados crean una experiencia multijugador dinámica y atractiva.

Gestión de puntuación en línea: Incorpora un sistema de gestión de puntuación en línea, donde las puntuaciones de los jugadores se almacenan en una base de datos remota. Esto permite a los jugadores competir globalmente y esforzarse por alcanzar altos rankings, agregando un aspecto competitivo al juego.
Compatibilidad multiplataforma: Está diseñado para ser compatible con múltiples plataformas, lo que permite a los jugadores disfrutar del juego en varios dispositivos, incluidos PC, consolas y dispositivos móviles. Esta accesibilidad amplía la base de jugadores potenciales y aumenta el alcance del juego.

Género

Roguelite de acción.

La gran idea

Eliminar a todos los enemigos consiguiendo la mejor puntuación posible con ayuda de tus amigos.

Público

Es un videojuego para todos los públicos. El rating en PEGI sería de 7.

Plataforma

Windows y Android.

Requisitos técnicos

Cooperativo en línea, sistema de ranking para puntuaciones online.

Tipo de jugabilidad

Lluvia de balas.

Narrativa

Eres una nave que mientras explora se encuentra enemigos a los cuales debes matar para poder seguir avanzando. El juego empieza nada más encender la partida, no hay un tutorial.

Controles

A continuación se muestra la Tabla 1 en la que se pueden ver las acciones que se pueden realizar en el juego junto al control que la provoca.

Acción	Control
Mover la nave	Botón principal del mouse o Joystick virtual para Android
Pausar	Tecla “ESC” o botón virtual en pantalla
Alternar la cámara libre/ cámara fija	Tecla “Y”
Centrar la cámara en el jugador	Tecla “Espacio”
Mover la cámara en modo cámara libre	Poner el cursor en los bordes o botón central del mouse

TABLA 1: RESUMEN CONTROLES

Requerimientos tecnológicos

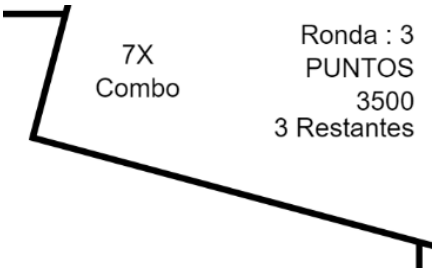
Se usará Unity 2021.3.11f1 para el desarrollo general. El multijugador se implementa con Photon Unity Networking y el sistema de clasificación con LootLocker.
Toda la funcionalidad se crea desde cero, excepto la relacionada con las utilidades locales que incorpora el propio Unity, como son: el sistema de navegación, las físicas, las luces, el renderizado y las colisiones
Los requerimientos del PC para ejecutar el juego son: Intel i3 o equivalente, con 4 GB de RAM y una tarjeta gráfica Intel iRIS Xe o equivalente. Asimismo, se requiere conexión a internet de al menos 500Kbps. En cuanto a móvil se requiere una versión de Android mínima de 5.1, con 2 GB de RAM y un procesador Snapdragon 660 o similar.

Cámara

Durante todo el juego se emplea una cámara en perspectiva isométrica.

Interfaz de usuario (HUDs etc.)

A continuación se muestra la Tabla 2 en la que se muestran las partes de la interfaz de usuario.

Descripción	Imagen
Interfaz de usuario durante la partida, mostrado en la figura 8. Presenta cuatro zonas activas, situadas en las esquinas de la pantalla, las cuales se describen a continuación.	 FIGURA 8: DIAGRAMA DE LA INTERFAZ DE USUARIO EN PARTIDA. FUENTE: PROPIA
En la figura 9 se muestra el marcador de puntos, cuyas partes nombradas de izquierda a derecha serian: Combo actual, ronda actual, puntos de la partida y los enemigos restantes.	 FIGURA 9: DIAGRAMA DEL BANNER DE PUNTOS. FUENTE: PROPIA

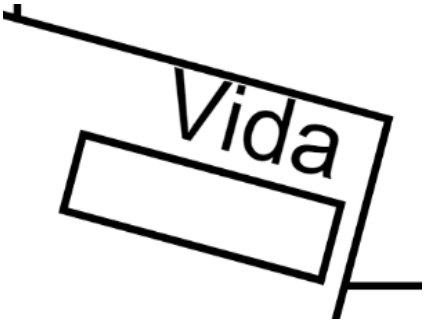
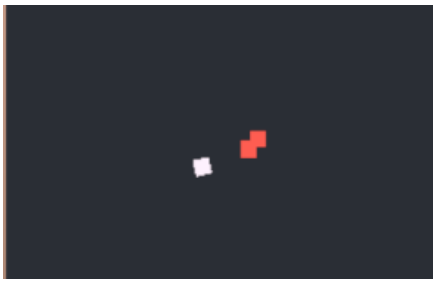
El indicador de vida, mostrado en la figura 10, se irá llenando o vaciando de forma dinámica según la cantidad de vida del jugador.	 FIGURA 10: DIAGRAMA DE LA REPRESENTACIÓN DE LA VIDA. FUENTE: PROPIA
El minimapa, como se ve en la figura 11 muestra: La posición relativa de jugadores y enemigos durante la partida. Un icono en blanco que representa al jugador e iconos en rojo a los enemigos	 FIGURA 11: DIAGRAMA DEL MINIMAPA. FUENTE: PROPIA
El botón de pausa se muestra en la figura 12 y como su propio nombre indica, permite detener la ejecución de la partida,.	 FIGURA 12: DIAGRAMA DEL BOTÓN DE PAUSA. FUENTE: PROPIA

TABLA 2: DIAGRAMAS DE LA INTERFAZ DE USUARIO

Wireframe Pantallas jugador

A continuación, se presenta en la figura 13 el diagrama de la interfaz de usuario del videojuego.

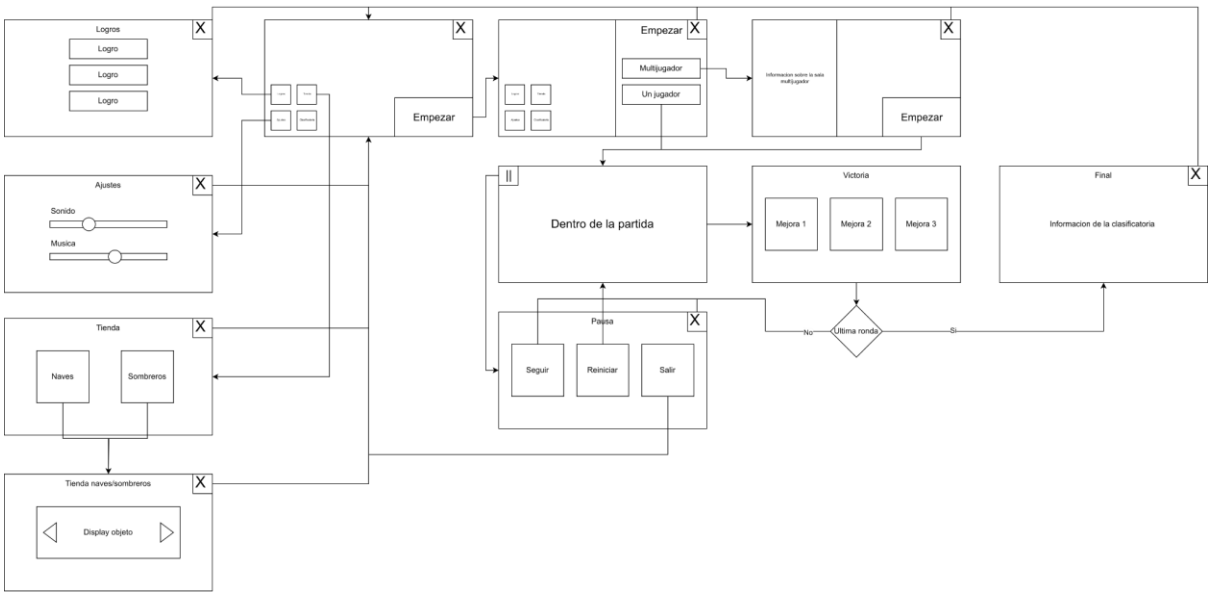


FIGURA 13: DIAGRAMA DE LA INTERFAZ DE USUARIO EN PARTIDA. FUENTE: PROPIA

2.5.2. High Level Player Interface

Pantalla de título, Pantalla de inicio e Idle

La idea es que el jugador pueda comprender de un vistazo que PIUM es un juego desenfadado, un juego que no busca contar una gran historia, sino que busca el simple entretenimiento del jugador. Algo que representa PIUM es el nombre, que no es solo un acrónimo sino también una onomatopeya del sonido al disparar un arma. En la figura 14 se muestra como cargará la escena para el jugador dando así un poco de dinamismo a las escenas estáticas.

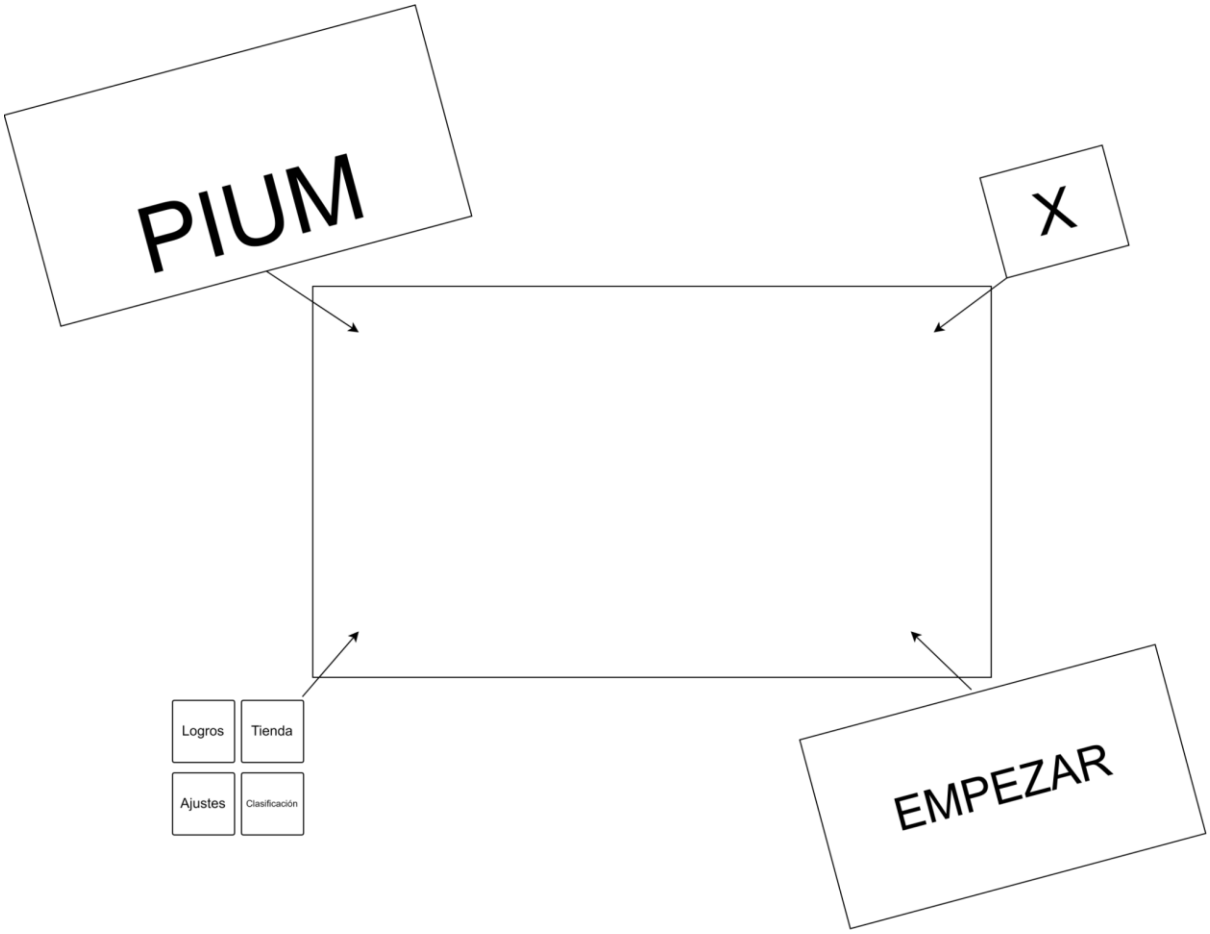


FIGURA 14: DIAGRAMA DEL MOVIMIENTO REALIZADO POR LA INTERFAZ AL CARGAR. FUENTE: PROPIA

Como se muestra en la figura 14, las flechas indican el desplazamiento que harán los elementos de la interfaz, siendo el rectángulo la pantalla del usuario. En la figura 15 podemos ver como quedaría la misma al finalizar las animaciones.

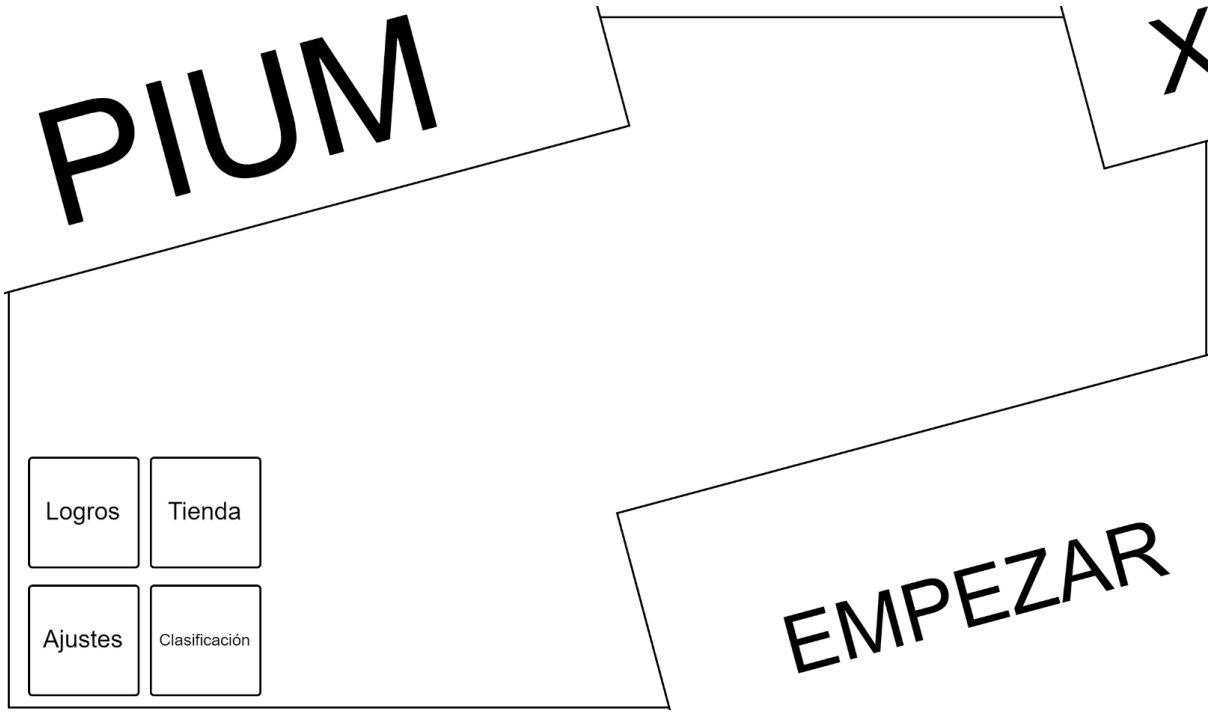


FIGURA 15: DIAGRAMA DEL MENÚ PRINCIPAL. FUENTE: PROPIA

En la Tabla 3 se indica la función de cada menú.

Nombre	Descripción
Logros	Abre la pantalla de logros
Tienda	Abre la pantalla de la tienda
Clasificación	Abre la página web con la clasificación
Ajustes	Abre la pantalla de ajustes
(X)	Cierra la aplicación
Empezar	Abre el menú de selección del modo multijugador o un jugador

TABLA 3: RESUMEN DE LAS FUNCIONES DEL MENÚ

Credenciales

Las credenciales necesarias son el logo de Unity puesto que se utiliza la licencia tipo Personal de Unity.

Pantalla de opciones

Poseerá una pantalla de ajustes para los sonidos y la música con un slider para poder ajustarlos.

Pantalla de carga

Habrá una pantalla de carga al inicio de las rondas para ocultar la generación del mapa. En la pantalla de carga se mostrará un texto “cargando” con un fondo plano.

Guardado de puntuación

La puntuación obtenida en las partidas se acumula, así, después de haber conseguido cierta cantidad, se puede canjear en la tienda por naves o cosméticos.

No se almacena la máxima puntuación del jugador en local, solo si él mismo desea publicarlo en la clasificatoria donde será almacenada.

En caso de que la subida fallara o el jugador no tuviera conexión, sí se almacenarían las puntuaciones para que cuando recupere la conexión pueda reintentar la subida.

Mejor puntuación/Clasificatoria

El juego cuenta con un sistema de ranking que está disponible en ambos modos de juego: el modo multijugador y el modo para un solo jugador.

El sistema de ranking de PIUM se centra en rastrear y mostrar las puntuaciones finales de los jugadores al completar niveles. Al finalizar o morir en una partida, ya sea en el modo multijugador o en el modo para un solo jugador, se da la oportunidad de registrar la puntuación alcanzada por cada jugador.

Estas puntuaciones finales se utilizan para clasificar a los jugadores en el sistema de ranking, lo que les permite comparar su rendimiento con el de otros jugadores.

Es importante destacar que PIUM gestiona las puntuaciones finales como una forma de fomentar la competencia y la superación personal. No se rastrean ni muestran estadísticas detalladas de los jugadores, sino que el sistema de ranking se centra en las puntuaciones obtenidas al finalizar los niveles.

2.5.3. Información general del juego

Core Game Mechanics

En PIUM, se utiliza una mecánica principal que forma parte de la jugabilidad central del juego, el movimiento. A raíz del mismo, el jugador disparará automáticamente a los enemigos cercanos, centrándose así el jugador solo en esquivar.
También, a medida que los enemigos vayan siendo derrotados, podrán soltar dos posibles consumibles: una pequeña regeneración de vida o una mejora para la nave.

Mundo

El mundo es generado en cada ronda con una base en forma de rombo, en cuyos bordes hay barreras para evitar que el jugador salga del nivel.
El mapa base es un cuadrado rotado 45 grados para lograr el efecto isométrico.
Se va agrandando según se completan rondas para poder almacenar más enemigos.
Estará compuesto de una cuadrícula con cuadrados de 10 x 10 donde dentro habrá diferentes escenarios para así, en la generación procedural, poderlos poner dentro.

Los niveles

Al tratarse de un juego procedural los niveles se generan automáticamente. Durante el transcurso de la partida el mapa se irá generando según la ronda actual, a mayor ronda mayor cantidad de enemigos.

Diagrama de progreso del jugador

El jugador irá completando rondas de forma continuada hasta alcanzar las 5 en su primera partida y las 10 en el resto de partidas.
Cada nivel será generado en el momento. Con cada ronda la cantidad de enemigos irá aumentando.

2.5.4. El jugador

Personaje del jugador

FIGURA 16: NAVE QUE CONTROLA EL JUGADOR.
FUENTE: PROPIA

El personaje está basado en la nave del "space kit" de Kenney (Kenney, 2020). Se ha usado el color rosa brillante para mostrar que esa pieza es la del usuario, mostrado en la figura 16. Se usa un negro para dar un contraste con el fondo y el rojo representa el chasis actual cambiando según el que se seleccione.

Matriz del jugador

En la tabla 4 se muestra un resumen de las estadísticas que tendrá el jugador.

Nombre	Valor
Velocidad del jugador	**7**
Aceleración del jugador	**25**
Daño de las balas	**10**
Tiempo entre cada disparo	**0.4 s**
Dispersión de las balas respecto al punto apuntado	**5 º**
Rango de detección de enemigos	**25 m**
Tiempo de vida de las balas (Alcance)	**5 s**
Velocidad de la bala	**12 m/s**
Balas disparadas simultáneamente	**1**

TABLA 4: RESUMEN DE LAS ESTADÍSTICAS DEL JUGADOR

Habilidades del jugador

Las mejoras están bloqueadas detrás de logros para así incentivar a jugar varias partidas. De este modo, se encuentran divididas en distintos niveles, los cuales se irán incrementando una vez se obtenga una recompensa. Existe la posibilidad de que el siguiente nivel aparezca en la próxima selección de mejoras. En el juego existen diferentes tipos de mejoras:

Balas: Las balas son mejoras que suelen perfeccionar las características de disparo tales como daño, tasa de disparo, dispersión, rango, tiempo de vida de la bala, velocidad de la bala, tamaño de la bala y las balas adicionales a disparar.

- Balas de metales (acero, cobre, plomo): Son balas que su principio se fundamenta en balas pesadas. Esto implica un mayor daño y una menor velocidad de la bala.
- Balas de energía (láser, plasma, iones): Son balas más rápidas y dañinas.
- Balas del cielo (majestuosa, heroica, legendaria, sagrada, celestial, divina): Son balas que mejoran varias estadísticas como: Daño, velocidad de ataque, velocidad de la bala, alcance y rango.
- Balas del infierno (mala, apocalíptica, infernal, demoníaca, belcebala): Son balas más pensadas en puro daño, son las que más lo aumentan.
- Aumento de la cantidad de balas (doble, triple, cuádruple): Como su nombre indica son mejoras que aumentan las balas a disparar.

Cañón: Estas mejoras se enfocan en daño, tasa de disparo, dispersión, rango, tiempo de vida de la bala, velocidad de la bala y el tamaño de la bala.

- Cañón preciso (preciso, francotirador, antitanque): Son mejoras enfocadas en reducir la dispersión, aumentando el alcance, rango y la velocidad de las balas. En los niveles más altos disminuye la velocidad de ataque, pero aumenta el daño.
- Cañón rápido (limpio, lubricado, turbo): Se basan en aumentar la velocidad de ataque a costa de reducir la precisión. También pueden aumentar la velocidad de las balas.
- Cañón grande (de boca ancha, ensanchado, grueso): Dispara balas más

grandes que hacen más daño.

Chasis: Éstas se enfocan en la resistencia y velocidad del jugador (vida, velocidad, aceleración)

- Chasis pesados (reforzado, blindado, titanio): Chasis que hacen la nave más resistente, pero reducen la velocidad.
- Chasis ligeros (ligero, carbono, kevlar): Reducen la resistencia a costa de dar velocidad, aunque a niveles altos pueden llegar a dar más resistencia.

Propulsor: Éstas afectan al desplazamiento del jugador (velocidad, aceleración)

- Propulsor mejorado (mejorado, sulfuro, nitro): Otorga una mayor velocidad y aceleración.
- Propulsor pseudo médico (homeopático, biomagnético, 433Hz): Son una serie de mejoras que no otorgan ninguna bonificación.

Áreas: Éstas se basan en crear efectos alrededor del jugador tales como balas o escudos.

- Áreas de balas (balas, satélites): Hace aparecer balas alrededor del jugador que lo orbitan. Estas balas poseerán un alto daño para fomentar el arriesgarse a acercarse al enemigo por su alto daño. Conforme mejore, habrá más balas simultáneamente.
- Áreas de escudos (escudo, escudos): Otorga un escudo que gira alrededor del jugador y parará todo proyectil que se le acerque en esa dirección. Conforme mejore dará más zonas de protección.

Utilidades: Estas son mejoras centradas en cambiar el funcionamiento de la partida dando ayudas fuera del aumento de estadísticas también.

- Radar: Otorgará un mayor rango de detección de los enemigos.
- Balas guiadas: Hará que las balas que en condiciones normales no impactarían en el enemigo cambien de dirección e impacten.

Cada mejora tendrá un reflejo visual en el jugador.

Combate

En la figura 17 se muestra un storyboard de cómo funcionará el combate.

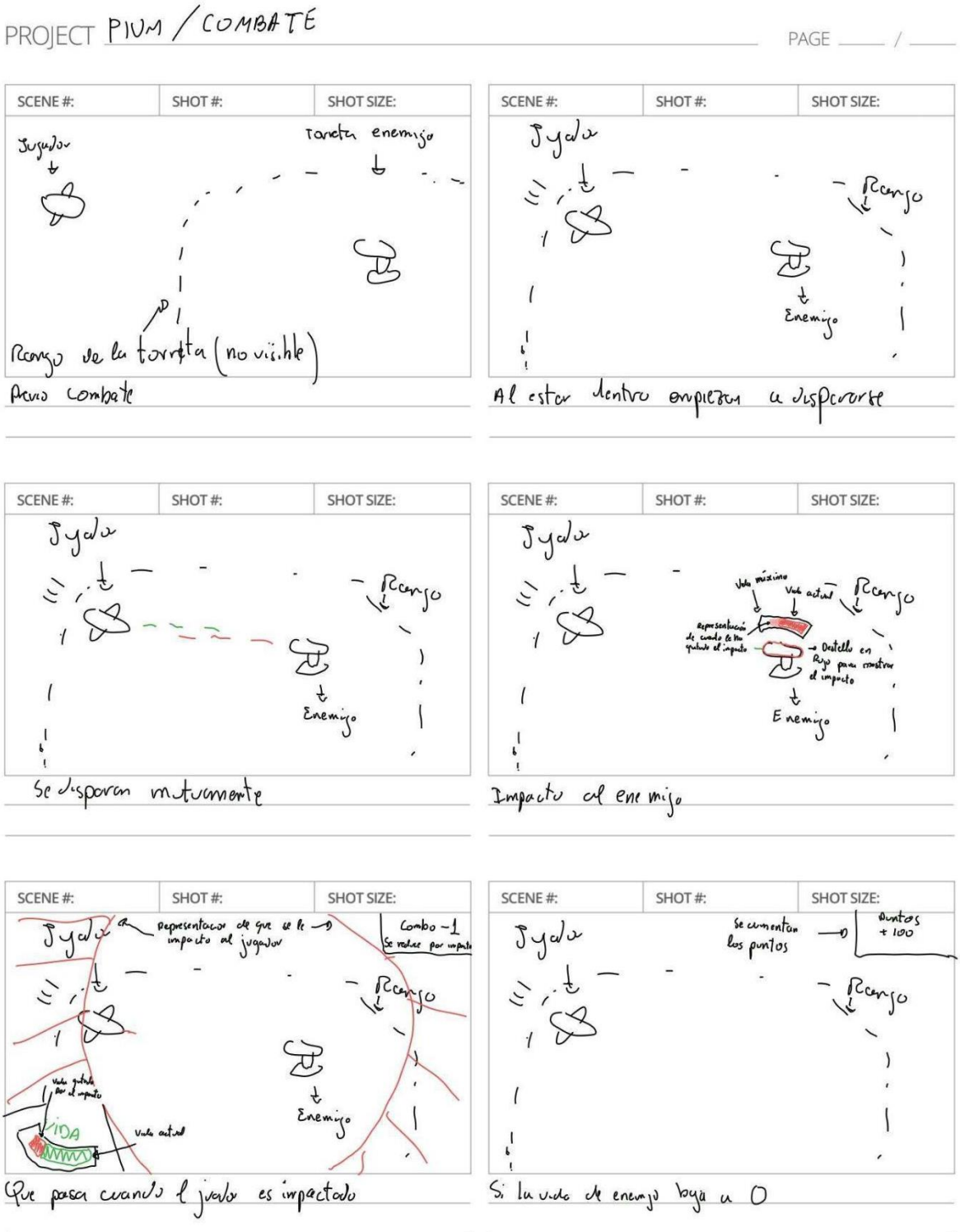


FIGURA 17: STORYBOARD DEL COMBATE. FUENTE: PROPIA

Muerte del jugador/enemigo

Cuando el jugador muere es redirigido a la pantalla para publicar la puntuación, mostrando un aviso de su situación.

Al morir, los puntos del jugador se le acumulan para su uso en la tienda.

En cuanto a la muerte de los enemigos, se representa con ellos desapareciendo, otorgando puntos y una posibilidad de soltar un consumible, el cual puede ser recogido por el jugador.

2.5.5. Pantallas dentro del juego

Guardar/Checkpoint

El juego guarda el progreso de forma automática cuando un valor es modificado.
Se guardan tanto los logros obtenidos, las naves y sombreros comprados, así como los activos actualmente y la puntuación total.

Logros

A continuación, en la tabla 5, se muestran los logros disponibles. Cada uno de ellos desbloquea una serie de mejoras basadas en la temática del logro.

Nombre	Requisito
Ruta base plus	Consigue 10.000 puntos en una partida
Ruta cencia	Consigue un combo de 50 al acabar la partida
Ruta cencia plus	Consigue un combo de 100 al acabar la partida
Ruta heroica	Completa la última ronda sin que te golpeen
Ruta divina	Completa una partida sin que te golpeen
Perseverancia	Consigue tu primer millón de puntos
Lo importante es terminar	Termina una partida con un combo inferior a 5

TABLA 5: LISTA DE LOS LOGROS DISPONIBLES

Economía/moneda

La economía se basa en los puntos que se generen durante las partidas. Estos puntos pueden ser utilizados para comprar naves o cosméticos.
Las naves disponibles son: bulky, chunky, plana, speed y última esperanza.
Cada nave poseerá diferentes atributos.
Sombreros disponibles: alien, radar y huesos.

La tienda de naves se mostrará como en la figura 18, teniendo una representación visual de la pieza, así como un botón para adquirirla, con el precio. En el caso de haberla adquirido previamente, el botón mostrará la palabra "equipar" si no está equipada. Además, se verán las estadísticas de la nave en la parte derecha en forma de barras de progreso.

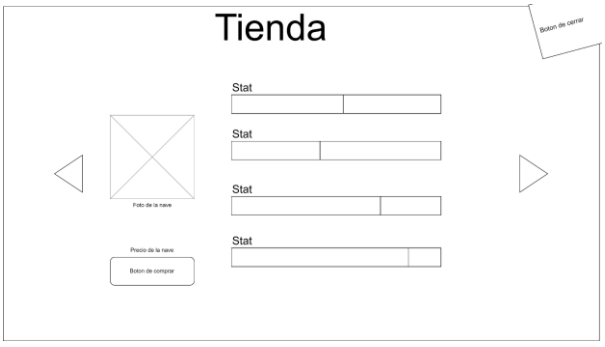


FIGURA 18: DIAGRAMA DE LA TIENDA DE NAVES. FUENTE: PROPIA

Por otra parte, la tienda de cosméticos, mostrada en la figura 19, mostrará una representación visual de la pieza, así como un botón para adquirirla, con el precio mostrado encima del botón o en caso de poseerla, pero no tenerla equipada, equipar.

Estos cosméticos otorgan unos bonos que el jugador no puede ver, con el fin de crear un poco de misticismo alrededor.

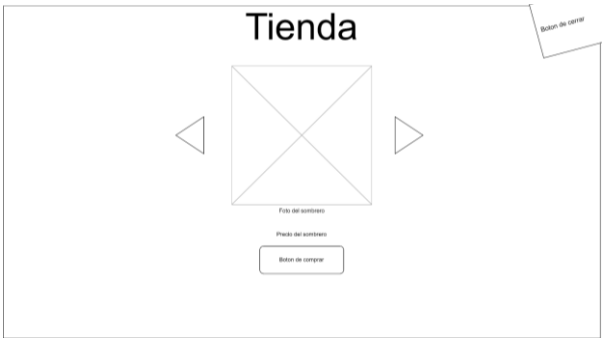


FIGURA 19: DIAGRAMA DE LA TIENDA DE COSMÉTICOS. FUENTE: PROPIA

Una vez adquirida cualquier tipo de compra, no se podrá devolver ni cambiar.

2.5.6. NPCs (non-player characters), Enemigos, IA, Transporte

En el diseño de este videojuego se han incluido 2 tipos de enemigos diferentes, las naves y las torretas.

Las torretas actúan de forma que cuando el jugador se acerque a su rango de disparo, éstas empezarán a disparar.

Para el caso de las naves, tienen dos rangos diferentes, un rango de detección, de forma que al detectar a un jugador irán a buscarlo, y un rango de disparo de igual funcionamiento que el de las torretas.

Los enemigos aparecen durante la generación procedural en las casillas donde se determine que deberán aparecer enemigos. Se generan en función del peso del mismo, por ejemplo, si en una casilla se destinan 4 puntos para enemigos, se podrán generar 4 de peso 1, 1 de peso 2 y 2 de peso 1, 2 de peso 2, 1 de peso 3 y 1 de peso 1 o 1 de peso 4.

Al ser derrotados tienen la posibilidad de soltar consumibles.

Devuelven una cantidad de puntos dependiendo de su “poder”, este poder es calculado según su vida y su daño por segundo. El daño por segundo depende del daño, la velocidad de ataque y cuantas balas dispara.

La fórmula usada seria:

$$puntos = \frac{daño * balas\ disparadas}{ms\ entre\ disparos} + vida$$

Para mostrar que el jugador ha golpeado a un enemigo, éste se tornará rojo para mostrar que ha sido “herido” y también verá su vida mermar.

2.6. Implementación

Se ha llevado a cabo la implementación de todas las funcionalidades previstas en el documento de diseño, incluyendo la generación procedural de niveles, el sistema de combate, la implementación de PUN para el modo multijugador y la integración de LootLocker para el sistema de ranking web. Se ha utilizado el lenguaje de programación C# y el motor de videojuegos Unity en su versión 2021.3.11f1.

2.6.1 Generación aleatoria de los niveles

En cuanto a la generación procedural, se ha realizado mediante la llamada “Wave Function Collapse”.

El método de "Wave Function Collapse" (Colapso de la Función de Onda, en español) es un algoritmo utilizado en la generación de contenido procedural, particularmente en el campo de la generación de imágenes y niveles de juegos (Gumin, 2016). Fue desarrollado por el programador Maxim Gumin y se basa en principios de la mecánica cuántica.

El objetivo del método de “Wave Function Collapse” es generar una representación visual coherente y variada a partir de una imagen o patrón de entrada más pequeño. La idea principal es tomar un fragmento inicial de la imagen y utilizarlo como una "superposición" de posibles configuraciones, conocidas como "superposiciones cuánticas". Luego, estas superposiciones se colapsan en una sola configuración en función de las restricciones definidas.

El algoritmo funciona de la siguiente manera:

1. Se toma una muestra de entrada, que son *prefabs* con partes de escenario dentro, éstos se llamarán casillas.
2. El mapa se divide en una cuadrícula, en cada casilla existen diferentes valores que puede tomar, basados en los vecinos cercanos y las restricciones locales. Estos posibles valores se conocen como "superposiciones".
3. Se crea una matriz de superposición que representa todas las posibles combinaciones de valores para cada casilla. Esta matriz se actualiza a medida que se aplican las restricciones.
4. A continuación, se elige una casilla para colapsar en una configuración determinada. Se selecciona una de las casillas con menos superposiciones posibles.
5. Cuando se colapsa una casilla, se propagan las restricciones a las casillas vecinas, lo que reduce las superposiciones posibles en esas casillas. Esto asegura que las restricciones se mantengan en todo momento y se preserve la coherencia de la imagen generada.
6. Se repiten los pasos 4 y 5 hasta que todas las casillas están colapsadas en configuraciones únicas.

Es importante destacar que el método de “Wave Function Collapse” es computacionalmente intensivo y puede requerir mucho tiempo y recursos para generar resultados de alta calidad, especialmente para cuadrículas grandes o complejas. Esta circunstancia no es un problema en este trabajo, ya que dicho tiempo queda camuflado en la carga de los niveles, ofreciendo, una forma única de generar contenido proceduralmente.

2.6.2 Juego colaborativo en red

Al implementar un juego colaborativo con PUN (Photon Unity Networking), se deben seguir ciertos pasos iniciales para configurar adecuadamente el entorno de desarrollo.

El primer paso consiste en descargar el paquete de PUN desde el sitio web de Photon. Este paquete contiene los archivos necesarios para la integración de la red en el juego. Una vez descargado, se debe importar el paquete en el proyecto de Unity. Para hacer esto, se puede ir a "Assets" en la barra de

menú de Unity, seleccionar "Import Package" y luego "Custom Package". A continuación, se busca el archivo descargado y se sigue el asistente de importación para completar el proceso.

Después de haber importado el paquete de PUN, se requiere crear una cuenta en Photon Cloud. Photon Cloud es un servicio en la nube proporcionado por Photon que facilita la comunicación y sincronización de datos entre los jugadores. Al crear una cuenta en Photon Cloud, se obtiene acceso a un panel de control donde se pueden administrar los juegos y obtener los AppID necesarios.

Una vez que se ha creado una cuenta en Photon Cloud y se ha iniciado sesión en el panel de control, es necesario crear un nuevo juego. Esto se puede hacer proporcionando un nombre para el juego y seleccionando las opciones de configuración adecuadas, como la región del servidor y el modelo de precios. Después de crear el juego, se generará un AppID único asociado a ese juego. Este AppID es esencial para establecer la conexión entre el juego y Photon Cloud.

Con el AppID obtenido, se debe agregar este código al proyecto de Unity. Esto se puede hacer a través del Inspector de Unity seleccionando el objeto PhotonServerSettings en la jerarquía. En el Inspector, se encuentran los campos correspondientes al AppID. Aquí se debe insertar el AppID generado en el panel de control de Photon Cloud.

Una vez que se ha completado la configuración inicial descargando e importando el paquete de PUN, creando una cuenta en Photon Cloud y obteniendo los AppID necesarios, se está listo para comenzar a utilizar PUN para desarrollar el juego colaborativo. A partir de este punto, se pueden utilizar los Photon Views para sincronizar los objetos del juego entre los diferentes jugadores.

Un Photon View es un componente que se agrega a un objeto en Unity y que permite la sincronización de sus propiedades a través de la red. Para utilizar los Photon Views, primero se debe agregar un Photon View al objeto que se desea sincronizar. Esto se puede hacer seleccionando el objeto en la jerarquía de Unity y haciendo clic derecho, luego seleccionando "Add Component" y buscando "Photon View" en la lista.

Una vez agregado el Photon View al objeto, se pueden configurar sus propiedades en el Inspector. Uno de los campos más importantes es el “Observed Components” (componentes observados), que es una lista que define las propiedades del objeto deben estar sincronizadas entre las diferentes instancias del juego.

Además, se debe establecer el "Synchronization" (sincronización) del Photon View. Hay varias opciones disponibles, como "Reliable Delta Compressed" (compresión delta confiable), que garantiza la sincronización precisa pero puede tener un mayor coste de ancho de banda, y "Unreliable" (no confiable), que es más rápido pero puede dar lugar a diferencias entre los jugadores.

Una vez configurado el Photon View en el objeto, se pueden agregar componentes adicionales para definir el comportamiento y la apariencia del objeto en el juego. Estos componentes pueden incluir controladores de movimiento, animaciones, colisiones, etc.

Cuando se inicia el juego y los jugadores se conectan a la sala, los objetos con Photon Views se sincronizarán automáticamente entre los jugadores. Esto significa que los cambios realizados en un objeto por un jugador se reflejarán en los objetos de los demás jugadores. Por ejemplo, si un jugador mueve un objeto con un Photon View, los demás jugadores verán el objeto moverse de acuerdo con las actualizaciones enviadas a través de Photon Cloud.

Es importante tener en cuenta que los Photon Views sólo sincronizan las propiedades y el estado de los objetos, no los scripts o lógica personalizada. Por lo tanto, si se desea que un objeto realice acciones

específicas en respuesta a eventos de red, es necesario implementar la lógica adicional en los scripts asociados al objeto y utilizar métodos de llamada remota (RPC, por sus siglas en inglés) para comunicarse entre los objetos de diferentes jugadores.

Antes de describir la implementación del juego colaborativo propiamente dicho, es necesario describir dos conceptos claves de Photon como son *Lobby* y *Room* (sala). El *Lobby* es un espacio virtual donde los jugadores pueden reunirse antes de ingresar a una sala de juego. Es como un vestíbulo o sala de espera donde los jugadores pueden interactuar y tomar decisiones antes de comenzar una partida. En el *Lobby*, los jugadores pueden ver la lista de salas disponibles, así como información relevante sobre ellas, como el número de jugadores actualmente presentes, la configuración del juego, etc. También pueden crear sus propias salas o unirse a una sala existente. Por lo que respecta a *Room*, es el espacio virtual donde tiene lugar la partida o juego multijugador. Una vez que los jugadores se unen a una sala desde el *Lobby*, pueden interactuar y jugar juntos en ese entorno específico. La *Room* puede ser un espacio privado o público, dependiendo de la configuración. Puede haber múltiples salas simultáneas en ejecución, y cada una de ellas puede tener diferentes reglas, configuraciones y límites de jugadores. Dentro de una *Room*, los jugadores pueden interactuar y compartir información en tiempo real. Pueden sincronizar datos, como la posición de los personajes, acciones del juego, puntuaciones, chat, etc. Además, Photon proporciona herramientas y funciones para facilitar la comunicación y sincronización de los jugadores en la sala.

En PIUM se hace uso de la arquitectura que se acaba de describir para el desarrollo del multijugador, así, el jugador, desde el *Lobby*, al seleccionar la sala a la que desea conectarse, comprobará que exista dicha sala, en caso de no existir, la crearía y comenzaría la partida. En caso contrario, si no hubiera comenzado la partida, se iniciaría, como se observa en la Figura 20.

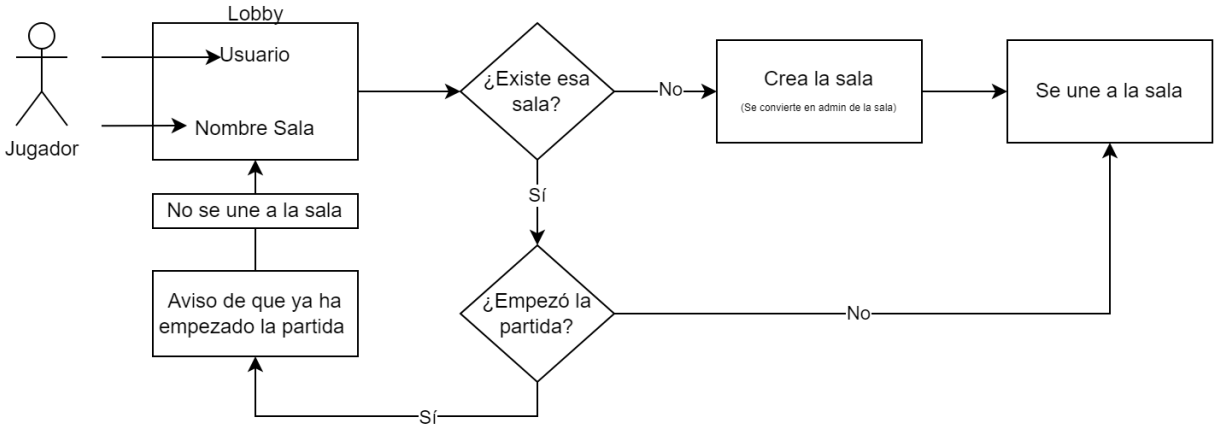


FIGURA 20: DIAGRAMA DE CÓMO SE REALIZA LA CONEXIÓN. FUENTE: PROPIA.

Pseudocódigo de la implementación:

```
Clase ControladorSala:
    Método OnConnectedToMaster():
        Llamar a OnConnectedToMaster() de la clase base
        Habilitar el botón de unirse

    Método OnCreatedRoom():
        Llamar a OnCreatedRoom() de la clase base
        Generar un número aleatorio (semilla) entre 1000 y 9999
        Establecer la semilla
        Declarar que no se ha empezado partida

    Variable customPropertyName = "EnPartida"

    Método conectarseSala():
```

```
        Obtener el nombre de usuario introducido por el usuario y convertirlo a minúsculas
        Mostrar el mensaje "Bienvenido: " + nombre del usuario
        Obtener el nombre de sala introducido por el usuario y convertirlo a minúsculas

        Si el estado del cliente de red es conectado al Servidor Master:
            Unirse al lobby

    Método OnJoinedLobby():
        Llamar a OnJoinedLobby() de la clase base
        Unirse a una sala existente con el nombre de sala actual
        Mostrar mensaje: Conectados y unidos a un lobby + nombre de la sala

    Método OnJoinedRoom():
        Si la sala actual tiene la partida empezada:
            Mostrar mensaje: Ya hay una partida en curso
            Salir de la sala
            Mostrar el error
            Finalizar el método

        Llamar a OnJoinedRoom() de la clase base
        Ocultar el botón de unirse
        Obtener el valor de la semilla de la sala
        Establecer la semilla del controlador

        Mostrar mensaje: Conectados y unidos a una sala

    Método desconectar():
        Si el cliente está en una sala:
            Salir de la sala

    Método volverOffline():
        Desconectar de Photon Network

    Método OnLeftRoom():
        Llamar a OnLeftRoom() de la clase base

        Si no se ha empezado la partida y el estado actual de la partida es lobby:
            Mostrar el botón de unirse
            Ocultar el botón de desconectar

    Método cargarEscenaPartida(a):
        Cambiar el estado de la partida a partidaEstado

    Método empezarPartida():
        Declarar que se ha empezado partida
        Ejecutar cargarEscenaPartida()
```

También se adjunta el código en el anexo 5.

Cabe destacar que el jugador podrá seleccionar un nombre de usuario que se mostrará al resto de jugadores en la sala antes de empezar la partida.

2.6.3 Puntuación

Antes de describir la implementación del sistema de puntuación y su gestión en red, es necesario describir la configuración inicial de la gestión en red con LootLocker. Así, se crea una cuenta en LootLocker y se obtienen las credenciales de API necesarias para acceder a su plataforma. Seguidamente, se importa y configura el SDK de LootLocker en el proyecto de Unity.

Una vez configurado el proyecto, se pasa a realizar la gestión de los puntos que se consiguen durante la partida. Los enemigos devuelven una cantidad de puntos dependiendo de su “poder”, éste es calculado según su vida y su daño por segundo. El daño por segundo depende del daño, la velocidad de ataque y cuantas balas dispara. En definitiva, se usa la fórmula especificada en la sección 2.5.6 del presente documento.

Además, el juego cuenta con un sistema de combo para aumentar la cantidad de puntos conseguidos. El combo aumenta por cada enemigo que se derrote en relación 1:1. El combo bajará cuando el jugador reciba un golpe. Por cada golpe recibido se le quitará 1 del combo total.

El combo afecta en cuanto a los puntos dados por los enemigos de forma multiplicativa, es decir, si se tiene un combo de 72, la puntuación que daría el enemigo sería los puntos base multiplicados por el combo, en este caso 72. Un ejemplo puede ser si un enemigo tiene 100 puntos base y se le mata con un combo de 72 daría 7200 puntos.

Al terminar la partida se le pregunta al jugador si quiere publicar la puntuación y con qué nombre quiere hacerlo. En ese momento el programa realizará la subida, que en caso de que falle avisará al jugador para que, si lo desea, lo vuelva a intentar subir. Para realizar la publicación se tienen que realizar varias acciones, en primer lugar, se hace una llamada a la API de LootLocker para establecer una conexión con la plataforma, concretamente llamando al método estático LootLockerSDK.Initialize() y proporcionando las credenciales de API necesarios, como la clave del cliente y la clave del jugador. La clave de cliente se obtiene tras crear una cuenta en LootLocker, creando un proyecto y en la sección de ajustes se encuentra la clave de cliente, en cuanto a la de jugador se utiliza una clave de invitado estática para todos los jugadores. Tras ello, se autentica el juego con las credenciales de API proporcionadas por LootLocker, para ello, se llama al método LootLockerSDK.Session.StartSession() pasándole las credenciales del jugador, como su ID como parámetros en esta llamada, que en este caso sería la de invitado como anteriormente se menciona. Por último, se hace una llamada a la API de LootLocker para enviar la puntuación del jugador al sistema utilizando el método LootLockerSDK.Scores.SubmitScore(). Se le proporciona el ID del jugador, la ID de la clasificatoria y la puntuación obtenida como parámetros en esta llamada. Esto permitirá registrar y guardar la puntuación del jugador en LootLocker.

En caso de que el jugador no tenga conexión a internet en el momento de subir los puntos, éstos se guardarán de forma local en un archivo de texto, y en la próxima ocasión que se requiera subir datos, lo intentará automáticamente.

2.6.4 Control del personaje

El control del personaje funciona mediante el sistema de trazado de rayos (Physics.Raycast) y el sistema de *pathfinding* (NavMesh) de Unity (Unity, 2019, 2023). El usuario pincha con el botón principal del ratón sobre la pantalla, dicha posición junto con la de la cámara virtual, proporcionan la dirección en la que se traza el rayo (Physics.Raycast), el cual indica, mediante detección de colisiones, si se ha tocado algún elemento interactuable. Así, si el rayo colisiona con una superficie que sea navegable, se actualiza la posición del marcador de destino con el punto de colisión. Dicho marcador de destino es la pieza que está siguiendo constantemente la nave del jugador.

El NavMesh se crea durante la generación del mapa de cada ronda con un sistema de NavMesh Obstacles, el cual hace “agujeros” en la superficie navegable.

El sistema de combate se ejecuta automáticamente cuando el jugador se acerca a un enemigo que sea visible, es decir, cuando no haya paredes que se interpongan entre ellos. Así, al aparecer en escena varios enemigos visibles, se elegirá al más cercano y la nave comenzará a instanciar balas en la dirección del mismo respetando la dispersión que posea el jugador en dicho momento.

Para el control de Android se introdujo un *joystick* virtual en pantalla, el cual permitía que el jugador desplazase a voluntad su personaje. Como se puede ver en la figura 21, es un *joystick* desplazable, de forma que el movimiento de éste sobre la circunferencia se traduce en una dirección y una magnitud a aplicar sobre el desplazamiento del jugador.

FIGURA 21: CAPTURA DE PANTALLA EN ANDROID. FUENTE: PROPIA

2.6.5 Mejoras

Para la gestión del sistema de mejoras se ha empleado un archivo CSV, ubicado en la carpeta *Resources*, el cual almacena todas ellas junto a su grado de consecución. Inicialmente, en la fase de programación, se añaden manualmente todas como no conseguidas. El juego las lee y se configura en consecuencia, además, va actualizándolas en función de su nivel de consecución.

Cada mejora está ligada a una ruta física en el juego, dicha ruta se desbloquea al alcanzar un logro específico.

Las mejoras cambian propiedades visuales de la nave al ser obtenidas, como por ejemplo, las balas añaden proyectiles de diferentes colores o los chasis cambian el color del material de la nave.

Existen mejoras que solo se podrán obtener si se tiene una mejora previa y ésta actualiza los beneficios de la anterior. Así, esta circunstancia se representa en la pantalla de victoria entre rondas, mediante unas cartas presentadas en forma de una escalera de color, desde el blanco hasta el negro. La progresión cromática incluye los siguientes colores: blanco, verde, azul, morado, amarillo, rojo y negro.

2.6.6 Web de resultados

En cuanto a la página web, se ha utilizado un sistema de alojamiento basado en Firebase, el sistema de alojamiento de Google (Google, 2023). El cual es un servicio gratuito si no se rebasa cierto límite de tráfico. El plan Spark (gratuito) es un plan básico que ofrece 1 GB de almacenamiento total, 20.000 escrituras/día, 50.000 lecturas/día y 20.000 eliminaciones/día (Google, 2023). Los cuales no son un factor limitante en el entorno de este TFG.

La web consta de 4 páginas que son:

Página de Inicio: Representa la pantalla principal del juego y tiene como objetivo dar la bienvenida a los usuarios y proporcionarles una visión general del juego. Utilizando HTML y CSS personalizado, se crea una estructura y diseño visual que captura la esencia y la temática del juego. Esta página incluye elementos como el logotipo del juego, un encabezado atractivo y elementos visuales que llamen la atención, como imágenes relacionadas con el juego. Además, incorpora opciones de navegación para que los usuarios puedan acceder fácilmente a otras secciones del sitio web, como el ranking, la información del juego y la wiki de objetos y naves.

Página de Información: Dedicada a proporcionar detalles sobre el juego. Aquí se incluye una descripción general del juego, características clave y otra información relevante. Para realzar la presentación visual, se utiliza HTML y CSS personalizado para estructurar y diseñar la página. Se emplean elementos multimedia, como imágenes promocionales y capturas de pantalla, para ilustrar visualmente el juego. En particular, se destaca el tráiler del juego como el primer objeto en la página, utilizando un *player* de video para mostrarlo. Además, se utilizan estilos y efectos CSS para resaltar aspectos importantes y garantizar una presentación atractiva y agradable para los visitantes.

Página de Ranking: Encargada de mostrar la clasificación de los jugadores según su rendimiento en el juego. Con HTML y CSS personalizado, se crea una estructura de tabla donde se despliegan los nombres de los jugadores y sus respectivas puntuaciones. Para obtener los datos del ranking, se utiliza la API de LootLocker basada en JavaScript. Mediante llamadas a la API, se consulta la información del ranking y se muestra de manera dinámica en la página web cada vez que se carga. Esto se logra mediante la manipulación del DOM utilizando JavaScript para insertar los datos del ranking en la estructura HTML previamente definida. Además, de aplicar estilos y formatos CSS para mejorar la legibilidad y la apariencia visual del ranking.

Página de la Wiki de Objetos y Naves: Basada en el archivo CSV que contiene información detallada sobre los objetos y naves presentes en el juego. Utilizando HTML, CSS y JavaScript personalizado, se realiza una lectura del archivo CSV utilizando JavaScript para extraer los datos y procesarlos. Luego, se genera dinámicamente el contenido de la página web utilizando JavaScript, creando tarjetas para cada objeto o nave con sus respectivas imágenes, descripciones y atributos. Mediante el uso de estilos CSS, se mejora la presentación visual y la experiencia de navegación para los usuarios.

En el anexo 5 se puede ver el código generado para crear esta página.

2.7. Testeo

Una vez implementado el videojuego, se ha llevado a cabo un proceso de testeo para asegurar que todas las funcionalidades y mecánicas del juego funcionan correctamente. Se han realizado pruebas de jugabilidad, pruebas de compatibilidad con diferentes plataformas y pruebas de integración con la plataforma de ranking web. Así, el plan de testeo quedó configurado con la siguiente estructura:

1. Objetivos:
   - Evaluar la funcionalidad y el rendimiento del videojuego en un entorno multijugador.
   - Identificar y solucionar posibles problemas de estabilidad, conexiones y sincronización entre los jugadores.
   - Verificar que todas las características y modos de juego multijugador funcionen correctamente.
   - Evaluar la experiencia del usuario y la jugabilidad en un entorno de juego en línea.
2. Configuración del entorno:
   - Entorno 1: Dos jugadores en la misma red local
   - Entorno 2: Dos jugadores conectados a diferentes conexiones de internet
   - Entorno 3: Dos jugadores conectados mediante redes de telefonía
   - Entorno 4: Veinte usuarios simultáneos

Los resultados de estas pruebas fueron satisfactorios, con retardos inapreciables para el usuario del juego.

3. Pruebas funcionales:
   - Verificar la conexión de los jugadores al entrar al juego.

- Probar la funcionalidad básica del juego, como movimiento, acciones y mecánicas específicas del multijugador.
- Evaluar la sincronización de los eventos del juego entre los jugadores.
- Comprobar la estabilidad de la partida y la detección de errores en situaciones de alta carga de jugadores.

En este caso, las pruebas funcionales revelaron algún problema menor en la funcionalidad básica, que fue solventado en la fase de desarrollo. Por lo que respecta a la sincronización, no se apreciaron problemas. Cabe destacar que nunca se superó el número de 20 jugadores simultáneos.

4. Pruebas de rendimiento:
    - Evaluar el rendimiento del juego en diferentes configuraciones de hardware y sistemas operativos.

Las pruebas de rendimiento se llevaron a cabo tanto en máquinas con Windows 10, como Windows 11, algunas de las especificaciones fueron, procesador i3 6100 y 4GB, procesador i7 10750H y 16 GB de RAM y un i5 6500 con 8GB de RAM. No se apreciaron diferencias de rendimiento entre estos dispositivos.

En cuanto a teléfonos móviles se probaron varios dispositivos Android. Se probó en un Pixel 6A, Samsung Galaxy Note 20, Samsung Galaxy S21, Xiaomi Redmi Note 7 y 10. En ninguno de estos se encontraron problemas de rendimiento salvo por alguna bajada puntual en el caso del Xiaomi Redmi Note 7.

5. Pruebas de compatibilidad:
    - Comprobar la compatibilidad entre diferentes sistemas operativos Windows.
    - Comprobar la compatibilidad entre Windows y Android.
    - Comprobar la compatibilidad con diferentes controladores y dispositivos de entrada.

Las pruebas de compatibilidad a nivel sistema operativo no presentaron problemas, ni siquiera cuando la partida tenía lugar entre sistemas Windows y Android. En cuanto a los dispositivos de entrada, el juego está diseñado para usarse con ratón y teclado, sin embargo, también se probó un *joystick* tipo Xbox 360 y un *joystick* simple. Como resultado, ninguno de los *joystick* funcionó con el juego, mientras que el ratón no presentó problemas.

6. Pruebas de usabilidad:
    - Evaluar la experiencia del usuario y la interfaz de usuario en el modo multijugador.
    - Realizar pruebas de usabilidad para asegurar que los jugadores comprendan las funcionalidades y controles del juego.
    - Recopilar comentarios de los jugadores sobre la jugabilidad, la interfaz y las características del multijugador.

Para evaluar la usabilidad se utilizó un cuestionario SUS (System Usability Scale) basado en el Brooke, 1996, el cual ha sido ampliamente utilizado en la evaluación de la experiencia del usuario (Broke, 1996). Este cuestionario se puede ver en el anexo 7. A continuación se muestran de forma gráfica los resultados obtenidos con dicho cuestionario en una población de 12 sujetos.

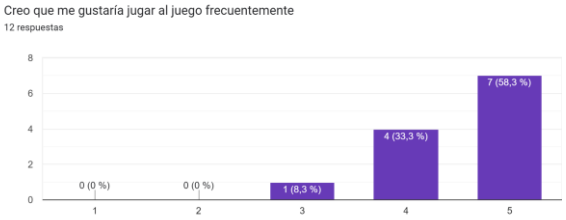


FIGURA 22: PROPENSIÓN A JUGAR CON FRECUENCIA. FUENTE: PROPIA

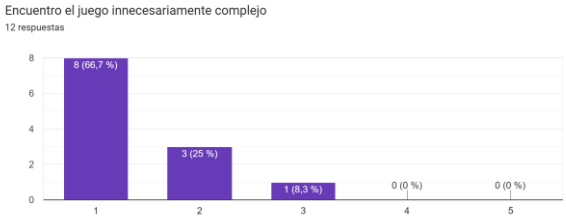


FIGURA 23: OPINIÓN SOBRE LA COMPLEJIDAD. FUENTE: PROPIA

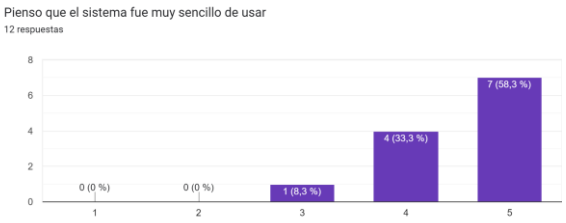


FIGURA 24: OPINIÓN DE LA FACILIDAD DE USO. FUENTE: PROPIA

FIGURA 25: OPINIÓN DE LA NECESIDAD DE UN EXPERTO. FUENTE: PROPIA

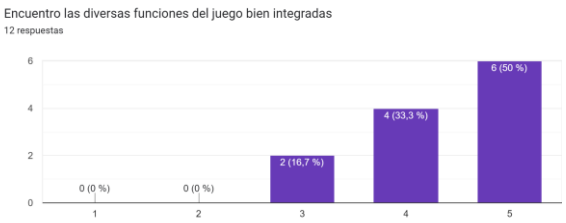


FIGURA 26: FUNCIONES BIEN INTEGRADAS. FUENTE: PROPIA

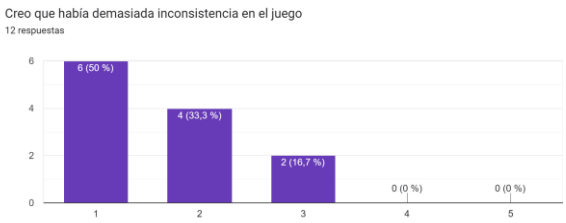


FIGURA 27: INCONSISTENCIA EN EL JUEGO. FUENTE: PROPIA

FIGURA 28: OPINIÓN SOBRE SI ES SENCILLO APRENDER A JUGAR. FUENTE: PROPIA

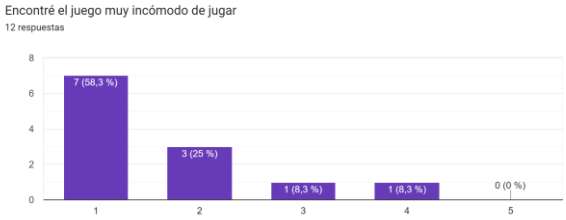


FIGURA 29: OPINIÓN DE INCOMODIDAD. FUENTE: PROPIA

FIGURA 30: CONFIANZA DURANTE EL JUEGO. FUENTE: PROPIA

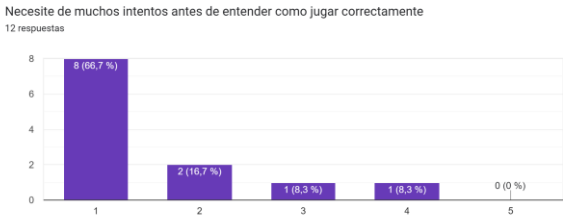


FIGURA 31: MUCHOS INTENTOS NECESARIOS. FUENTE: PROPIA

Según los resultados observados el valor medio de la encuesta resultó en un 86,53 de un máximo de 100. Este cálculo viene dado por el valor de las preguntas 1, 3, 5, 7 y 9 restando 1 al valor, 5 menos el valor de las preguntas 2, 4, 6, 8 y 10 y todo esto es sumado y multiplicado por 2,5.

Sin embargo, como vemos en las figuras 29 y 31 la incomodidad en cuanto a la jugabilidad y la necesidad de varios intentos son los puntos más bajos pese a tener una buena media.

2.8. Resultados

Tras el proceso de desarrollo y testeo, se ha obtenido el videojuego multijugador PIUM, que cuenta con un sistema de niveles generados de forma procedural, un sistema de combate fluido y desafiante, un modo multijugador en línea y un sistema de ranking web. El juego ha sido probado con éxito en diferentes plataformas y ha demostrado su funcionalidad y jugabilidad.

A continuación, se mostrarán diferentes pantallas del resultado.

La pantalla de título, mostrada en la figura 29, será la primera pantalla que observará el jugador. En la parte superior izquierda, se encuentra el nombre del juego. Como fondo, se puede observar una imagen del propio juego, que transmite la temática y el estilo visual del mismo. Desde esta pantalla se puede acceder a las demás, de izquierda a derecha, cerrar el juego (rojo), lista de logros (verde), tienda (morado), abrir el menú de selección de modo (amarillo), ajustes (azul) y abrir la página con la clasificatoria (naranja). Además, la iluminación irá cambiando en una especie de ciclo día-noche para dar un poco de dinamismo a la pantalla.

FIGURA 29: PANTALLA DE TÍTULO DE PIUM. FUENTE: PROPIA

En la pantalla de logros, mostrada en la figura 30, se muestran todos los disponibles en el juego. Para diferenciar los logros que se han obtenido o no, se mostrará el icono oscurecido, así como el título "Logro no desbloqueado" para aquellos logros que el jugador no haya desbloqueado. Los requisitos serán siempre visibles para que así el jugador sepa cómo desbloquearlos.

FIGURA 30: PANTALLA DE LOGROS DE PIUM. FUENTE: PROPIA

La pantalla de selección de tiendas, mostrada en la figura 31, presenta las dos pantallas disponibles a las que puede acceder el jugador, así como una representación gráfica del contenido que se ofrece.

FIGURA 31: PANTALLA DE TIENDAS DE PIUM. FUENTE: PROPIA

En la tienda de naves, figura 32, se muestra la imagen del modelo de la nave. En el caso de no tenerla adquirida, el precio estaría situado en la parte superior del cartel que pone “Activa”.

FIGURA 32: PANTALLA DE LA TIENDA DE NAVES DE PIUM. FUENTE: PROPIA

Al pulsar el botón de comprar se resta el precio de la nave a los puntos del jugador, en caso de que éstos fueran menores que el precio de venta, no se podría efectuar la compra. A la derecha se muestran los atributos de la nave para así poder decidir qué nave se prefiere. En la figura 33 podemos ver un caso de una nave no adquirida.

FIGURA 33: PANTALLA DE TIENDA DE NAVES CON LA NAVE SIN DESBLOQUEAR DE PIUM. FUENTE: PROPIA

La tienda de sombreros, mostrada en la figura 34, es idéntica a la tienda de naves, con la excepción de las estadísticas, puesto que, se tratan de opciones cosméticas solamente.

FIGURA 34: PANTALLA DE LA TIENDA DE SOMBREROS DE PIUM. FUENTE: PROPIA

En la pantalla de ajustes se muestran dos sliders, figura 35, con los que el usuario puede ajustar los volúmenes de la música y los efectos de forma independiente.

FIGURA 35: PANTALLA DE AJUSTES DE PIUM. FUENTE: PROPIA

Una vez pulsado el botón de empezar, se mostrará el panel de selección del modo un jugador o multijugador. Es un menú desplegable que se abre por la derecha. En la figura 36 se muestra cómo quedó el menú.

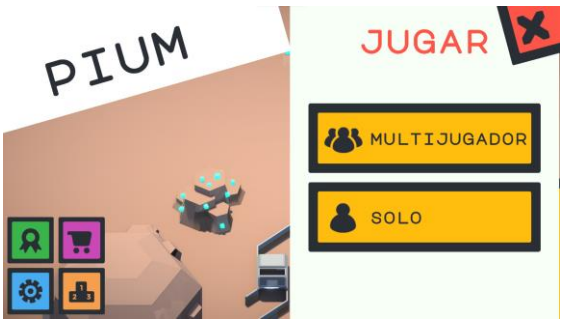


FIGURA 36: PANTALLA DE SELECCIÓN DE MODO DE PIUM. FUENTE: PROPIA

Una vez seleccionado el modo multijugador, se accederá a la pantalla del *Lobby*. Aquí es donde el usuario puede seleccionar el nombre que desea emplear y la sala (*Room*) a la que conectarse. En la figura 37 se ve como en la parte izquierda está la zona de configuración y los botones de salir y unirse en la derecha.

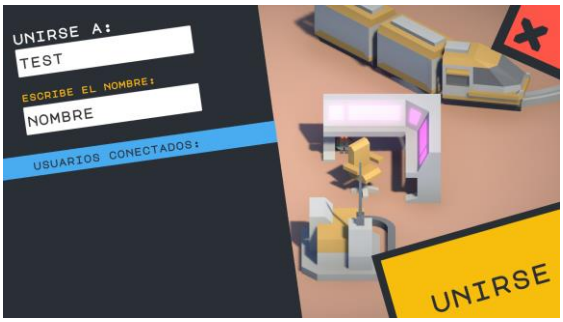


FIGURA 37: PANTALLA DE *LOBBY* DE PIUM. FUENTE: PROPIA

Una vez unidos a una sala, aparecerán los usuarios conectados y el administrador de la sala podrá empezar la partida. En la figura 38 se muestra la vista del administrador unido a una sala con dos personas.

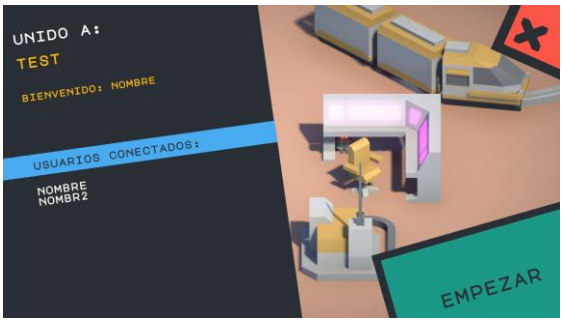


FIGURA 38: PANTALLA DE *LOBBY* CON VARIOS JUGADORES DE PIUM. FUENTE: PROPIA

Al completar cada ronda, el usuario podrá seleccionar una de tres mejoras. Se muestran en formato de cartas con el icono de cada mejora, las estadísticas que aumentan o disminuyen y cuál es la siguiente mejora. En la figura 39 se muestra dicha pantalla.

FIGURA 39: PANTALLA DE SELECCIÓN DE MEJORA DE PIUM. FUENTE: PROPIA

Una vez iniciada la partida se verá la GUI con los puntos, el minimapa, la vida y el botón de pausa, en la figura 40 se puede apreciar dicha interfaz.
Se mostrará la vida de los enemigos en color rojo sobre los mismos.

FIGURA 40: PANTALLA DENTRO DE LA PARTIDA DE PIUM. FUENTE: PROPIA

Cuando se juega en modo multijugador, se muestra la vida de los otros usuarios en verde para dar información de cómo está el pelotón y así poder diferenciarlos de los enemigos. En la figura 41 se muestra un fotograma.

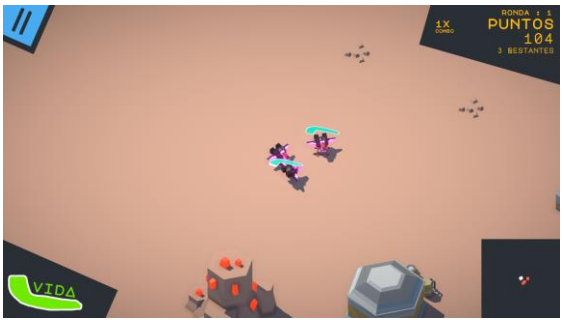


FIGURA 41: PANTALLA DENTRO DE LA PARTIDA CON VARIOS JUGADORES DE PIUM. FUENTE: PROPIA

Según si se completa la partida o si se es derrotado, cambia el cartel de la izquierda que se ve en la figura 42.
En la derecha muestra la clasificatoria en torno a los puntos del jugador y la opción de publicar la puntuación.

FIGURA 42: PANTALLA DE FINAL DE PARTIDA DE PIUM. FUENTE: PROPIA

En cuanto a la página web, se muestra una *landing page* con una estética similar al juego. En la figura 43 se muestra la página, en la que aparece el nombre del juego arriba a la izquierda, abajo a la izquierda se muestran los botones para ir a la zona de descargar el juego (amarillo), la wiki con la información del juego (azul) y la clasificatoria (morado). En la parte inferior derecha está el botón de jugar que te manda a la página con información. Ver anexo 4.

FIGURA 43: LANDING PAGE DE LA WEB DEL JUEGO. FUENTE: PROPIA

La página de la clasificatoria muestra las puntuaciones de los jugadores. En la figura 44 se muestra la clasificatoria a fecha de 4 de julio de 2023.

FIGURA 44: PÁGINA CON LA CLASIFICACIÓN DE PIUM. FUENTE: PROPIA

Para más resultados ver anexo 6 donde se encuentra un enlace a un *gameplay* de una partida multijugador.

3. Conclusiones y trabajos futuros

Se ha realizado una revisión teórica exhaustiva sobre plataformas para la creación de videojuegos, herramientas disponibles para crear videojuegos multijugador y sistemas para almacenaje y gestión de puntuaciones en sistemas colaborativos. Esta investigación nos ha permitido obtener conocimientos fundamentales y seleccionar las herramientas Unity, PUN y LootLocker como las más adecuadas para el desarrollo del videojuego PIUM.

Se ha llevado a cabo un estudio detallado y en profundidad de uno de los algoritmos más populares en cuanto a generación, “Wave Function Collapse”, que permiten generar niveles de forma automática y aleatoria. Esto ha sido esencial para implementar el sistema de generación procedural de niveles.

Mediante el análisis de videojuegos existentes, se ha obtenido una perspectiva amplia de las fortalezas y áreas de oportunidad en el diseño y desarrollo del juego. Esto nos ha permitido mejorar el diseño de PIUM, incorporando elementos exitosos de otros juegos y adaptándolos a nuestra propia propuesta.

Se ha realizado una planificación detallada de las fases de desarrollo del videojuego. Esta planificación nos ha permitido gestionar eficientemente los recursos y cumplir con los plazos establecidos para cada etapa del desarrollo.

Se ha logrado implementar tanto el videojuego como el sistema de gestión de puntuaciones en red. Esto nos ha permitido crear una experiencia multijugador dinámica y conectar a los jugadores a través de la competencia en tiempo real con la visualización de las puntuaciones más altas en una página web dedicada.

Finalmente, se ha sometido el videojuego a pruebas con usuarios para evaluar su jugabilidad, nivel de reto y experiencia de juego. Las pruebas han permitido identificar áreas de mejora y realizar ajustes en la confianza e incomodidad a la hora de jugar para ofrecer una experiencia de juego más satisfactoria. Además, en las pruebas de usabilidad revelaron que el juego presenta fortalezas en términos de rejugabilidad, sencillez y consistencia. No obstante, se identificaron áreas de mejora en la accesibilidad. Estos hallazgos proporcionan una base sólida para realizar ajustes y mejoras con el objetivo de optimizar la experiencia del usuario en futuras versiones del juego.

Por lo que respecta a trabajos futuros, se tiene planteado añadir más contenido (DLC) del tipo: mejoras, enemigos, naves, cosméticos e incluso mapas para así seguir dando posibilidades al jugador de descubrir nuevas combinaciones. Con la implementación de este sistema se logrará aumentar la

flexibilidad y la longevidad del videojuego. Asimismo, se podría publicar el videojuego en plataformas de distribución ampliamente utilizadas, como son Steam y Play Store. La publicación en estas plataformas permitiría alcanzar una audiencia más amplia y generar ingresos a través de las ventas del juego.

También se podrían organizar competiciones dentro del videojuego para promover la participación de los jugadores, fomentar la competencia y mejorar la experiencia de juego. Las competiciones pueden incluir torneos, desafíos o eventos especiales en los que los jugadores compiten entre sí por premios y reconocimiento.

Por otro lado, se podrían explorar oportunidades de colaboración con empresas o la licencia del juego para adaptaciones a otros medios, como juguetes, merchandising, entre otros. Estas colaboraciones y licencias pueden generar ingresos adicionales y aumentar la visibilidad del juego, expandiendo su alcance más allá del ámbito digital.

Por último, se podría crear la versión para iOS del juego, la cual, debería ser optimizada para mejorar el rendimiento en dispositivos con características más reducidas, aumentando así la base de jugadores.

4. Referencias bibliográficas

Berg, T. (2021). Leaderboards. https://lootlocker.com/blog/leaderboards [Consulta: 26 de Enero de 2023]

Brooke, J. (1996). SUS: a "quick and dirty" usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.), Usability Evaluation in Industry (pp. 189-194).

Craddock, D. L. (2021). Dungeon Hacks: How NetHack, Angband, and Other Rougelikes Changed the Course of Video Games. CRC Press. "The History of Rogue: Have @ You, You Deadly Zs" de Bill Loguidice, publicado en Game Developer.

COCOS. (2019). Cocos Creator accounts for over 70% of WeChat's top mini games. https://www.cocos.com/en/post/cocos-creator-accounts-for-over-70-of-wechats-top-mini-games [Consulta: 16 de Enero de 2023]

Dacuba, S. (2020) Netcode for GameObjects. https://docs-multiplayer.unity3d.com/netcode/current/about/ [Consulta: 14 de Enero de 2023]

Dormans, J. (2012). Video Game Design: Principles and Practices from the Ground Up. A K Peters/CRC Press.

Epic Games (2004). Epic Developer Community. https://dev.epicgames.com/community/ [Consulta: 15 de Enero de 2023]

Epic Games (2023). Unreal Engine End User License Agreement. https://www.unrealengine.com/es-ES/eula-reference/unreal-es-es [Consulta: 15 de Enero de 2023]

Galante, L.. (2022). Vampire Survivor https://poncle.itch.io/vampire-survivors [Consulta: 3 de Enero de 2023].

Google (2023). Firebase. https://firebase.google.com/?hl=es [Consulta: 28 de Enero de 2023].

Google (2023). Google cloud– Marketplace. https://console.cloud.google.com/marketplace/product/google/games.googleapis.com?tutorial=toc&pli=1&project=skilled-anthem-295214 [Consulta: 01 de Febrero de 2023]

Google (2023). Servicios de juego de Google Play. https://developers.google.com/games/services?hl=es-419 [Consulta: 01 de Febrero de 2023]

Google Trends (2023). Interés a lo largo del tiempo del término Godot. https://trends.google.es/trends/explore?date=today%205-y&q=Godot&hl=es [Consulta: 15 de Enero de 2023]

Gumin, M. (2016). GitHub - WaveFunctionCollapse. https://github.com/mxgmn/WaveFunctionCollapse [Consulta: 23 de Enero de 2023]

Hopoo Games (2020). Risk of Rain 2. https://www.riskofrain.com/ [Consulta: 3 de Enero de 2023]

House, B. (2018). Evolving multiplayer games beyond UNet. Unity Blog. https://blog.unity.com/technology/evolving-multiplayer-games-beyond-unet [Consulta: 01 de Marzo de 2023]

Kenney (2020). Space kit. https://kenney.nl/assets/space-kit [Consulta: 26 de Noviembre de 2022]

Linietsky, J. & Manzur, A. (2007) Godot Engine. https://godotengine.org/ [Consulta: 15 de Enero de 2023]

Linietsky, J. & Manzur, A. (2014) Godot Engine Documentation. https://docs.godotengine.org/en/stable/community/tutorials.html [Consulta: 15 de Enero de 2023]

LootLocker (2020). Página oficial. https://lootlocker.com/ [Consulta: 26 de Enero de 2023]

LootLocker (2020).Precios. https://lootlocker.com/pricing [Consulta: 26 de Enero de 2023]

Mayer, J (2020). Unisave. https://unisave.cloud/ [Consulta: 01 de Febrero de 2023]

McMillen, E. & Himsl, F. (2022). The Binding of Isaac https://store.steampowered.com/app/113200/The_Binding_of_Isaac/ [Consulta: 3 de Enero de 2023].

Mischa (2014). Mirror Github https://github.com/MirrorNetworking/Mirror [Consulta: 01 de Marzo de 2023]

Opera Norways (2023). GX Games. https://gx.games/es/ [Consulta: 20 de Mayo de 2023]

Orús, A. (2023). Statista - Evolución del valor de mercado de la industria del videojuego en el mundo entre 2022 y 2027. Recuperado de https://es.statista.com/estadisticas/598622/valor-de-mercado-del-videojuego-en-el-mundo/ [Consulta: 23 de Maro de 2023]

Overmars, M. (1999) GameMaker Studio. https://gamemaker.io/es [Consulta: 14 de Enero de 2023]

Overmars, M. (1999) GameMaker Studio Precios. https://gamemaker.io/es/get [Consulta: 14 de Enero de 2023]

Photon Team (2023). Photon Engine - Documentation. https://doc.photonengine.com/fusion/current/getting-started/fusion-intro [Consulta: 01 de Marzo de 2023]

Photon Team (2023). Photon Unity Networking (PUN) - Photon Engine. Recuperado de https://www.photonengine.com/pun [Consulta: 01 de Marzo de 2023]

Rémy, P. & Marais, P. (2007). Shared Authority Networking Architecture for Real-time Multiplayer Games. En Proceedings of the 2007 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (pp. 173-176).

Ries, E. (2011). The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Publishing Group

Supergiant Games (2020). Hades. https://www.supergiantgames.com/games/hades/ [Consulta: 3 de Enero de 2023]

Unity Technologies (2019). Construyendo un NavMesh https://docs.unity3d.com/es/2019.4/Manual/nav-BuildingNavMesh.html [Consulta: 18 de Enero de 2023]

Unity Technologies. (2020). Multiplayer Networking. Recuperado de https://docs-multiplayer.unity3d.com/ [Consulta: 01 de Marzo de 2023]

Unity Technologies (2023). Physics.Raycast. https://docs.unity3d.com/ScriptReference/Physics.Raycast.html [Consulta: 17 de Enero de 2023]

Unity Technologies. (2023) Planes y precios. https://unity.com/es/pricing#plans-individualsand-teams [Consulta: 15 de Enero de 2023]

Wijman, T. (2021). The Games Market and Beyond in 2021: The Year in Numbers. https://newzoo.com/resources/blog/the-games-market-in-2021-the-year-in-numbers-esports-cloud-gaming#:~:text=The%20games%20market%20in%202021,%2B1.4%25%20over%20last%202020 [Consulta: 23 de Marzo de 2023]

Zapata, S. (2005) RogueBasin Wiki. https://www.roguebasin.com [Consulta: 17 de Enero de 2023]

5. Anexos

Anexo 1: ODS

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No procede
ODS 1. **Fin de la pobreza.**				■
ODS 2. **Hambre cero.**				■
ODS 3. **Salud y bienestar.**				■
ODS 4. **Educación de calidad.**		■		
ODS 5. **Igualdad de género.**				■
ODS 6. **Agua limpia y saneamiento.**				■
ODS 7. **Energía asequible y no contaminante.**				■
ODS 8. **Trabajo decente y crecimiento económico.**				■
ODS 9. **Industria, innovación e infraestructuras.**	■			
ODS 10. **Reducción de las desigualdades.**				■
ODS 11. **Ciudades y comunidades sostenibles.**				■
ODS 12. **Producción y consumo responsables.**				■
ODS 13. **Acción por el clima.**				■
ODS 14. **Vida submarina.**				■
ODS 15. **Vida de ecosistemas terrestres.**				■
ODS 16. **Paz, justicia e instituciones sólidas.**				■
ODS 17. **Alianzas para lograr objetivos.**	■			

TABLA 6: ALIENACIÓN DEL TFG/TFM CON LOS ODS

El proyecto de producción de un videojuego multijugador, titulado PIUM, se relaciona con varios Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030. Estos objetivos globales

establecidos por las Naciones Unidas abordan desafíos sociales, económicos y ambientales urgentes. En la tabla 6 se muestran las alineaciones.

ODS 4 - Educación de calidad: La revisión teórica sobre plataformas de creación de videojuegos y el análisis de videojuegos existentes contribuyen al conocimiento y aprendizaje en el campo del diseño y desarrollo de videojuegos, fomentando la educación y formación en esta área.

ODS 9 - Industria, Innovación e Infraestructura: El desarrollo de PIUM implica la utilización de tecnologías innovadoras, de hecho, requiere explorar nuevas herramientas, lenguajes de programación, motores de juego y técnicas de diseño que permitan crear una experiencia única y atractiva para los jugadores. A través de la investigación y aplicación de nuevas tecnologías, el TFG contribuye a la promoción de la innovación en el ámbito de los videojuegos.

ODS 17 - Alianzas para lograr los objetivos: La participación de personas en el testeo del juego implica una colaboración y asociación activa con los jugadores y la comunidad. Esta interacción permite obtener valiosos comentarios y opiniones para mejorar el juego y adaptarlo a las preferencias y necesidades de los usuarios. La participación de la comunidad aporta una perspectiva externa y ayuda a identificar áreas de mejora y oportunidades de crecimiento.

De esta manera, podemos determinar que el proyecto de producción del videojuego PIUM se alinea con varios Objetivos de Desarrollo Sostenible de la Agenda 2030, contribuyendo a la educación, la innovación y la colaboración para lograr un futuro más sostenible.

Anexo 2: Ejecutable Windows

https://piumpium.web.app/Pium.zip

Anexo 3: APK

https://piumpium.web.app/a.apk

Anexo 4: Página ranking

https://piumpium.web.app/

Anexo 5: Código

Anexo adjuntado: pium.pdf

Anexo 6: Gameplay

Enlace al gameplay subido a youtube: https://youtu.be/x-ZqZqr_PmE

Anexo 7: Cuestionario de usabilidad

Puntuados de 1 a 5 siendo 1 muy en desacuerdo y 5 muy de acuerdo.

1. Creo que me gustaría jugar al juego frecuentemente
2. Encuentro el juego innecesariamente complejo

3. Pienso que el sistema fue muy sencillo de usar
4. Creo que necesitaría la ayuda de un experto para poder jugar bien al juego
5. Encuentro las diversas funciones del juego bien integradas
6. Creo que había demasiada inconsistencia en el juego
7. Me imagino que a la mayoría de las personas les resultaría sencillo aprender a jugar
8. Encontré el juego muy incómodo de jugar
9. Me sentí muy confiado jugando al juego
10. Necesite de muchos intentos antes de entender como jugar correctamente

Anexo 8: Git del proyecto

https://github.com/Llambies/piumpium