Florida Universitària

Grado en Diseño y Desarrollo de Videojuegos y Experiencias Interactivas

Trabajo de Fin de Grado

Diseño y desarrollo de la mecánica principal para un videojuego del género FPS

Autor: Jorge Robledo Durendez

DNI: 24474772-N

Tutor: Francisco José Mollá Gandía

Fecha: 04/12/2024

Curso: 2024-2025

Titulación adscrita a

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

RESUMEN

Este Trabajo de Fin de Grado se centra en el diseño y desarrollo de una mecánica principal modular para un videojuego de género FPS (First-Person Shooter), integrando armas tradicionales, habilidades mágicas y reacciones dinámicas entre enemigos. La motivación principal surge del rediseño de un sistema basado en un proyecto previo que, debido a un alcance excesivamente ambicioso, no logró cumplir con los estándares esperados. Este nuevo enfoque prioriza la creación de sistemas modulares y fácilmente ampliables, entregando un prototipo funcional como base para futuros desarrollos.

La metodología empleada incluye un diseño modular utilizando ScriptableObjects y estructuras desacopladas para gestionar armas, habilidades mágicas, reacciones y comportamientos enemigos. Se utilizaron herramientas como Unity y Cinemachine para garantizar una experiencia de usuario inmersiva y dinámica. Entre los principales resultados destaca un sistema de reacciones capaz de generar interacciones como "Sobrecarga" (daño en área y empuje), "Electrocución" (daño en cadena) y "Congelación" (ralentización de enemigos), todas integradas en un entorno coherente.

La implementación de una consola de comandos personalizada facilitó la validación eficiente de las mecánicas mediante pruebas iterativas, permitiendo ajustes en tiempo real y mejorando el proceso de depuración. Este enfoque iterativo no solo asegura un desarrollo estructurado y escalable, sino que también enriquece la experiencia de juego al promover decisiones estratégicas y una interacción fluida.

En conclusión, este proyecto demuestra cómo la aplicación de principios de modularidad y una iteración constante puede resultar en un sistema de juego sólido, centrado en la funcionalidad y escalabilidad, estableciendo las bases para proyectos más ambiciosos en el futuro.

Palabras clave: FPS, modularidad, reacciones mágicas, sistema de armas, depuración, Unity.

ABSTRACT

This Final Degree Project focuses on the design and development of a modular core mechanic for an FPS (First-Person Shooter) video game, integrating traditional weapons, magical abilities, and dynamic reactions between enemies. The primary motivation stems from redesigning a system based on a previous project, which, due to an overly ambitious scope, failed to meet expected standards. This new approach prioritizes the creation of modular and easily extendable systems, delivering a functional prototype as a foundation for future developments.

The methodology employed includes a modular design using ScriptableObjects and decoupled structures to manage weapons, magic abilities, reactions, and enemy behaviors. Tools such as Unity and Cinemachine were utilized to ensure an immersive and dynamic user experience. Among the key outcomes is a reaction system capable of generating interactions such as "Overload" (area damage and pushback), "Electrocution" (chain damage), and "Freezing" (enemy slowdown), all seamlessly integrated into a cohesive environment.

A custom command console facilitated efficient validation of mechanics through iterative testing, enabling real-time adjustments and improving the debugging process. This iterative approach not only ensures structured and scalable development but also enhances the gameplay experience by fostering strategic decision-making and fluid interactions.

In conclusion, this project demonstrates how the application of modular design principles and constant iteration can result in a robust gameplay system, focused on functionality and scalability, laying the groundwork for more ambitious projects in the future.

Keywords: FPS, modularity, magical reactions, weapon system, debugging, Unity.

AGRADECIMIENTOS

Quisiera expresar mi más sincero agradecimiento a todas las personas que, de una manera u otra, han contribuido al desarrollo de este Trabajo de Fin de Grado y a mi crecimiento profesional a lo largo de este proceso.

En primer lugar, deseo agradecer a Francisco José Molla Gandia, mi tutor del TFG, por su invaluable apoyo, orientación y paciencia durante todo el proceso de este proyecto. Su experiencia, sugerencias y recomendaciones han sido fundamentales para mejorar la calidad tanto del documento como de la demo funcional presentada. Su guía me ha permitido superar los retos técnicos y conceptuales del proyecto.

Quisiera también expresar mi gratitud al tribunal del TFG anterior. Sus observaciones y críticas constructivas fueron esenciales para identificar los errores y carencias del proyecto original. Gracias a sus comentarios, adquirí una nueva perspectiva sobre el enfoque del trabajo, lo que me impulsó a rediseñar y optimizar el proyecto actual. Este aprendizaje no solo enriqueció este TFG, sino que también me brindó herramientas valiosas para proyectos futuros.

Por último, pero no menos importante, quiero agradecer profundamente a mi pareja, amigos y familia por su apoyo incondicional durante todo este proceso. Sus palabras de ánimo, comprensión en momentos de frustración y su constante motivación, sus consejos y recomendaciones han sido una fuente de energía y conocimiento para no rendirme y continuar adelante. Incluso en los momentos más complicados. Este proyecto no habría sido posible sin su respaldo y consejos.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

El género First Person Shooter (FPS) es uno de los pilares de la industria de los videojuegos, conocido por su acción dinámica y su capacidad para ofrecer experiencias inmersivas desde la perspectiva del jugador. Dentro de este género, el subgénero Boomer Shooter ha resurgido en popularidad, inspirado en los títulos de los años 90 como Doom y Quake. Estos juegos se caracterizan por su ritmo rápido, niveles intrincados y una jugabilidad que combina simplicidad en los controles con profundidad estratégica en el combate.

El objetivo principal de este Trabajo de Fin de Grado (TFG) es diseñar y desarrollar una mecánica innovadora que combine el uso de armas tradicionales con habilidades mágicas, pensada para integrarse en un videojuego de estilo FPS clásico o Boomer Shooter. La propuesta se centra en crear un sistema modular que sea escalable, entretenido y que permita al jugador experimentar nuevas dinámicas dentro de un entorno de combate frenético y estratégico.

Este proyecto surge como una reestructuración de un intento anterior en el que se desarrolló un videojuego completo con mecánicas similares. Aunque el juego se terminó en su totalidad, el resultado final no alcanzó la calidad esperada debido a una ejecución que carecía de profundidad y optimización en las mecánicas centrales. En esta nueva iteración, se ha decidido concentrar los esfuerzos en el diseño de una demo técnica que refine y mejore estas mecánicas, priorizando la funcionalidad y la escalabilidad como objetivos principales.

En resumen, este TFG busca aprovechar las lecciones aprendidas del proyecto anterior para desarrollar un sistema modular y funcional, centrado en las mecánicas de combate que definen la esencia de los FPS clásicos. Este enfoque garantiza una base sólida para futuros desarrollos y una experiencia de juego enriquecedora para los jugadores, alineándose con las características que han hecho del Boomer Shooter un género atemporal en la industria de los videojuegos.

2 ESTADO DEL ARTE

El desarrollo de videojuegos es un campo en constante evolución que refleja tanto avances tecnológicos como tendencias culturales. Desde sus inicios en los años 50 hasta la actualidad, los videojuegos han pasado de ser simples experimentos universitarios a una de las industrias de entretenimiento más importantes a nivel global. Este apartado explora los fundamentos históricos y técnicos que han influido en el diseño y desarrollo de este Trabajo de Fin de Grado, enfocándose en la evolución de los videojuegos, los géneros FPS, y el motor Unity como herramienta clave para su creación. Además, se incluirá un análisis de los "Boomer Shooters" y una breve descripción de títulos que sirven como referencia e inspiración para este proyecto.

2.1 Historia de los Videojuegos

El inicio de los videojuegos se podría decir que es relativamente reciente, dado que data de la década de los 1950, poco después de la aparición de los ordenadores. Seguido de esto y de forma experimental surgieron programas como Nimrod (1951) y OXO (1952) que son versiones de juegos tradicionales trasladados a lo electrónico. Los primeros videojuegos pioneros fueron Tennis for Two (1958) y Spacewar! (1962). A pesar de considerarse ya videojuegos, estos eran juegos muy simples y su uso estaba limitado a universidades o centros de investigación.

Junto a la aparición de los walkmans y los televisores llegaron las máquinas recreativas y con ellas los primeros videojuegos dirigidos al público. En esta línea, Computer Space (1971) y Pong (1972) fueron los que inauguraron las primeras máquinas recreativas. El auge y la popularidad que estas obtuvieron, provocó que las recreativas fueran comunes en bares y salones recreativos. Gracias a esto, los clásicos como Space Invaders (1978), Asteroids (1978) o Pac-Man (1980) marcaron lo que conocemos como la primera época de los videojuegos, creando un nuevo fenómeno sociocultural.

Hasta la década de los 1980, el mundo de los videojuegos estaba monopolizado por Atari y sus recreativas, pero desde Japón aparecieron dos nuevos competidores: Nintendo y SEGA. Estas dos empresas, que competían con Atari, se enfocaron en los videojuegos para los hogares con las consolas NES (1983) y la Master System (1983). Paralelo al auge de estas dos, aparecieron los primeros ordenadores domésticos como el Spectrum (1982) o el Commodore 64 (1982). Gracias a este gran fenómeno de los ordenadores domésticos, los videojuegos se convirtieron en una gran industria. A finales de dicha década

aparecieron las primeras consolas portátiles como la Game Boy (1989) que implicó un gran avance tecnológico.

En los 1990 hubo un gran salto de tecnología con la tecnología de 16-bit en consolas como la SNES (1990) y la Mega Drive (1988). Esta mejora en la tecnología trajo un avance gráfico significativo y un nuevo género de moda: los shooters en 3D. Con este gran cambio, empiezan a aparecer consolas nuevas de empresas que ya estaban en el sector de los videojuegos como es el caso de la Nintendo 64 (1996) o la Sega Saturn (1994). Pero esto también atrajo al sector nuevos competidores como Sony con la PlayStation (1994) o Microsoft con la Xbox (2001). El surgimiento de tantas consolas provocó que los videojuegos se volvieran más complejos, con narrativas más desarrolladas y gráficos más realistas.

La década de los 2000 vio la expansión masiva de los videojuegos en línea, con juegos como World of Warcraft (2004) y Call of Duty (2003). Las consolas también evolucionaron, con la PlayStation 2 (2000), Xbox 360 (2005), y Wii (2006) como líderes del mercado. Cabe destacar que la alta definición (HD) se convirtió en el estándar de la tecnología, llevando los gráficos y las experiencias de juego a nuevos niveles de realismo.

Finalmente, en la década de 2010, los videojuegos se diversificaron aún más con la llegada de los dispositivos móviles y la consolidación de las plataformas de juego como Steam (2003). Juegos como Angry Birds (2009) y Fortnite (2017) consolidaron los videojuegos móviles y los videojuegos multijugador en línea. Además, la realidad virtual y aumentada también comenzaron a ganar tracción con dispositivos como Oculus Rift () y juegos como Pokemon Go (2016). Junto a esto, el mercado de los eSports creció exponencialmente, convirtiéndose en una industria multimillonaria con una audiencia global.

(Facultat d'Informàtica de Barcelona (Universitat Politècnica de Catalunya) & L'Illa Diagonal, 2008)

2.2 Evolución de los Géneros: FPS

El origen de los First Person Shooter (FPS) puede rastrearse hasta los años 70 y 80, con juegos como "Maze War" (1973) y "Spasim" (1974), que son considerados los primeros ejemplos del género. Estos juegos, aunque primitivos en comparación con los estándares modernos, introdujeron la idea de la perspectiva en primera persona y el movimiento a través de un espacio tridimensional.

Sin embargo, el verdadero precursor del género fue "Wolfenstein 3D" (1992) desarrollado por id Software. Este juego estableció muchas de las convenciones que se convertirían en estándar para los FPS, como la exploración de laberintos, el combate contra enemigos y la recolección de objetos. "Wolfenstein 3D" fue un éxito comercial y crítico, y su motor de juego sirvió como base para futuros desarrollos.

La década de los 90 se considera la era dorada de los FPS, marcada por la innovación y la popularización del género. En 1993, id Software lanzó "Doom", un juego que revolucionó la industria con sus gráficos avanzados, jugabilidad rápida y soporte para mods creados por la comunidad. "Doom" no solo fue un hito técnico, sino que también estableció una base sólida para la cultura de los FPS.

Otro título influyente de esta época fue "Quake" (1996), también de id Software, que introdujo gráficos en 3D completamente renderizados y un enfoque en el multijugador en línea. "Quake" y su motor de juego dieron lugar a una serie de secuelas y derivados, y establecieron un estándar para los FPS multijugador.

Con la llegada del nuevo milenio, los FPS comenzaron a diversificarse en términos de ambientación, narrativa y mecánicas de juego. Juegos como "Half-Life" (1998) de Valve, que ofrecía una narrativa profunda y una experiencia inmersiva, demostraron que los FPS podían ser mucho más que simples juegos de disparos.

El lanzamiento de "Halo: Combat Evolved" (2001) para Xbox fue otro hito importante, popularizando el género en las consolas y estableciendo una franquicia duradera. "Halo" introdujo mecánicas innovadoras como el regenerador de salud y el uso de vehículos en combate.

Durante esta década, los FPS multijugador también ganaron popularidad. Juegos como "Counter-Strike" (1999) y "Battlefield 1942" (2002) se centraron en la experiencia competitiva en línea, mientras que "Call of Duty" (2003) comenzó una de las franquicias más exitosas y prolíficas en la historia de los videojuegos.

La década de 2010 vio un enfoque creciente en el realismo y la narrativa dentro de los FPS. Juegos como "Call of Duty: Modern Warfare" (2007) y sus secuelas ofrecieron experiencias cinematográficas y gráficos realistas, mientras que títulos como "Bioshock" (2007) combinaron elementos de FPS con narrativas profundas y ambientes detallados.

En años más recientes, el género ha continuado evolucionando con la introducción de juegos de servicio en vivo como "Destiny" (2014) y "Overwatch" (2016), que combinan elementos de RPG y juegos de disparos en un entorno multijugador continuo. Además, los juegos Battle Royale como "Fortnite" (2017) y

"Apex Legends" (2019) han redefinido el género con su enfoque en la supervivencia y la competencia masiva en línea.

(PCMag & Jensen, 2022)

2.3 Los "Boomer Shooters": Un Resurgimiento Clásico

Los *Boomer Shooters* representan un subgénero de los FPS que rinde homenaje a los clásicos de la década de 1990. Su nombre, aunque humorístico, hace referencia a la nostalgia de los jugadores veteranos que experimentaron la era dorada de los primeros shooters en primera persona. Este estilo de juego nació como una respuesta a los títulos modernos más centrados en narrativas cinematográficas y sistemas complejos, volviendo a las raíces del género con un enfoque en la acción rápida y el diseño simplificado pero efectivo.

El concepto de los Boomer Shooters se basa en títulos pioneros como:

- ***Doom* (1993):** Considerado uno de los padres fundadores del género, estableció un estándar para la acción frenética y el diseño de niveles con múltiples rutas y secretos escondidos.
- ***Quake* (1996):** Introdujo gráficos en 3D completamente renderizados y un enfoque en el combate multijugador que sentó las bases para el juego competitivo.
- ***Duke Nukem 3D* (1996):** Ofreció un tono humorístico y niveles interactivos, marcando una diferencia frente a los FPS serios de la época.

Durante los años 2000 y 2010, el género FPS evolucionó hacia experiencias más narrativas y gráficas, lo que dejó un vacío para los jugadores que preferían la acción pura y desenfrenada de los clásicos. Este vacío fue llenado por desarrolladores independientes que, aprovechando herramientas modernas como Unity y Unreal Engine, comenzaron a crear títulos inspirados en la era dorada de los FPS.

El renacimiento de los Boomer Shooters comenzó a finales de la década de 2010 con títulos que mezclaban mecánicas clásicas y tecnología moderna. Algunos ejemplos clave son:

- ***Dusk* (2018):** Inspirado en *Doom* y *Quake*, este título capturó la esencia de los shooters clásicos con gráficos minimalistas y un diseño de niveles retorcidos.

- ***Amid Evil* (2019):** Siguiendo la línea de *Heretic* y *Hexen*, introdujo armas mágicas y niveles visualmente impresionantes que respetan la estética retro.
- ***Prodeus* (2020):** Combinando pixel art y efectos modernos, *Prodeus* ofreció una experiencia visual única mientras mantenía la jugabilidad intensa de los años 90.

Sus características principales son:

1. **Jugabilidad Frenética:** Movimientos rápidos, precisión en el disparo y acción constante.
2. **Diseño de Niveles no Lineal:** Fomenta la exploración, con secretos escondidos y caminos alternativos.
3. **Estética Retro:** Gráficos pixelados o simplificados que evocan nostalgia.
4. **Enfoque en el Combate:** Poca narrativa, centrándose en la experiencia de disparar y sobrevivir.

El resurgimiento de los Boomer Shooters no solo revivió la nostalgia de los jugadores veteranos, sino que también introdujo este estilo a una nueva generación. Al adoptar herramientas modernas y técnicas de diseño actuales, estos juegos lograron mantener viva la esencia de los clásicos mientras atraían a audiencias más amplias. Títulos como *Dusk*, *Amid Evil* y *Prodeus* son ejemplos de cómo un enfoque simple y efectivo puede resonar en una industria que muchas veces prioriza la complejidad.

(JOT DOWN & Cuevas, n.d.)

2.4 Motor de Videojuegos utilizado: Unity

Unity es un motor de desarrollo de videojuegos multiplataforma, reconocido por su capacidad para crear experiencias 2D y 3D interactivas. Es ampliamente utilizado en la industria de los videojuegos, así como en otros campos como la realidad aumentada (AR), la realidad virtual (VR) y la simulación. Unity es conocido por su interfaz amigable y sus potentes herramientas que permiten a los desarrolladores diseñar, probar y publicar juegos en diversas plataformas.

2.4.1 Historia de Unity

Unity Technologies fue fundada en 2004 por David Helgason, Nicholas Francis y Joachim Ante. El primer lanzamiento, Unity 1.0, en 2005, se enfocó en desarrolladores para Mac OS. Con el tiempo, Unity se expandió a Windows y otras plataformas, convirtiéndose en un motor de referencia para el desarrollo multiplataforma.

El lanzamiento de Unity 2.0 en 2007 y la adición de soporte para iOS en 2008 ampliaron su base de usuarios. En 2010, Unity añadió soporte para Android, consolidando su posición en el desarrollo de juegos móviles. Unity 5, lanzado en 2015, trajo mejoras significativas en calidad visual y herramientas de desarrollo.

En 2023, Unity enfrentó controversia debido a cambios en su estructura de precios y modelo de licencias, lo que provocó una reacción negativa de la comunidad de desarrolladores. Las nuevas tarifas y condiciones generaron preocupaciones sobre la sostenibilidad y accesibilidad del motor para estudios pequeños e independientes. Unity Technologies tuvo que ajustar su estrategia y comunicarse activamente con los desarrolladores para abordar sus preocupaciones y mantener la confianza en la plataforma.

Unity ha evolucionado desde sus inicios como una herramienta para Mac OS a ser uno de los motores de desarrollo de videojuegos más populares y versátiles. A pesar de las recientes controversias, sigue siendo una opción preferida por millones de desarrolladores gracias a su capacidad para soportar múltiples plataformas y sus potentes herramientas de desarrollo.

(VANDAL & Díaz Herreros, 2023)

3 OBJETIVOS Y METODOLOGÍA

El presente apartado aborda los objetivos principales y específicos del proyecto, así como la metodología empleada para alcanzarlos. Se detalla el enfoque en la creación de una mecánica modular y escalable de reacciones mágicas que se complemente con las mecánicas generales de un videojuego FPS, ampliando tanto el diseño técnico como la experiencia del jugador. Este enfoque busca ofrecer una experiencia enriquecida, inmersiva y estratégica. Además, se describe como la metodología aplicada permitió iteraciones rápidas y mejoras continuas a lo largo del proceso de desarrollo.

3.1 Objetivo principal

El objetivo principal de este proyecto es desarrollar y perfeccionar una técnica de reacciones mágicas para un juego FPS, de manera que sea modular y escalable para un futuro desarrollo de un videojuego completo. Este sistema tiene como finalidad combinar las mecánicas tradicionales de disparo con las magias, añadiendo profundidad mecánica y variedad estratégica en el gameplay. En particular se busca que las reacciones mágicas sean dinámicas, que impactan significativamente en el jugador y se integren de forma fluida en la experiencia de combate.

3.2 Metas Específicas

A continuación, se desglosan los objetivos que permitieron la implementación del sistema y su integración en la demo.

3.2.1 Ampliar y optimizar el sistema de reacciones mágicas

Para enriquecer la experiencia de juego y garantizar la escalabilidad del sistema, se diseñó un modelo centralizado para la gestión de reacciones mágicas. Este enfoque centralizado permite añadir nuevas combinaciones de reacciones y estados con facilidad, gracias a la simplificación en la lógica y a la reducción de redundancias entre diferentes scripts. Las metas dentro de este objetivo incluyen:

- Crear un sistema que centraliza todas las reacciones mágicas en un único script. Este sistema gestiona tanto las combinaciones de elementos y estados en los enemigos como la ejecución de las reacciones. Esto facilita la implementación y ampliación de nuevas mecánicas con un impacto mínimo en el código base existente.

- Diseñar reacciones variadas que afecten a enemigos individuales o en área, proporcionando así diferentes formas de interactuar con los enemigos.
- Asegurar que cada reacción tenga un efecto visual claro que informe al jugador sobre su efecto.

3.2.2 Implementar un sistema de armas modular

El sistema de armas fue diseñado para ser ampliable y flexible, permitiendo la integración de nuevos tipos de armas en un futuro. Para lograr esto se plantearon los siguientes objetivos:

- Creación de un sistema que permite incluir armas con comportamientos únicos como pistolas, escopetas, etc.
- Diseño que facilita la adición de armas futuras mediante el uso de Scriptable Objects (Unity, n.d.) para configurar valores como daño, cadencia, capacidad de munición, entre otros.

3.2.3 Mejorar las mecánicas de FPS:

La experiencia del jugador se mejoró mediante la implementación de efectos visuales y mecánicas inmersivas. Las metas fueron:

- Garantizar una experiencia fluida en movimiento y combate, asegurando que las interacciones del jugador con el entorno y los enemigos sean satisfactorias y equilibradas.
- Implementar efectos de animación como *weapon sway*, *camera bob* y *recoil* para mejorar la inmersión y el *game feel* del jugador.

3.2.4 Facilitar la iteración y pruebas:

La iteración fue clave para ajustar la jugabilidad y los parámetros técnicos. Los puntos principales fueron:

- Implementar una consola de comandos funcional que permita depurar y probar de manera eficiente las mecánicas del juego.
- Implementar comandos en la consola para facilitar el reinicio del escenario de pruebas y enemigos.

3.3 Metodología

Para alcanzar estos objetivos, se emplearon estrategias basadas en un diseño modular, iteración continua y un enfoque centrado en el jugador. Estas estrategias aseguraron un desarrollo eficiente y orientado a la calidad.

3.3.1 Diseño modular y reutilización de código

- Se estructuraron los sistemas en scripts independientes para manejar aspectos clave como armas, magias y enemigos, asegurando que cada módulo fuese autónomo y fácil de escalar.

- Se emplearon Scriptable Objects de Unity para gestionar configuraciones de armas, permitiendo ajustes rápidos y evitando dependencias rígidas entre scripts.
- Esta estructura modular minimiza el impacto de futuros cambios en el código base, permitiendo iteraciones rápidas y eficientes.

3.3.2 Herramientas y tecnología

- Unity: El motor de juego elegido por su versatilidad, amplia documentación y soporte para herramientas avanzadas como Cinemachine.
- Cinemachine: Herramienta utilizada para gestionar el movimiento de la cámara del jugador, añadiendo efectos como *cámara bob* para mejorar la sensación de inmersión.
- Consola de comandos personalizada: Implementada para facilitar la depuración y prueba de mecánicas. Ejemplo de comandos:
    - reset_enemies: reinicia los enemigos en la escena.
    - debug_window: activa una ventana que muestra información técnica y estadísticas del sistema.

3.3.3 Iteración continua

Se realizaron pruebas constantes para ajustar parámetros del gameplay, incluyendo:

- Daño y alcance de las armas.
- Duración e intensidad de las reacciones mágicas.
- Utilidad de las reacciones en el gameplay de un videojuego real.

La consola de comandos fue clave para estas iteraciones, permitiendo realizar cambios en tiempo real y evaluar su impacto de manera inmediata.

3.3.4 Enfoque en la experiencia del jugador

El diseño y las decisiones técnicas estuvieron centrados en mejorar la interacción del jugador con el sistema. Ejemplos:

- Reacciones visualmente claras y con retroalimentación inmediata para el jugador.
- Implementación de animaciones y efectos visuales que refuercen la sensación de impacto en las mecánicas.

Se investigaron otros FPS populares para identificar buenas prácticas en diseño de armas, magias y reacciones. Títulos como *Doom Eternal* y *Destiny 2* sirvieron como referencia.

4 DESARROLLO

En este apartado se detalla el proceso de diseño, implementación y pruebas del sistema desarrollado, abarcando desde el rediseño conceptual hasta las mecánicas específicas de armas, magias, enemigos e interfaz.

4.1 Rediseño del Proyecto Original

El proyecto inicial enfrentó varios problemas que llevaron a un resultado final insatisfactorio. Estos problemas incluyen:

- **Alcance excesivo:** Se intentó implementar demasiados sistemas y mecánicas de manera simultánea.
- **Falta de priorización:** No se definieron claramente las prioridades entre los diferentes sistemas a desarrollar.

- **Diseño no modular:** Las mecánicas no estaban diseñadas para ser fácilmente ampliables o mantenibles.

En el nuevo proyecto, se redefinieron las prioridades y se diseñó una estructura modular y funcional. Los objetivos fueron claros:

- **Modularidad:** Sistemas como armas, magias y reacciones están diseñados para ser apilables y reutilizables.
- **Demo funcional:** La meta fue crear una demo que mostrará las mecánicas principales en un estado terminado y pulido.
- **Fluidez en el control del jugador:** Se trabajó en un controlador FPS que ofreciera una experiencia agradable y cómoda.
- **Herramienta de depuración:** Se diseñó un sistema de consola de comandos para facilitar pruebas rápidas y precisas.

4.1.1 Versión anterior

En la versión inicial de la mecánica, las magias se limitaban a tres elementos básicos: fuego, rayo y agua. Las reacciones eran simples y no diferenciaban entre objetivos individuales o múltiples. Este diseño tenía menos énfasis en modularidad y escalabilidad.

Tabla 1 - Magias, Estados y Reacciones versión anterior.

		Estados		
		En llamas	Con carga	Mojado
Magias	Fuego			Evaporación
	Rayo			Electrocución
	Agua	Evaporación	Electrocución	

Fuente: elaboración propia

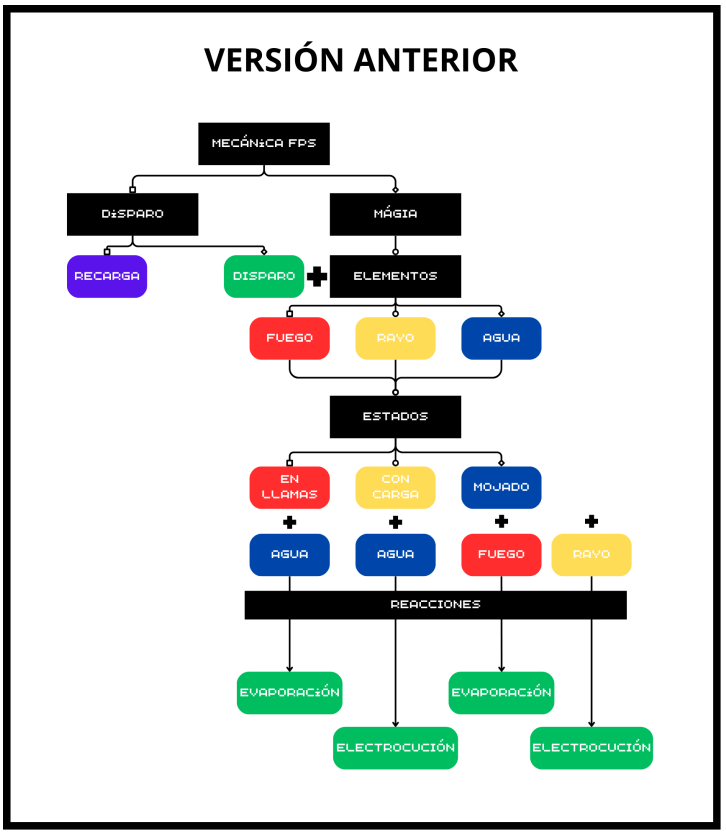


Ilustración 1 - Esquema del diseño anterior. **Fuente:** elaboración propia.

4.1.2 Versión mejorada

En la versión actual, las magias incluyen un cuarto elemento: hielo. Esto añade nuevas reacciones. Esta es una lista de las reacciones actuales:

- **Evaporación:** Esta reacción se consigue aplicando rayo en un enemigo en estado “en fuego” o viceversa.
- **Sobrecarga:** Esta reacción se consigue aplicando agua en un enemigo en estado “en fuego” o viceversa.
- **Implosión:** Esta reacción se consigue aplicando rayo en un enemigo en estado “en fuego” o viceversa.
- **Electrocución:** Esta reacción se consigue aplicando agua en un enemigo en estado “en rayo” o viceversa.
- **Fisura:** Esta reacción se consigue aplicando rayo en un enemigo en estado “en hielo” o viceversa.
- **Congelación:** Esta reacción se consigue aplicando hielo en un enemigo en estado “en agua” o viceversa.

Además, las reacciones se dividen en dos categorías:

- **Utilidad:** Evaporación (aplica ceguera temporal), Fisura (multiplicador para el siguiente daño entrante) y Congelación (ralentización temporal).
- **Daño:** Sobrecarga (explosión que aleja enemigos y aplica daño), Implosión (explosión que atrae enemigos y aplica daño) y Electrocución (daño en cadena a enemigos cercanos, el daño aumenta en función de los enemigos afectados).

Las reacciones de Utilidad sólo tienen efecto sobre el enemigo que recibe la combinación, mientras que las de daño aplican daños y efectos en área a múltiples enemigos.

Tabla 2 - Magias, Estados y Reacciones versión mejorada.

		Estados			
		En Fuego	En Rayo	En Agua	En Hielo
Magias	Fuego		Sobrecarga	Evaporación	Implosión
	Rayo	Sobrecarga		Electrocución	Fisura
	Agua	Evaporación	Electrocución		Congelación
	Hielo	Implosión	Fisura	Congelación	

Fuente: elaboración propia.

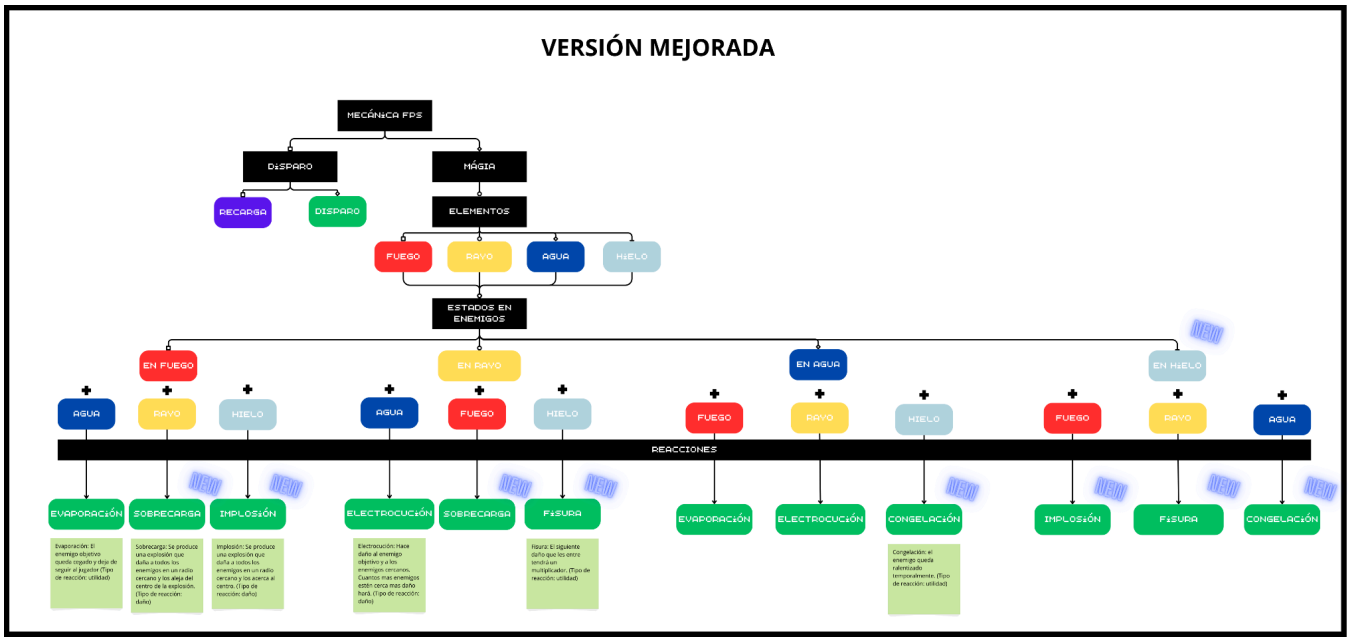


Ilustración 2 - Esquema del diseño mejorado. **Fuente:** elaboración propia.

4.2 Sistema dos ataques

Este sistema se implementa como la mecánica central que combina las armas tradicionales con las magias en un entorno estratégico. Este sistema permite al jugador abordar los enfrentamientos con enemigos resistentes a la magia mediante la interacción entre disparos simples y ataques mágicos. La mecánica está diseñada para fomentar la toma de decisiones en tiempo real, exigiendo al jugador que adapte sus acciones según el estado y la resistencia del enemigo.

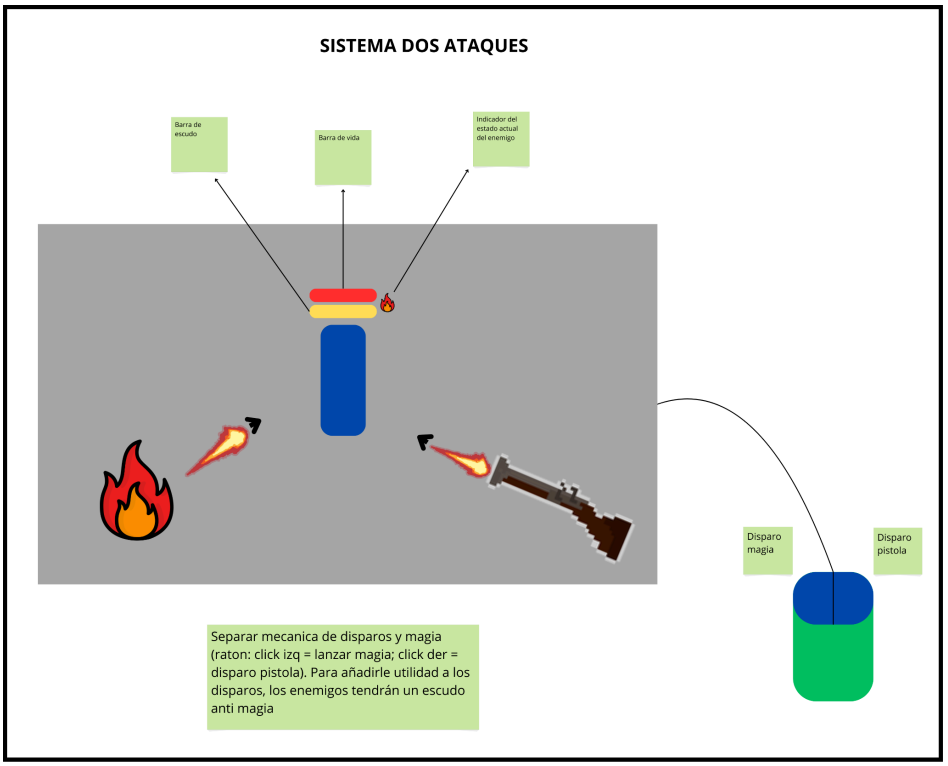
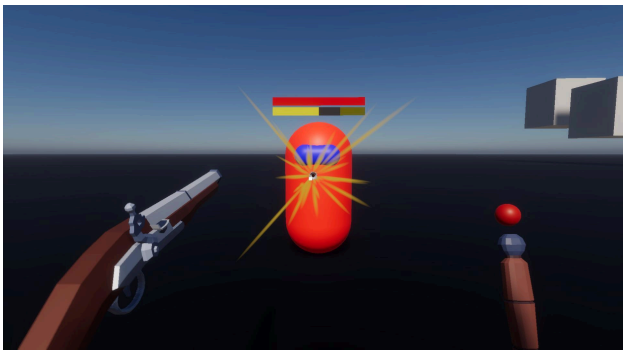


Ilustración 3 - Esquema del diseño del sistema de dos ataques. **Fuente:** elaboración propia.

4.2.1 Disparo simple

El diseño del disparo simple busca un balance entre ser eficaz para romper el escudo y no trivializar los combates. Por esta razón, el escudo no se regenera automáticamente.

- **Objetivo principal:** Destruir el escudo mágico para preparar al enemigo para ataques mágicos.
- **Objetivo secundario:** El disparo también hace daño a los enemigos, aunque este es muy bajo.
- **Consideraciones estratégicas:** El jugador debe evaluar si invertir tiempo y munición en destruir escudos rápidamente o manejar múltiples enemigos simultáneamente.



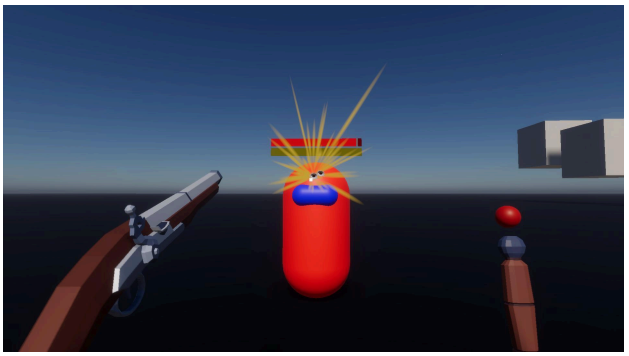


Ilustración 4 - Efecto disparo simple al escudo mágico. **Fuente:** elaboración propia.

Ilustración 5 - Efecto disparo simple al enemigo. **Fuente:** elaboración propia.

4.2.2 Disparo de Magia: Aplicación de Reacciones

Una vez que el enemigo ha perdido su escudo mágico, el disparo de magia entra en juego. Este ataque se divide en dos categorías principales: efectos de utilidad y efectos de daño, dependiendo del estado del enemigo y la magia utilizada. Las magias están diseñadas para ofrecer versatilidad y creatividad al jugador, permitiéndole adaptar su estrategia según el escenario.

- **Reacciones de utilidad:** Estas están diseñadas para modificar el comportamiento del enemigo. Son de un solo objetivo y afectan únicamente al enemigo al que se le aplica la magia, brindando al jugador opciones estratégicas para manejar la situación.
    - **Evaporación:** Ciega temporalmente al enemigo, permitiendo al jugador huir o reposicionarse.

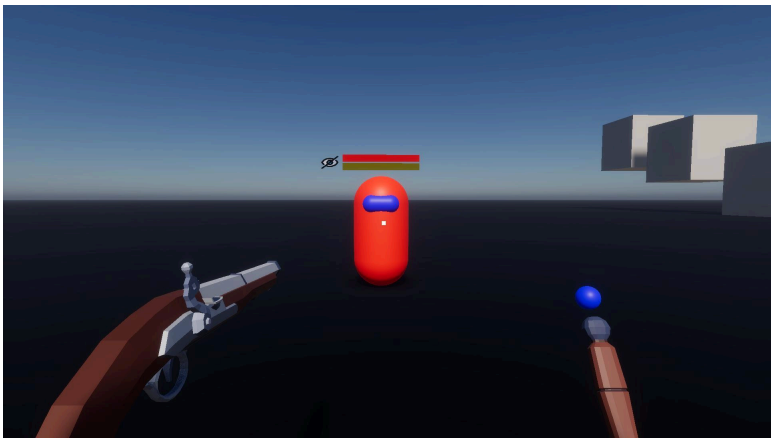


Ilustración 6 - Reacción evaporación. **Fuente:** elaboración propia.

- **Fisura:** Aplica un multiplicador al siguiente daño recibido por el enemigo, maximizando la efectividad del disparo o de magias de daño. También aumenta el daño del disparo simple.

Ilustración 7 - Reacción Fisura. **Fuente:** elaboración propia.

- **Congelación:** Ralentiza al enemigo, reduciendo su velocidad de movimiento y ataque, ideal para controlar a enemigos rápidos o agresivos.

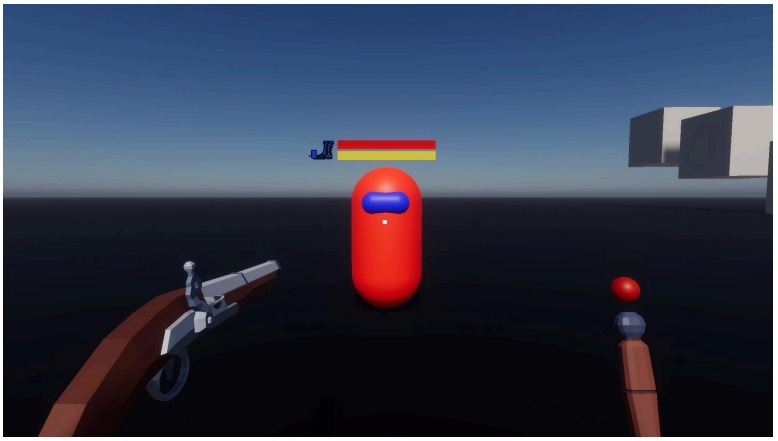


Ilustración 8 - Reacción congelación. **Fuente:** elaboración propia.

- **Reacciones de daño:** Estas tienen un impacto directo en el combate y afectan a múltiples enemigos en un área. Son ideales para eliminar grupos de enemigos o generar daño masivo en situaciones críticas.
  - **Sobrecarga:** Genera una explosión que daña y empuja a todos los enemigos cercanos, alejándose del enemigo al que se le aplica la reacción, su utilidad se centra en aislar a un solo objetivo

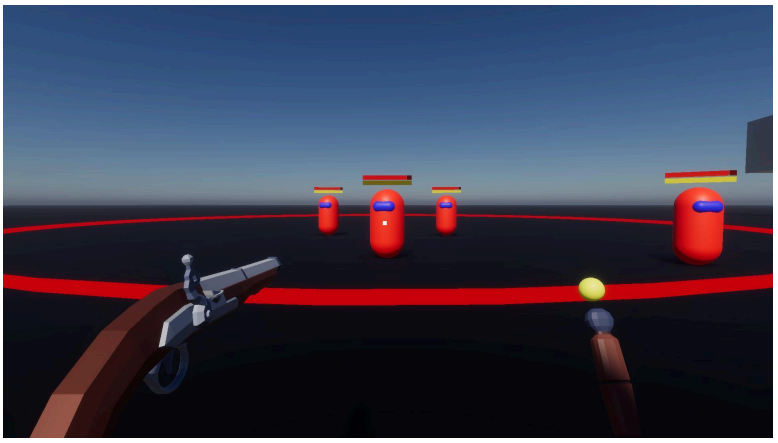


Ilustración 9 - Reacción sobrecarga. **Fuente:** elaboración propia.

- **Implosión:** Atrae a todos los enemigos cercanos hacia el enemigo al que se le ha aplicado la reacción, su utilidad se centra en juntar a muchos enemigos.

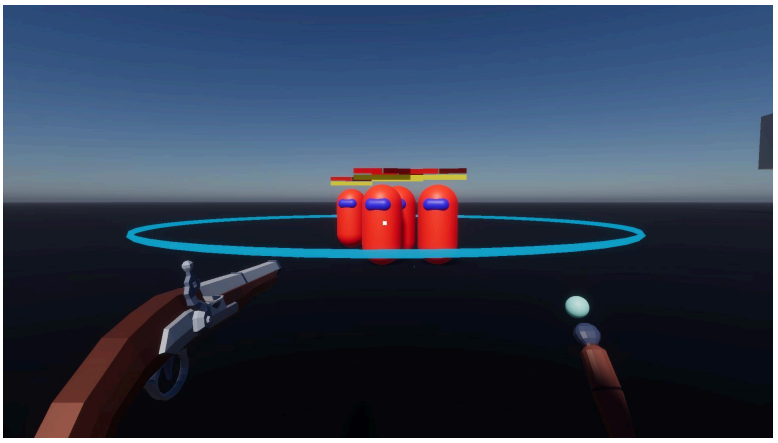


Ilustración 10 - Reacción implosión. **Fuente:** elaboración propia.

- **Electrocución:** Genera un daño en cadena que afecta al enemigo principal y a los cercanos, con mayor daño cuanto más enemigos estén en el área de efecto.

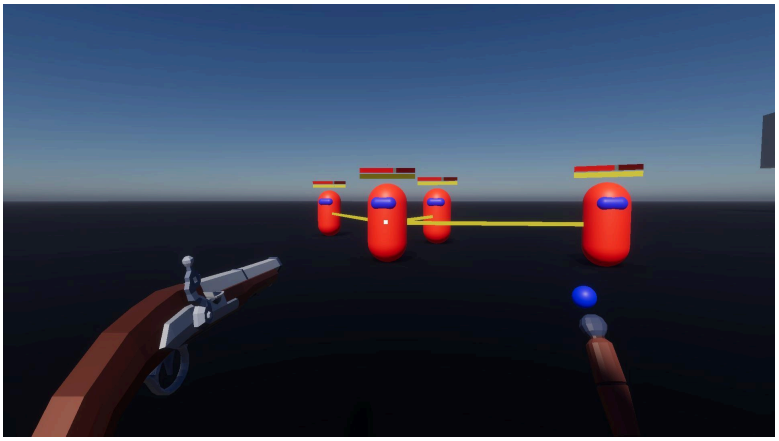


Ilustración 10 - Reacción implosión. **Fuente:** elaboración propia.

Estas reacciones sólo se pueden aplicar en enemigos sin escudo mágico, pero el escudo no bloquea el daño o los efectos de las reacciones en área. Facilitando así la eliminación de enemigos en grupos grandes.

4.3 Scripts

En esta sección se describen los diferentes sistemas principales del proyecto, estructurados de manera modular para facilitar su ampliación y mantenimiento. Cada sistema contiene componentes específicos diseñados para gestionar aspectos clave de la jugabilidad, como el movimiento del personaje, el uso de armas y magias, las reacciones dinámicas y la lógica de los enemigos.

4.3.1 Character Controller

Este script se centra en la mecánica principal del movimiento del jugador, proporcionando una experiencia fluida y precisa en primera persona. Para facilitar el desarrollo se hace uso del componente *CharacterController* (Unity, n.d.) de Unity, mediante este se aplica el movimiento y la lógica de detección de suelo. Este sistema incluye la lógica de control del jugador, interacciones con el entorno y efectos visuales como el *camera bob*.

El script utiliza *Cinemachine* para manejar la cámara, asegurando que los movimientos sean suaves y que la experiencia sea inmersiva. Este controlador sirve como base para las mecánicas de combate, ya que determina la posición y orientación del jugador.

Ilustración 11 - Estructura FP_Contoller. **Fuente:** elaboración propia.

4.3.2 Sistema de armas simples

Este sistema está diseñado para manejar las armas tradicionales, como pistolas o escopetas, asegurando una lógica modular que permite añadir nuevas armas fácilmente. Los scripts que componen este sistema se basan en una jerarquía bien definida, con *ScriptableObjects* para almacenar datos y scripts específicos para implementar las funcionalidades únicas de cada arma.

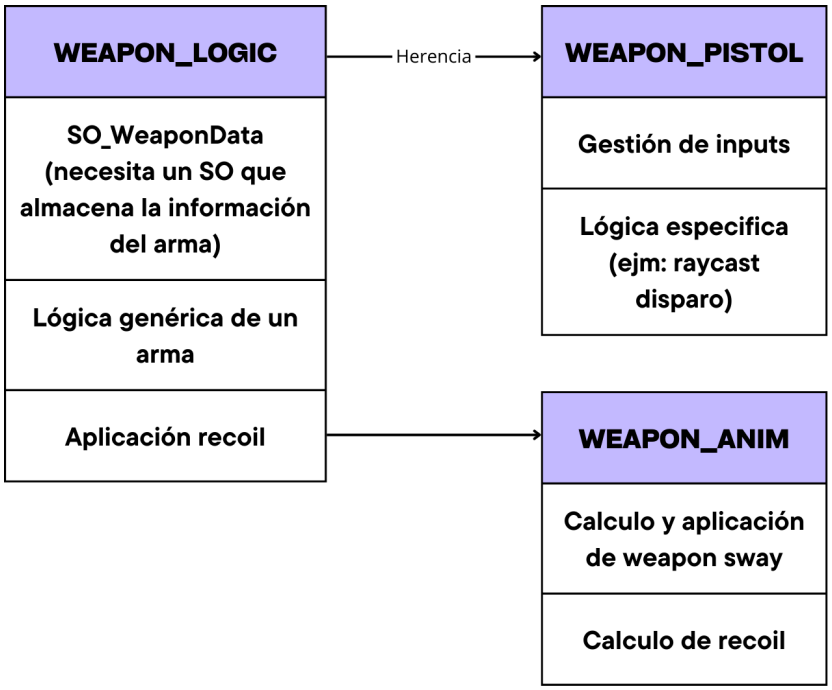


Ilustración 12 - Estructura sistema de armas simples. **Fuente:** elaboración propia.

4.3.2.1 SO_WeaponData

El *SO_WeaponData* es un *ScriptableObject* que almacena los datos esenciales de las armas, como el daño, el rango, la capacidad del cargador y la velocidad de recarga. Este enfoque permite gestionar diferentes configuraciones de armas de forma sencilla, sin necesidad de modificar el código base. Aunque actualmente solo se utiliza para una pistola, este diseño modular permite añadir fácilmente nuevas armas con diferentes parámetros.

4.3.2.2 Weapon_Logic

El script *Weapon_Logic* actúa como la base para todas las armas, centralizando funcionalidades generales como disparar, recargar y reproducir efectos de sonido. Este diseño facilita la creación de nuevas armas al permitir que los scripts específicos hereden esta lógica y añadan características únicas según sea necesario. Entre las funcionalidades principales se encuentran:

- **Disparo:** Gestiona la lógica de reducción de balas y asegura que el disparo solo ocurra cuando sea válido.
- **Recarga:** Aumenta las balas del cargador siempre que el arma lo permita.

- **Efectos de sonido:** Reproduce sonidos de disparo y recarga cuando las condiciones son correctas.

4.3.2.3 Weapon_Pistol

El script *Weapon_Pistol* hereda de *Weapon_Logic* y añade la lógica específica de la pistola. Su principal funcionalidad es gestionar el disparo mediante un *raycast*, lo que permite detectar impactos en el entorno o en enemigos. También maneja los efectos visuales asociados, como los fogonazos y trazos de bala. Este enfoque modular permite añadir nuevas armas con características específicas sin afectar la lógica general del sistema.

4.3.3 Sistema de magia, reacciones y enemigos

Este sistema está diseñado para integrar habilidades mágicas con un sistema de reacciones dinámicas el cual se comunica con la lógica de los enemigos. El sistema de magias permite alternar entre diferentes tipos, como fuego, rayo, agua y hielo, mientras que el *Magic_ReactionManager* gestiona cómo estas magias interactúan con los estados de los enemigos. La lógica de los enemigos se divide en componentes que controlan tanto los atributos básicos como su interacción con las magias.

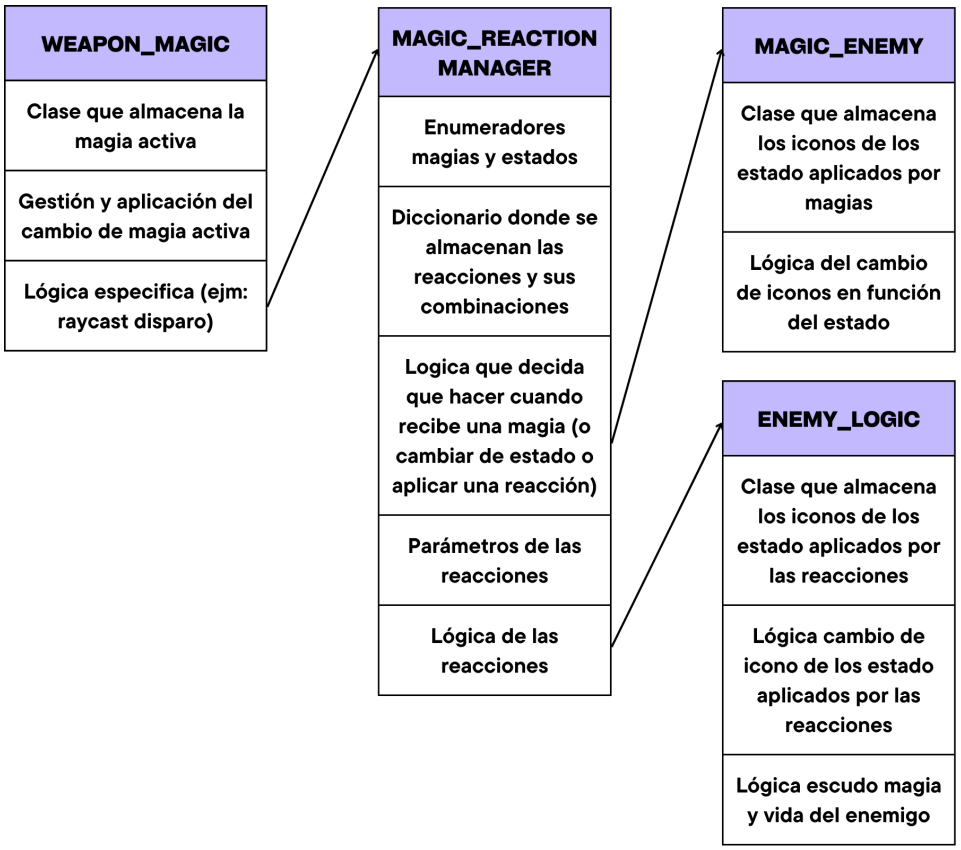


Ilustración 13 - Estructura sistema magias, reacciones y enemigos. **Fuente:** elaboración propia.

4.3.3.1 Weapon_Magic

El script *Weapon_Magic* gestiona el disparo de magias mediante un *raycast* y permite al jugador alternar entre diferentes tipos de magias. Cada magia tiene un efecto único que puede ser de utilidad (Evaporación, Fisura, Ralentización) o de daño (Sobrecarga, Implosión, Electrocución). Además, el script se comunica con el *Magic_ReactionManager* para aplicar los efectos correspondientes a los enemigos.

4.3.3.2 Magic_ReactionManager

El **Magic_ReactionManager** es el núcleo central del sistema de reacciones mágicas del proyecto. Este script es responsable de gestionar todas las interacciones entre las magias lanzadas por el jugador y los estados de los enemigos, ofreciendo un sistema centralizado, modular y escalable para implementar reacciones dinámicas. Su diseño permite añadir o modificar reacciones sin afectar otras partes del código, asegurando una flexibilidad y mantenibilidad excepcionales.

- **Diseño del sistema**
    - **Centralización de la lógica**: Todas las combinaciones de magias y estados se gestionan desde un único lugar utilizando un diccionario. Esto elimina la necesidad de lógica distribuida y asegura que todas las interacciones se definan en un único componente.
    - **Centralización de la lógica**: Todas las combinaciones de magias y estados se gestionan desde un único lugar utilizando un diccionario. Esto elimina la necesidad de lógica distribuida y asegura que todas las interacciones se definan en un único componente.
    - **Aplicación de reacciones**: Cuando un enemigo recibe un impacto de una magia, el *Magic_ReactionManager* consulta el diccionario. Si existe una entrada para la combinación de magia y estado actual del enemigo, se ejecuta la función correspondiente, que aplica el efecto deseado.
    - **Visualización de efectos**: El script también maneja la visualización de las reacciones mediante el uso de *LineRenderer* para rayos y animaciones para efectos de área como explosiones o implosiones. Esto asegura que las reacciones sean claras y tengan un impacto visual significativo en la jugabilidad.
- **Reacciones implementadas**
    - **Sobrecarga (Overload)**: Esta reacción ocurre cuando un enemigo en estado de “en fuego” recibe un rayo. Genera una explosión que empuja a los enemigos cercanos y daña al enemigo central y a los cercanos dentro de un radio de efecto. El empuje se implementa utilizando *AddExplosionForce* para simular una fuerza que combina un desplazamiento hacia atrás y un empuje vertical.
        - Efectos visuales: Un círculo que crece desde el centro del enemigo afectado simula la expansión de la explosión.

- **Implosión (Implosion)**: Se activa cuando un enemigo en estado de hielo recibe fuego. Esta reacción atrae a los enemigos cercanos hacia el centro del enemigo afectado, infligiendo daño. Se simula utilizando *AddForce* con fuerzas negativas para atraer en lugar de empujar.
    - Efectos visuales: Un círculo que disminuye de tamaño hacia el centro del enemigo.
- **Electrocución (Electrocution)**: Aplica daño en cadena entre el enemigo central y los enemigos cercanos. El daño aumenta en función del número de enemigos afectados.
    - Efectos visuales: Los rayos se mueven dinámicamente desde el enemigo central a los cercanos.
- **Evaporación (Evaporation)**: Cuando un enemigo en estado de fuego recibe agua, se aplica un efecto de ceguera temporal. Esto afecta únicamente al enemigo central
    - Efectos visuales: Un icono que indica la modificación.
- **Fisura (Fissure)**: Cuando un enemigo en estado de “en rayo” recibe hielo se aplica el efecto, este hace que el siguiente daño recibido por el enemigo sea significativamente mayor.
    - Efectos visuales: Un icono que indica la modificación.
- **Congelación (Freezing)**: Reduce temporalmente la velocidad de movimiento de los enemigos afectados, ralentizando sus acciones. Esto ocurre cuando un enemigo en estado de agua recibe hielo.
    - Efectos visuales: Un icono que indica la modificación.

- **Ventajas del diseño**
    - **Modularidad**: Gracias al uso del diccionario, las nuevas combinaciones de magias y reacciones pueden añadirse fácilmente sin modificar el código base. Esto asegura que el sistema sea escalable.
    - **Desacoplamiento**: Las reacciones están completamente separadas de la lógica del enemigo y del jugador, lo que permite que los sistemas trabajen de forma independiente y coordinada.

- **Clara separación de responsabilidades**:
  - El script *Magic_ReactionManager* se encarga únicamente de gestionar las interacciones entre magias y enemigos.
  - La visualización y aplicación de efectos son responsabilidades separadas, facilitando su depuración y mantenimiento.

En resumen, el script *Magic_ReactionManager* es una pieza clave del sistema, centralizando la lógica de las reacciones mágicas y garantizando que estas sean claras, impactantes y fáciles de expandir en futuros desarrollos.

4.3.3.3 Magic_Enemy

El script *Magic_Enemy* gestiona los estados mágicos de los enemigos y su interacción con las magias del jugador. Cada estado tiene un icono asociado que se actualiza dinámicamente según el efecto aplicado. Los estados disponibles son: En Fuego, En Rayo, En Agua, En Hielo. Este script también coordina la eliminación de efectos cuando un enemigo es atacado con un disparo simple.

4.3.3.4 Enemy_Logic

El script *Enemy_Logic* maneja los aspectos fundamentales de los enemigos, como la vida, el escudo mágico y los cambios de estado. Además, se encarga de la visualización de estos atributos mediante barras de vida y escudo. Este script utiliza la función *ChangeReactionState* para gestionar la aplicación de efectos de manera eficiente, permitiendo un sistema flexible para añadir nuevos estados en el futuro.

4.4 Interfaz y Experiencia del Jugador

El diseño de la interfaz y la experiencia del jugador se centró en ofrecer una jugabilidad clara, fluida, asegurando que las acciones del jugador recibieron retroalimentación inmediata y efectiva. Este apartado aborda tanto la interfaz gráfica como los efectos visuales y de movimiento que refuerzan la inmersión del jugador.

4.4.1 Interfaz gráfica (UI)

La interfaz gráfica desempeña un papel crucial en la comunicación entre el sistema del juego y el jugador. Se diseñó para proporcionar información relevante de manera clara y accesible, optimizando la experiencia de juego y minimizando la necesidad de explicaciones externas. Entre los elementos destacados de la interfaz se incluyen:

- **UI de los enemigos:**
    - **Barra de vida:** Representa la vida restante del enemigo, disminuye visualmente a medida que el enemigo recibe daño.
    - **Barra de escudo mágico:** Indica si el enemigo tiene un escudo mágico activo. Este elemento es fundamental para que el jugador sepa si puede aplicar magias o necesita primero destruir el escudo con un arma tradicional.
    - **Indicadores de estados:** Iconos que muestran el estado actual del enemigo (en fuego, en rayo, en hielo o en agua). Esto permite al jugador adaptar su estrategia en tiempo real según el estado del enemigo.
    - **Indicadores de modificadores de reacciones:** Señalan los efectos activos de las reacciones, como la ralentización por congelación o la ampliación del daño entrante.
- **Retroalimentación visual:**
    - Aunque en este prototipo no se han llegado a incluir efectos visuales complejos se plantea añadir en el entorno y sobre los enemigos para reforzar la percepción de las reacciones mágicas. Por ejemplo, partículas de hielo rodeando a un enemigo bajo el efecto de congelación o un destello de electricidad en electrocución.

- **Información del jugador:**
    - Aunque en este prototipo no se profundiza en la interfaz del jugador, se plantea incluir indicadores de munición, habilidades mágicas activas y cooldowns en futuros desarrollos.

4.4.2 Experiencia del jugador

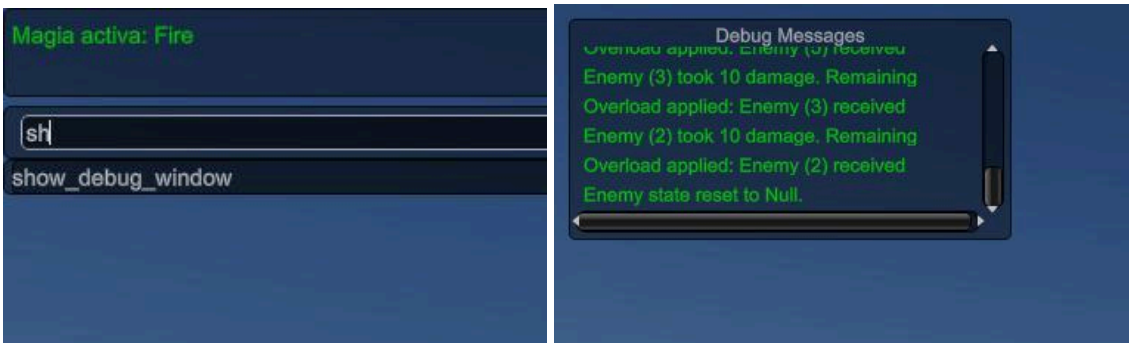
La experiencia del jugador fue diseñada teniendo en cuenta el "game feel", es decir, las sensaciones táctiles y visuales que refuerzan la inmersión y satisfacción al interactuar con el juego. Se implementaron varias técnicas y sistemas para mejorar esta experiencia:

- **Oscilación del arma y retroceso (Weapon Sway y Recoil)**:
    - Mediante código se gestiona el movimiento oscilante del arma del jugador, simulando el movimiento natural de un arma al caminar. Este efecto, conocido como *weapon sway*, varía según la velocidad del movimiento, proporcionando una sensación más realista y dinámica.
    - El retroceso del arma o *recoil* se aplica al disparar, desplazando el arma hacia atrás y hacia arriba durante un instante, acompañado de un movimiento de retorno suave. Esto refuerza la sensación de potencia de las armas, especialmente en combates frenéticos.
- **Oscilación de la cámara (Camera Bobbing)**:
    - La cámara del jugador cuenta con un efecto de oscilación vertical al caminar, simulado mediante un *CinemachineBasicMultiChannelPerlin*. Este efecto sutil añade realismo al movimiento del personaje y mejora la inmersión.
    - La frecuencia y amplitud de la oscilación están calibradas para que no resulten intrusivas, ajustándose al ritmo del movimiento del jugador.
- **Sistema de vibración y retroalimentación en el disparo**:
    - Para reforzar la acción de disparar, se implementó un sistema de vibración de la cámara mediante *CinemachineImpulseSource* y *CinemachineImpulseListener*. Cada disparo genera un impulso que simula un pequeño retroceso de la cámara, creando una sensación de impacto y dinamismo.

- **Efectos de reacciones mágicas**:
    - Las reacciones mágicas, como la sobrecarga o la electrocución, incluyen efectos visuales dinámicos que refuerzan el impacto en el jugador.
- **Feedback de impacto**:
    - Al disparar a los enemigos, se muestran efectos de partículas y animaciones específicas según el arma y la reacción aplicada. Esto asegura que el jugador reciba retroalimentación clara sobre sus acciones y el estado de los enemigos.
    -

4.5 Consola de Comando

La consola de comandos se diseñó como una herramienta de soporte para facilitar las pruebas, la depuración y el desarrollo iterativo del prototipo. Si bien no es un componente central de la mecánica del juego, su implementación ha demostrado ser clave para optimizar el flujo de trabajo y garantizar ajustes dinámicos en tiempo real.



4.5.1 Propósito y funcionalidad

La consola permite ejecutar comandos específicos para tareas comunes en el desarrollo y pruebas. Sus principales funcionalidades incluyen:

- **Depuración en tiempo real**: Comando *show_debug_window*, que activa una ventana para visualizar eventos y mensajes del sistema mientras el juego está en ejecución. Esto resulta especialmente útil en builds, donde las herramientas internas de depuración son limitadas.
- **Iteración eficiente**: Comando *reset_enemies*, que reinicia todos los enemigos de la escena, incluyendo sus posiciones, estados y atributos. Este comando agiliza las pruebas repetitivas de las reacciones mágicas y comportamientos enemigos.

4.5.2 Beneficios para el desarrollo

La consola mejora significativamente el flujo de trabajo al permitir:

- **Pruebas rápidas y dinámicas**: Reduciendo tiempos de reinicio y ajustes manuales.
- **Flexibilidad y extensibilidad**: Su estructura permite la adición de nuevos comandos conforme evolucionan las necesidades del proyecto.
- **Análisis continuo**: Facilita la identificación y resolución de problemas al mantener un registro claro de las interacciones y eventos.

4.6 Controles del Juego

El diseño de los controles es fundamental para garantizar una experiencia de juego fluida e intuitiva. En este apartado, se explican los controles básicos asignados al jugador, con un enfoque en cómo se relacionan con las mecánicas de disparo y magia, así como las interacciones generales en el entorno.

- **Controles Básicos**
    - **Movimiento del jugador**:
        - **Teclas W, A, S, D**: Movimiento hacia adelante, atrás, izquierda y derecha.
        - **Espacio**: Saltar.
        - **Shift Izquierdo:** Correr
    - **Control de la cámara**:
        - **Ratón**: Control de la vista en primera persona.
- **Controles de combate**
    - **Disparo Simple**:
        - **Botón izquierdo del ratón**: Disparo tradicional con el arma equipada. Útil para eliminar escudos mágicos enemigos.
        - **Recarga**:
            - **R**: Recargar el arma equipada.
- **Disparo de Magia**:
    - **Botón derecho del ratón**: Lanza la magia activa, aplicando efectos de utilidad o daño dependiendo del estado del enemigo.

        - **Rueda del ratón**: Alternar entre los diferentes tipos de magia disponibles (Fuego, Rayo, Agua, Hielo).

- **Herramientas**
    - **Consola de comandos**:
        - **Tecla º (izquierda del 1)** : Abrir o cerrar la consola de comandos.
        - **Tecla TAB o BLOQ MAYÚS:** Autocompletar el texto
        - **Ejemplo de comandos**:
            - help: Te dice todos los comandos posibles
            - reset_enemies: Reinicia los enemigos en la escena.
            - show_debug_window: Muestra mensajes de depuración en tiempo real.

5 CONCLUSIÓN

El desarrollo de este proyecto se ha centrado en crear un sistema modular y escalable que integre armas, mecánicas mágicas y reacciones en un marco cohesivo y extensible. A través de un diseño basado en modularidad y gestión centralizada, se ha logrado un equilibrio entre flexibilidad y eficiencia, cumpliendo con los objetivos iniciales del proyecto.

Aunque la consola de comandos tuvo un papel secundario, su impacto fue relevante para optimizar los procesos de prueba y depuración. Esta herramienta permite realizar ajustes en tiempo real, reiniciar escenarios y validar las mecánicas de juego de forma inmediata, mejorando significativamente el flujo de trabajo sin ser el foco principal del proyecto.

La organización en esquemas de los sistemas clave de gestión de armas, gestión de magias e implementación de reacciones han demostrado ser esenciales para mantener un código limpio, organizado y ampliable. El uso de *ScriptableObjects* para las armas y la lógica centralizada del *Magic_ReactionManager* para las reacciones permiten futuras expansiones y modificaciones sin complicaciones, estableciendo una base técnica sólida.

La implementación de reacciones dinámicas, como Sobrecarga, Electrocución e Implosión, ha enriquecido la experiencia de juego al ofrecer diversas opciones tácticas a los jugadores. Estas mecánicas, divididas en tipos de

utilidad y daño, fomentan la creatividad y el enfoque estratégico, añadiendo profundidad al sistema. El diseño modular y escalable garantiza su adaptabilidad para futuros desarrollos e iteraciones.

En comparación con el intento previo, este proyecto demuestra un enfoque más estructurado y refinado. Al priorizar la creación de un prototipo funcional sobre un producto completo, se ha mantenido el control durante el desarrollo, entregando resultados tangibles y estableciendo una base sólida para escalar hacia un juego completo.

En conclusión, la aplicación de principios de diseño modular, respaldada por estrategias de prueba y depuración bien definidas, demuestra el potencial de este sistema para crear una experiencia de juego dinámica y estratégica. Este enfoque establece las bases para evolucionar hacia un juego completo, destacando la importancia de la iteración y la organización clara de sistemas complejos.

6 BIBLIOGRAFÍA

Facultat d'Informàtica de Barcelona (Universitat Politècnica de Catalunya) & L'Illa Diagonal. (2008). *Història dels videojocs*. https://www.fib.upc.edu/retroinformatica/historia/videojocs.html

JOT DOWN & Cuevas, D. (n.d.). *OK, boomer shooter*. https://www.jotdown.es/2023/01/ok-boomer-shooter/

PCMag & Jensen, K. T. (2022, septiembre 14). *Knee Deep in the Dead: The History of First-Person Shooters*. https://www.pcmag.com/news/the-complete-history-of-first-person-shooters

Unity. (n.d.). *ScriptableObject*. Unity - Manual. https://docs.unity3d.com/Manual/class-ScriptableObject.html

Microsoft. (n.d.). *Herencia en C# y .NET*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/tutorials/inheritance

VANDAL & Díaz Herreros, R. (2023, septiembre 19). *La historia de Unity, el motor gráfico más utilizado por desarrolladores indies*. https://vandal.elespanol.com/noticia/1350765440/la-historia-de-unity-el-motor-grafico-mas-utilizado-por-desarrolladores-indies/

Unity. (n.d.). *Unity - Documentation*. Character Controller. https://docs.unity3d.com/ScriptReference/CharacterController.html

7 ANEXOS

7.1 One Pager de un juego que implementa la mecánica

7.2 Enlace del proyecto subido en Itch.io

Build de la demo en itch.io

7.3 Grado de relación con los ODS

Los Objetivos de Desarrollo Sostenible (ODS) son un conjunto de 17 metas globales adoptadas en 2015 por los Estados Miembros de las Naciones Unidas como parte de la Agenda 2030. Estos objetivos buscan abordar los mayores desafíos a los que se enfrenta la humanidad, como la pobreza, la desigualdad, el cambio climático, la paz y la justicia. Los ODS tienen un enfoque integral y equilibrado que promueve el desarrollo sostenible en sus dimensiones económica, social y ambiental.

En el contexto del diseño y desarrollo de videojuegos, es importante reflexionar sobre cómo estos pueden contribuir a la promoción de los ODS, ya sea fomentando valores como la inclusión, el acceso a la educación, el uso responsable de la tecnología, o inspirando cambios positivos en la sociedad.

Este proyecto, orientado al diseño y desarrollo de la mecánica principal de un juego de género FPS, tiene como objetivo documentar el proceso para servir como guía para futuros desarrollos. A continuación, se describen los ODS y se evalúa su posible relación con el proyecto.

- **ODS 1 Fin de la pobreza:** Erradicar la pobreza en todas sus formas y dimensiones. Este objetivo busca garantizar que todas las personas, especialmente las más vulnerables, tengan acceso a recursos básicos como alimentos, vivienda, educación y empleo.
- **ODS 2 Hambre cero:** Eliminar el hambre y garantizar el acceso a alimentos nutritivos para todas las personas, con especial énfasis en la sostenibilidad de los sistemas agrícolas.
- **ODS 3 Salud y bienestar:** Garantizar una vida sana y promover el bienestar para todos en todas las edades, incluyendo el acceso a la atención médica y la promoción de estilos de vida saludables.
- **ODS 4 Educación de calidad:** Garantizar que todas las personas tengan acceso a una educación inclusiva, equitativa y de calidad, promoviendo oportunidades de aprendizaje permanente.
- **ODS 5 Igualdad de género:** Lograr la igualdad entre los géneros y empoderar a mujeres y niñas, eliminando las barreras que perpetúan la discriminación y la desigualdad.
- **ODS 6 Agua limpia y saneamiento:** Garantizar la disponibilidad y la gestión sostenible del agua y el saneamiento para todos.

- **ODS 7 Energía asequible y no contaminante:** Asegurar el acceso a energía asequible, confiable, sostenible y moderna para todos.
- **ODS 8 Trabajo decente y crecimiento económico:** Promover el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo, y el trabajo decente para todos.
- **ODS 9 Industria, innovación e infraestructura:** Desarrollar infraestructuras resilientes, promover la industrialización inclusiva y sostenible, y fomentar la innovación.
- **ODS 10 Reducir la desigualdad en y entre los países:** Reducir la desigualdad dentro de los países y entre ellos, promoviendo la inclusión económica, social y política de todas las personas.
- **ODS 11 Ciudades y comunidades sostenibles:** Lograr que las ciudades sean inclusivas, seguras, resilientes y sostenibles, fomentando la planificación urbana y el transporte público.
- **ODS 12 Producción y consumo responsable:** Garantizar modalidades de consumo y producción sostenibles, promoviendo la eficiencia en el uso de recursos y reduciendo el desperdicio.
- **ODS 13 Acción por el clima:** Adoptar medidas urgentes para combatir el cambio climático y sus impactos, promoviendo la resiliencia y la adaptación.
- **ODS 14 Vida submarina:** Conservar y utilizar de manera sostenible los océanos, los mares y los recursos marinos.
- **ODS 15 Vida de ecosistemas terrestres:** Gestionar sosteniblemente los bosques, combatir la desertificación, detener e invertir la degradación de las tierras y detener la pérdida de biodiversidad.
- **ODS 16 Paz, justicia e instituciones sólidas:** Promover sociedades pacíficas e inclusivas, garantizar el acceso a la justicia para todos y construir instituciones eficaces y responsables.
- **ODS 17 Alianza para lograr los objetivos:** Fortalecer los medios de implementación y revitalizar la alianza mundial para el desarrollo sostenible.

El proyecto desarrollado en este Trabajo de Fin de Grado tiene un enfoque técnico y creativo que, aunque no está directamente vinculado a un ODS específico, puede contribuir indirectamente a varios objetivos:

- **ODS 4: Educación de calidad:**
  La memoria documenta el proceso de diseño y desarrollo como una guía práctica para futuros desarrolladores. Esto promueve el aprendizaje y la formación técnica en diseño y programación de videojuegos, alineándose con la meta de acceso a la educación de calidad.
- **ODS 9: Industria, innovación e infraestructura:**
  Al centrarse en la modularidad y escalabilidad de las mecánicas, el proyecto fomenta la innovación en el desarrollo de videojuegos. La implementación de un sistema centralizado y flexible sirve como base para proyectos futuros más ambiciosos, apoyando la industrialización inclusiva y sostenible en el sector del entretenimiento digital.
- **ODS 12: Producción y consumo responsables:**
  Aunque este objetivo está más enfocado en el ámbito ambiental, la metodología de diseño modular empleada en este proyecto fomenta la reutilización de recursos (código y mecánicas), minimizando el esfuerzo duplicado y promoviendo una programación eficiente.

Este proyecto, aunque limitado en su alcance, sienta las bases para explorar cómo los videojuegos pueden ser herramientas valiosas para promover principios de sostenibilidad, educación e innovación, alineándose con la filosofía de los Objetivos de Desarrollo Sostenible.