UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Refactorización y ampliación de "Clara Yema": Mejora de un juego desarrollado con C# y Unity

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Navarro García, Alejandro

Tutor/a: Abad Cerdá, Francisco José

CURSO ACADÉMICO: 2023/2024

Resumen

Este trabajo de fin de grado se centra en la refactorización, mantenimiento y ampliación de un juego llamado "Clara Yema" desarrollado en una asignatura previa. El principal objetivo es estudiar la mantenibilidad del código actual y aplicar las refactorizaciones necesarias para mejorar su funcionalidad. Además de mejorar la estructura del código, se añadirá mayor funcionalidad al juego, así como mejoras a la funcionalidad actual y cambios para arreglar mecánicas que no funcionan correctamente a la hora de ejecutar el programa. El problema principal del código actual es el entrelazado de las clases, el cual causa muchos problemas en la ejecución del juego, ya que hay muchos elementos que son dependientes entre sí. Esto crea una estructura de código frágil y propensa a errores, donde cualquier modificación o adición puede tener repercusiones imprevistas en otras partes del programa. Para abordar esta situación, se planifica desacoplar las clases, reduciendo así la interdependencia y aumentando la modularidad del código. Se emplearán patrones de diseño para mejorar la cohesión entre las clases y facilitar la gestión de cambios en el futuro.

Palabras clave: refactorización, Unity, juego, C#.

Abstract

This undergraduate thesis focuses on the refactoring, maintenance, and expansion of a game called "Clara Yema," which was developed in a previous course. The main objective is to study the maintainability of the current code and apply the necessary refactoring to improve its functionality. In addition to enhancing the code structure, the game will be given additional functionality, as well as improvements to existing functionality and changes to fix mechanics that do not work correctly when running the program. The main problem with the current code is the entanglement of the classes, which causes many issues during the game's execution, as there are many interdependent elements. This creates a fragile and error-prone code structure, where any modification or addition can have unforeseen repercussions in other parts of the program. To address this situation, the plan is to decouple the classes, thereby reducing interdependence and increasing the code's modularity. Design patterns will be employed to improve cohesion between the classes and facilitate change management in the future.

Keywords: refactoring, Unity, game, C#.

Índice de contenidos

Índice de tablas

Índice de figuras

1. Introducción

El proyecto desarrollado en este Trabajo de Fin de Grado es el mantenimiento y ampliación de un videojuego. Aunque la aparición de los videojuegos sea relativamente nueva, lleva existiendo más de siete décadas. Comenzó alrededor del año 1950 y ha ido evolucionando de manera exponencial desde entonces. El desarrollo de videojuegos está en constante crecimiento, así como la base de usuarios que consumen los productos.

Los videojuegos requieren el mismo o más mantenimiento que cualquier software para asegurar su correcta funcionalidad y rendimiento. La refactorización del software es necesaria para optimizar el código, corregir errores, mejorar la eficiencia y facilitar implementaciones de características futuras (Fowler, 2018). El mantenimiento también incluye la adaptación a nuevas tecnologías, así como la incorporación de *feedback* de los usuarios para mejorar la experiencia de juego, la detección y corrección de errores.

El videojuego objeto de este trabajo, titulado Clara Yema, pertenece a los géneros de aventuras y plataformas. Estos géneros se componen de una historia en la cual los usuarios participan mediante la superación de obstáculos y puzles (Adams, 2013), además de interacciones con personajes no jugables, comúnmente conocidos por sus siglas en inglés como *NPCs (Non Playable Characters)*. La combinación de narrativa y jugabilidad desafiante requiere un código robusto y bien estructurado para garantizar que los jugadores puedan disfrutar de una experiencia fluida y sin interrupciones. Por tanto, este proyecto se enfoca no solo en mantener la integridad técnica del juego, sino también en mejorar sus elementos narrativos y de interacción para ofrecer una mejor experiencia de usuario.

1.1. Motivación

La motivación del porqué se ha elegido el mantenimiento y desarrollo de este videojuego es el interés en el sector del desarrollo del videojuego, así como el atractivo de desarrollar un producto jugable y disfrutable para contribuir al sector. El desarrollo de un videojuego permite aplicar los conocimientos adquiridos a lo largo del grado de una manera diferente, ya que es necesario aplicar conocimientos de distintas ramas de la Informática.

El desarrollo de videojuegos siempre me ha parecido muy interesante, ya que permite al desarrollador crear un producto muchísimo más interactivo que un software convencional. Además, permite recibir una retroalimentación del usuario más subjetiva, ya que la experiencia entre distintos usuarios puede ser muy diferente debido a los distintos gustos y preferencias. Otra de las motivaciones que me llevó a

seleccionar este proyecto es la posibilidad de poder dedicarme profesionalmente a este sector, que ha sido la inspiración de mis elecciones educativas.

1.2. Objetivos

El objetivo del Trabajo de Fin de Grado es refactorizar y ampliar el videojuego Clara Yema, desarrollado en la asignatura Desarrollo de Videojuegos 3D del Grado de Ingeniería Informática, junto con alumnos de la Facultad de Bellas Artes.

Como podemos observar en la Figura 1, algunos de los objetivos más específicos son:

1. Optimización del código y rendimiento: revisar y refactorizar el código existente para mejorar su eficiencia y reducir la cantidad de errores de ejecución existentes.
2. Mejora de la jugabilidad: introducir nuevas habilidades especiales de los personajes, perfeccionar las existentes para mejorar la jugabilidad y el desafío, permitiendo hacer un juego con más interacción y con unos puzles más desafiantes.
3. Pruebas: implementar un sistema de control de calidad del software, desarrollando e implementando pruebas unitarias y con usuarios.
4. Desarrollo de MVP: realizar al menos dos Minimum Viable Products (MVPs) durante el proceso de desarrollo, permitiendo iteraciones rápidas y la incorporación de retroalimentación con las pruebas de usuario.

Figura 1 Diagrama de los objetivos específicos

1.3. Impacto

Se espera que el juego encuentre un sitio en las plataformas de publicación de videojuegos y se aspira a ampliar el catálogo de juegos cooperativos del género de plataformas y aventuras con pantalla dividida.

Además, se pretende trabajar en el desarrollo cognitivo y motriz de los usuarios, así como en la toma de decisiones o la resolución de problemas.

Al tratarse de un videojuego multijugador, también permite desarrollar habilidades sociales como la comunicación efectiva y la colaboración para el trabajo en equipo. Los jugadores interactúan y trabajan juntos para lograr objetivos comunes, lo que permite desarrollar habilidades interpersonales como la resolución de conflictos, la empatía y la capacidad de llegar a consensos.

Por último, se debe tener en cuenta el impacto cultural de los videojuegos, pues sirven como medio para difundir culturas, tradiciones y valores. Internet permite compartir nuestra cultura con otros países a través de los videojuegos, contribuyendo consecuentemente al proceso de globalización.

Como proyecto preocupado por las generaciones futuras, en este trabajo se tienen en cuenta los ODS (OBJETIVOS DE DESARROLLO SOSTENIBLE). Concretamente, el objetivo 9, construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación, es uno de los pilares básicos de nuestro desarrollo, pues se apoya en la innovación y el progreso tecnológico. (Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación, s.f.)

1.4. Metodología

Se ha utilizado la metodología ágil para el desarrollo de este proyecto (McCarthy, 2020). La metodología ágil es una forma de desarrollar proyectos que necesitan flexibilidad y rapidez. Esta metodología nació de la necesidad de entregar proyectos en un plazo de entrega más reducido, ya que la forma tradicional de trabajo retrasaba la entrega (del Dedo, García, & Gómez, 2018).

Existen diferentes tipos de metodología ágil (Chandler, 2020). A continuación, se definirán algunos tipos:

- Cascada: metodología con un enfoque más tradicional, en la que se define cada fase claramente antes de empezar el proyecto y el paso entre fases depende de la finalización de la fase anterior. La ventaja principal de esta metodología es que es fácil de entender, la desventaja es que no se adapta a imprevistos.
- Scrum: metodología más popular en el desarrollo de proyectos software. Los proyectos se dividen en *sprints,* es decir, periodos cortos

de tiempo con objetivos concretos. Se trata de una metodología flexible con una buena capacidad adaptativa. (Keith, 2010)

- Kanban: metodología ágil que se enfoca en la visualización del flujo de trabajo y en la mejora continua. Utiliza un tablero con columnas para representar el estado de las tareas y establece límites de trabajo en progreso para optimizar la eficiencia. Es flexible y se adapta bien a cambios, pero puede carecer de la estructura de otras metodologías más formales.
- Programación extrema: metodología que tiene en cuenta los comentarios de los clientes y pretende modificar continuamente para garantizar la calidad del software.
- Lean: metodología basada en producción JIT (*Just In Time*). Consiste en una entrega continúa reduciendo la inversión, el tiempo y el esfuerzo. (Jones & Womack, 2012)

En este proyecto, se ha optado por la metodología Kanban, ya que se puede utilizar tanto en proyectos grupales como individuales. Esta metodología se considera una de las más apropiadas para el desarrollo de videojuegos debido a su flexibilidad y su alta capacidad de autoorganización. Esto es crucial para tener la capacidad de pivotar si alguna parte del producto no cumple con las expectativas. La flexibilidad de Kanban permite hacer ajustes rápidos y eficaces en el desarrollo, fundamental en la creación de videojuegos, donde la calidad y la experiencia del usuario son prioridades.

Además, se han empleado herramientas como GitHub y Trello para mejorar la organización del videojuego. GitHub (Figura 3: https://github.com) facilita la gestión del código fuente, permitiendo un seguimiento detallado de los cambios. Trello (Figura 2: https://trello.com), por otro lado, es una herramienta de gestión de proyectos estilo Kanban que permite organizar y priorizar las tareas de manera visual. Con estas herramientas, se puede tener una visión clara del estado del proyecto en todo momento, lo que contribuye a la identificación rápida de posibles problemas o áreas de mejora.

Figura 2 Trello®

Figura 3 GitHub®

1.5. Estructura

A lo largo de la memoria, se hablará sobre diferentes puntos que se han tenido en cuenta al hacer el proyecto. En primer lugar, se comentará el estado del arte, donde se hará una recapitulación sobre la historia de los videojuegos, los títulos más relevantes en cuanto a videojuegos de plataformas cooperativos además de una

crítica a los mismos. También en este mismo punto se verá en que motor gráfico podemos desarrollar un videojuego y cuál es la propuesta de este proyecto.

A continuación, se analizará el por qué se realiza este proyecto, definiendo así los requisitos tanto funcionales como no funcionales, realizando un modelado conceptual y detectando los problemas previos a la refactorización. Una vez detectados los problemas, se analizarán las posibles soluciones, se escogerá una solución y se definirá un plan de trabajo a seguir.

Otro de los puntos más relevantes de este trabajo es el diseño de la solución escogida. Para ello es necesario definir la arquitectura del sistema, detallar el diseño y emplear la tecnología correcta. Todo ello se concretará en este punto.

A partir del diseño de la solución, será necesario desarrollar la solución. Se explicará cómo se ha desarrollado el nivel, las mecánicas básicas de los personajes, los elementos interactivos, el sistema de diálogos, el jefe final y la interfaz visual.

A posteriori, se explicará cómo se ha realizado la implementación del proyecto, así como su exportación y como se han realizado las pruebas de rendimiento. Además, ya que este proyecto se centra en parte en el mantenimiento, se realizarán pruebas tanto previas como posteriores a la refactorización. Se calcularán las nuevas métricas y relaciones del proyecto.

Por último, pero no menos importante, se concluirá y se detallará una proyección futura sobre el desarrollo de este videojuego. También se añadirá en el anexo el GDD del videojuego y los objetivos de desarrollo sostenible relacionados con el proyecto.

1.6. Colaboraciones

El reparto de tareas original de este videojuego es el siguiente:

- Alejandro Navarro
  - Programación del jefe final
  - Sistema de rotación de cámaras
  - Programación de menú de opciones
  - Programación general del nivel
  - Integración de elementos
  - Diseño de nivel
- Adrián Soriano
  - Movimientos y habilidades de los personajes
  - Programación de la vista a través de otros objetos
  - Programación general del nivel
  - Integración de elementos
  - Diseño de nivel
- Marta Colomer
  - Sistema de diálogos

    - Implementación de la interfaz y del texto de los diálogos
    - Aplicación de sonido
    - Programación general del nivel
    - Integración de elementos
    - Diseño de nivel
- Ana Paula Moreno
    - Concept art, modelado, rig, y animaciones de Patata, Tomate, Batidora
    - Modelado de Calabacín y el mortero
    - Modelado, rig, y animaciones de Yema
    - Integración interfaz de usuario
    - Integración de elementos
    - Diseño de nivel
- Clara García
    - Concept art de Remi y Calabacín
    - Modelado, rig y animaciones de Clara, Salchicha, Zanahoria y jefe final.
    - Rig y animaciones de Calabacín
    - Diseño interfaz menú principal
    - Storyboard animáticas
    - Elección elementos sonoros
    - Integración de elementos
    - Diseño de nivel
- Celia Veiga
    - Concept art de Clara, Yema, escenario y props
    - Modelado de Escenario
    - Animáticas
    - Elección elementos sonoros
    - Integración de elementos
    - Diseño de nivel

A lo largo de este Trabajo de Fin de Grado, se han realizado las siguientes tareas:

- Reestructuración de la estructura de clases del videojuego.
- Refactorización y rediseño de las mecánicas del jefe final.
- Sistema de cámaras para la pelea final.
- Diseño e integración de elementos en el nivel.
- Refactorización del movimiento de los personajes.
- Desarrollo de nuevas habilidades de los personajes.
- Desarrollo de nuevos puzles repartidos a lo largo del mapa.
- Nuevo sistema de diálogos más ligero y optimizado.
- Recreado el esqueleto de los personajes para adaptarlos a las nuevas habilidades y movimientos.

1.7. Convenciones

- El nombre de las clases se mostrará en negrita.
- Los términos en inglés estarán en cursiva.
- El código fuente estará en Courier New tamaño 9.
- El texto de las tablas estará en Aptos Narrow tamaño 9.
- Los botones utilizados en el juego en comillas.

2. Estado del arte

En este capítulo se describirá el contexto del videojuego. Se analizarán los títulos existentes dentro del género y los que implementen mecánicas similares al juego para determinar los elementos distintivos de Clara Yema. Se investigará el origen de las mecánicas que conforman el juego, se ofrecerá posibles alternativas y se razonará las decisiones tomadas para las mecánicas seleccionadas (Historia de los videojuegos, s.f.).

2.1. Historia de los videojuegos

El primer videojuego nació en 1952, llamado OXO (Velasco, 2011), una versión del tres en raya ejecutada en la EDSAC creado por Alexander S. Douglas. En 1958, William Higginbotham desarrolló un juego que simulaba un partido de tenis llamado Tennis For Two (Trilnick, s.f.), que fue el primer videojuego para dos jugadores. Más tarde, en 1962, Steve Russell del MIT creó Spacewar! (Radice, s.f.), un juego de combate espacial.

El desarrollo de los videojuegos domésticos comenzó en 1966 con Fox and Hounds (Villar, 2016) de Ralph Baer, que posteriormente evolucionó hasta convertirse en Magnavox Odyssey en 1972, el primer sistema doméstico de videojuegos que se conectaba a la televisión.

El inicio de los videojuegos como industria se consolidó con el lanzamiento de Computer Space en 1971 por Nolan Bushnell, una versión comercial de Spacewar!. Sin embargo, el verdadero impulso llegó en 1972 con Pong (Ruiz, 2023), que fue el primer gran éxito comercial que estableció a Atari como un pionero en la industria. Durante esta década, los avances tecnológicos significativos, como los microprocesadores y los chips de memoria, permitieron el desarrollo de videojuegos más complejos y accesibles. Juegos como Space Invaders (Fernández, 2020) y Asteroids lograron un enorme éxito en los salones recreativos.

Los años 80 marcaron un crecimiento explosivo en la popularidad de los videojuegos, tanto en salones recreativos como en sistemas domésticos. Consolas como la Odyssey 2, la Intellivision y la Atari 5200 lideraron el mercado, mientras que juegos icónicos como Pac-Man (Perazo, 2024) y Pole Position (Linares, 2019) se convirtieron en fenómenos culturales. Sin embargo, en 1983, la industria sufrió una crisis significativa que afectó principalmente a Estados Unidos y Canadá, resultando en un declive temporal en la popularidad de los videojuegos.

En 1985, Japón emergió como un líder global en la industria de los videojuegos con el lanzamiento de la Nintendo Entertainment System (NES), que revitalizó el mercado. Ese mismo año, la aparición de Super Mario Bros marcó un punto de inflexión, ya que introdujo la idea de objetivos y finales en los juegos, ampliando considerablemente la experiencia de juego. Japón también dominó el mercado de los videojuegos portátiles, consolidando su liderazgo con el lanzamiento de la Game Boy en 1989.

A principios de los años 90, el mundo de los videojuegos experimentó un salto tecnológico con la aparición de consolas de 16 bits como la Mega Drive y la Super Nintendo Entertainment System (SNES). Durante esta década, la tecnología de gráficos tridimensionales comenzó a revolucionar el diseño de videojuegos. Títulos como DOOM (Carabaña, 2020) y Alone in the Dark (Gallego, 2008) en PC fueron pioneros en la utilización de gráficos 3D, ofreciendo a los jugadores entornos más inmersivos y dinámicos. A su vez, en la misma década, surgieron las consolas como la PlayStation y la Sega Saturn, de 32 bits, y la Nintendo 64 y la Atari jaguar, de 64 bits.

En los años 2000 y 2001, Sony lanzó la PlayStation 2 y Microsoft entró en la industria de las consolas con la Xbox, ampliando el mercado y la competencia. Nintendo, por su parte, presentó la GameCube y la Game Boy Advance, consolidando su posición como líder en el mercado de consolas. Sega, incapaz de competir con las nuevas consolas de Sony y Microsoft, cesó la producción de hardware en 2002 y se enfocó exclusivamente en el desarrollo de software. Mientras tanto, los ordenadores personales se establecieron como la plataforma más flexible para videojuegos, permitiendo actualizaciones continuas y personalizaciones a través de la adición de nuevos componentes y accesorios. La industria de los videojuegos se expandió y diversificó significativamente, estableciendo bases sólidas para su futuro.

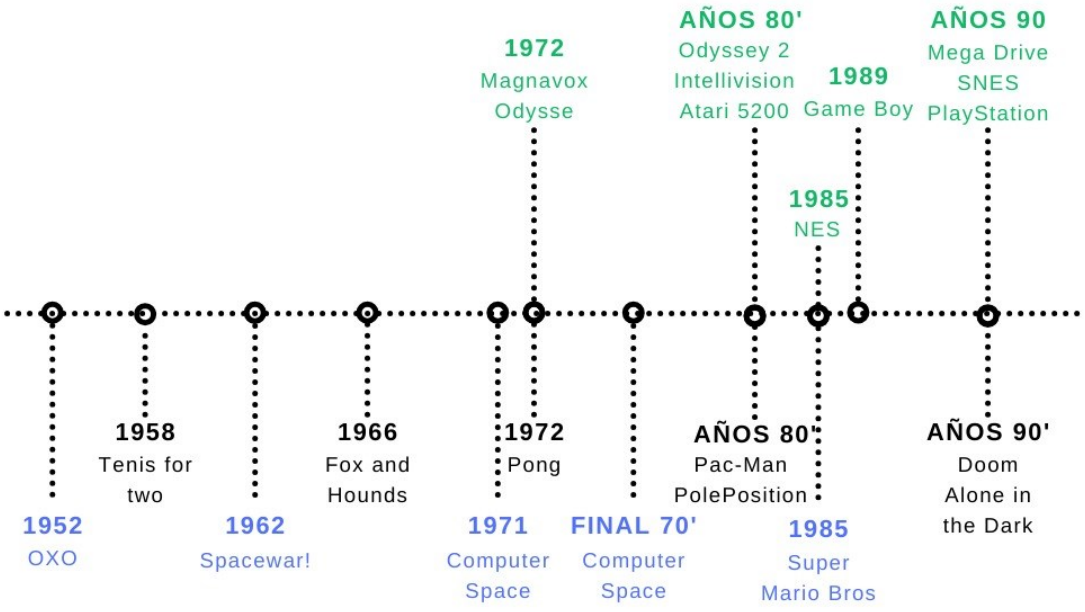


Figura 4 Línea temporal de la historia de los videojuegos

2.2. Análisis de títulos existentes

Esta sección describe las diferencias y similitudes entre nuestro videojuego y otros videojuegos del mercado similares, así como una justificación de las decisiones tomadas sobre las mecánicas desarrolladas.

2.2.1. Human: Fall Flat

Lanzado en julio de 2016, es un videojuego de puzles y plataformas en tercera persona con modo de juego cooperativo opcional. Este juego ha sido desarrollado por No Brakes Games y distribuido por Curve Digital.

Es un juego basado en la física en el que los jugadores controlan a personajes. Estos personajes, cuyas características se pueden personalizar, deben navegar por una serie de niveles abiertos, resolviendo puzles y superando obstáculos para avanzar.

El juego simula el movimiento y la interacción de los personajes con el entorno de manera realista y, a menudo, humorística. Los jugadores tienen control total sobre los brazos y la dirección de los personajes, lo que permite interaccionar con objetos y el entorno de múltiples maneras.

Cada nivel presenta un entorno único con desafíos y puzles distintos, desde castillos medievales hasta plantas industriales y paisajes surrealistas. Además, el juego soporta hasta ocho jugadores en línea, fomentando la cooperación y el caos, mientras los jugadores intentan resolver los puzles juntos.

Figura 5 Captura del juego Human Fall Flat

2.2.2. Portal 2

Lanzado en abril de 2011, es un videojuego de puzles y plataformas en primera persona con modo cooperativo opcional. Este juego ha sido desarrollado y publicado por Valve.

Los jugadores utilizan una pistola de portales para crear portales interconectados en superficies especiales, lo que les permite resolver una variedad de puzles ambientales y de física.

Incluye una campaña cooperativa separada, en la que dos jugadores controlan a robots llamados Atlas y P-Body, trabajando juntos para resolver puzles.

Figura 6 Captura del juego Portal 2

2.2.3. A Way Out

Lanzado en marzo de 2018, es un videojuego de acción y aventuras cooperativo en tercera persona. Este juego también ha sido desarrollado por Hazelight Studios y distribuido por Electronic Arts.

Es un juego diseñado exclusivamente para el modo cooperativo, en el que dos jugadores deben trabajar juntos para avanzar en la historia utilizando habilidades únicas y tomando decisiones conjuntas que afectan el desarrollo de la historia.

La historia se centra en la relación entre Leo y Vincent, dos prisioneros con antecedentes diferentes pero un objetivo común: escapar de la prisión y resolver asuntos personales pendientes en el exterior.

Posee una trama que se adapta según las decisiones tomadas por los jugadores. Incluye una variedad de mecánicas de juego, como sigilo, combate, conducción de vehículos, y resolución de puzles.

Además, el juego utiliza una pantalla dividida que cambia de tamaño y forma según la acción, lo que permite a los jugadores ver y coordinar sus acciones simultáneamente.

Figura 7 Captura del juego A Way Out

2.2.4. Unravel Two

Lanzado en junio de 2018, es un videojuego de plataformas y puzles con modo cooperativo opcional. Este juego ha sido desarrollado por Coldwood Interactive y distribuido por Electronic Arts.

Sigue la historia de Yarny, un pequeño ser hecho de hilo. En esta entrega, Yarny encuentra a otro ser similar y los dos deben trabajar juntos para superar los desafíos del juego.

En el modo cooperativo, los jugadores deben trabajar juntos estrechamente. Las habilidades de balanceo, trepado y creación de puentes con el hilo son esenciales para resolver los puzles. Muchos de los puzles están diseñados para ser resueltos por dos jugadores, requiriendo coordinación para manipular objetos y usar el entorno de manera efectiva.

Figura 8 Captura del juego Unravel Two

2.2.5. It Takes Two

Lanzado en marzo de 2021, es un videojuego de aventuras y plataformas cooperativo en tercera persona, que no permite la opción de jugar con un solo jugador.

El juego fue desarrollado por Hazelight Studios y publicado por Electronic Arts. Está disponible para PC, PlayStation, Xbox y Nintendo Switch. Se ha traducido a diferentes idiomas como italiano, francés, español y japonés, entre otros. Aparte de un modo historia, encontramos disponibles diferentes minijuegos.

Está diseñado exclusivamente para el juego cooperativo, lo que significa que dos jugadores deben trabajar juntos para progresar en la historia. Los jugadores asumen el rol de Cody y May, una pareja que está en proceso de divorcio y que se ve transformada en muñecos por un hechizo. Deben colaborar para superar desafíos y obstáculos en su camino para revertir el hechizo.

Los jugadores deben colaborar y utilizar habilidades únicas de cada personaje para resolver puzles y avanzar en el juego. Combina diferentes géneros de juego, incluyendo plataformas, rompecabezas, disparos, y más.

Figura 9 Captura del juego It Takes Two

2.3. Critica al estado del arte

Con esta información, hemos detectado las siguientes características para los videojuegos de plataformas en 3D, cooperativos y casuales.

- Videojuego de plataformas: el juego está compuesto por un mapa con plataformas a las que se accede mediante saltos.
- Perspectiva en tercera persona: cada jugador controla una cámara y permite visualizar al personaje desde atrás.
- Pantalla dividida dinámica: se muestra una pantalla dividida que se adapta a la posición de los jugadores en cada momento de la partida.

Figura 10 Ejemplo de pantalla dividida dinámica en A Way Out

- Diálogos/historia: el juego tiene un sistema de diálogos que ayuda a desarrollar una historia y pone en situación a los jugadores.

- Puzles: el juego presenta puzles que los jugadores deberán resolver ayudándose entre ellos.
- Gráficos 3D: utilización de gráficos 3D de alta resolución.

Nombre	Plataformas	Tercera persona	Pantalla dividida dinámica	Historia	Puzles	Gráficos 3D
Clara Yema	✓	✓	✓	✓	✓	✓
It Takes Two	✓	✓	✗	✓	✓	✓
A Way Out	✗	✓	✓	✓	✓	✓
Human: Fall Flat	✓	✓	✗	✗	✓	✓
Portal 2	✓	✗	✗	✓	✓	✓
Unravel Two	✓	✓	✗	✓	✓	✗

Tabla 1 Comparativa de las soluciones existentes

Como se muestra en la tabla, los juegos de aventuras y plataformas de este tipo suelen usar una perspectiva en tercera persona. Este estilo es fundamental porque permite a los jugadores ver más el entorno, lo que facilita la navegación y la interacción en el juego. En juegos cooperativos, esta vista es especialmente útil porque ayuda a los jugadores a coordinarse mejor al poder ver no solo su personaje, sino también a los compañeros, además del escenario completo. Esta amplia perspectiva favorece la colaboración y la toma de decisiones conjuntas, lo que es esencial para resolver los desafíos que plantea el videojuego.

También, muchos de estos juegos incluyen puzles que fomentan el trabajo en equipo y el pensamiento crítico. Resolver estos desafíos requiere que los jugadores se comuniquen y colaboren, lo que hace que la experiencia de juego sea más enriquecedora y gratificante. Los puzles no solo añaden un nivel de complejidad que mejora la jugabilidad, sino que también promueven habilidades importantes como la lógica y la cooperación.

Otra característica común en estos juegos es la presencia de una historia y diálogos que enriquecen la experiencia de juego. La inclusión de una narrativa bien desarrollada y personajes con los que los jugadores pueden interactuar aporta contexto y profundidad, ayudándoles a los jugadores a sumergirse en el mundo del juego. Los diálogos y la trama proporcionan pistas y motivaciones adicionales que hacen que el juego sea más atractivo y envolvente.

Aunque no todos los juegos analizados utilizan la pantalla dividida dinámica, su presencia en algunos títulos destaca su importancia en la experiencia cooperativa. Esta función permite que los jugadores se muevan de manera independiente mientras comparten la misma pantalla, lo que mejora la flexibilidad y la estrategia durante el juego. La pantalla dividida dinámica permite a los jugadores explorar diferentes áreas al mismo tiempo o realizar tareas específicas, lo que facilita la cooperación.

Así, concluimos que el videojuego Clara Yema pretende cumplir con las características básicas de los demás juegos del género, como historia, plataformas, gráficos, puzles, etc.

2.4. Análisis de motores existentes

2.4.1. Unity

Unity es uno de los motores de videojuegos más populares y versátiles en la industria actual. Se lanzó en 2005 y destaca por su capacidad de crear juegos en 2D y 3D. Unity ofrece una amplia gama de herramientas que permiten a los desarrolladores crear desde juegos simples hasta experiencias complejas y detalladas. Su interfaz amigable y su motor gráfico lo hacen una opción ideal tanto para principiantes como para desarrolladores experimentados.

Unity permite exportar juegos a múltiples plataformas, incluyendo consolas, PC, dispositivos móviles y realidad virtual. Además, cuenta con una amplia comunidad de desarrolladores y una tienda de recursos propia (Unity Asset Store), lo que facilita el acceso a herramientas adicionales y activos de alta calidad.

Figura 11 Logo de Unity

2.4.2. Unreal Engine

Unreal Engine es un motor de videojuegos desarrollado por Epic Games, conocido por sus gráficos de alta calidad y su robusto conjunto de herramientas. Lanzado por primera vez en 1998, este motor destaca por su capacidad para renderizar gráficos fotorrealistas y soportar efectos visuales avanzados, lo que lo convierte en la opción preferida para juegos AAA.

Unreal Engine incluye una herramienta de scripting visual llamada Blueprints, que permite a los desarrolladores crear y prototipar sin necesidad de escribir código. Su soporte para experiencias de realidad virtual y aumentada también es robusto, ofreciendo una plataforma completa para desarrollos en estas áreas.

Unreal Engine permite la exportación a múltiples plataformas y es especialmente potente en términos de renderización gráfica y simulación de física avanzada. No obstante, puede ser más complicado de aprender y dominar en comparación con otros motores.

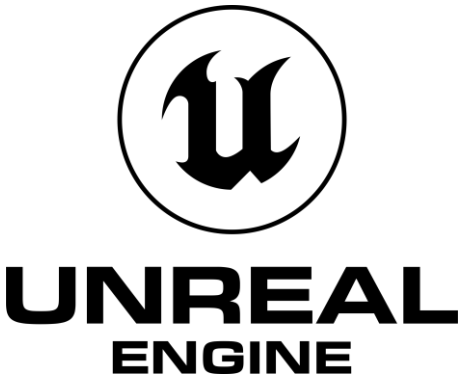


Figura 12 Logo de Unreal Engine

2.4.3. Godot Engine

Godot Engine es un motor de código abierto que ha ganado popularidad por su flexibilidad y por contar con una comunidad activa. Se lanzó oficialmente en 2014 y se especializa en desarrollar juegos 2D y 3D. Su estructura modular y su enfoque en la simplicidad y la eficiencia lo convierten en una opción atractiva para desarrolladores de todos los niveles de experiencia.

Godot es completamente gratuito y de código abierto, lo que permite a los desarrolladores modificar y personalizar el motor según sus necesidades. Además, soporta la exportación de juegos a múltiples plataformas sin costo adicional. Godot utiliza GDScript, un lenguaje de scripting similar a Python, que facilita la programación y el prototipado rápido.

Aunque Godot es menos potente para gráficos 3D comparado con motores como Unreal, es una opción excelente para proyectos que se centran en el desarrollo 2D y en mecánicas de juego innovadoras.

Figura 13 Logo de Godot

2.4.4. GameMaker

GameMaker es un motor de juegos que permite a los desarrolladores crear juegos 2D de manera rápida y eficiente. Se lanzó por primera vez en 1999 y es conocido por su facilidad de uso y por permitir la creación de prototipos y juegos completos en poco tiempo.

Este motor es ideal para principiantes debido a su interfaz intuitiva y a sus herramientas de desarrollo simplificadas. GameMaker soporta exportar juegos a múltiples plataformas y es perfecto para el desarrollo rápido de juegos 2D. Sin embargo, no es adecuado para el desarrollo de juegos 3D complejos debido a sus limitaciones en esta área.

Es una excelente opción para proyectos pequeños y medianos que se centran en juegos de plataformas, puzles y otros géneros que no requieren gráficos tridimensionales avanzados.

Figura 14 Logo de GameMaker

2.4.5. Comparativa de Motores de Videojuegos

Motor	Gráficos	Facilidad de Uso	Flexibilidad	Multiplataforma
Unity	Alto	Alta	Alta	Alta
Unreal Engine	Muy Alto	Media	Alta	Alta
Godot Engine	Medio	Alta	Alta	Alta
GameMaker	Bajo/Medio	Muy Alta	Media	Alta

Tabla 2 Comparativa de los motores de videojuegos existentes

2.5. Propuesta

Este trabajo aporta mejoras significativas al juego existente mediante optimizaciones de código y rendimiento, así como una mejor organización del proyecto. Mediante el uso de pruebas, detalladas en apartados posteriores, se comprueba su correcto funcionamiento, asegurando una óptima calidad del juego.

Asimismo, se han implementado mejoras en la jugabilidad, que hacen el juego más entretenido, divertido y ameno. Mediante la integración de diversos elementos, como diálogos o nuevos puzles, se mejora la interacción de los personajes con el entorno, fomentando una mayor cooperación entre los jugadores.

Además de ofrecer entretenimiento, el juego también pretende contribuir al desarrollo cognitivo y social de los jugadores. Este nuevo juego sigue un patrón común con los otros videojuegos de plataformas y aventuras como se ha descrito en el apartado 2.3. Si bien no revoluciona el género, ofrece una propuesta original e inédita dentro del mundo de los videojuegos.

3. Análisis del problema

Este Trabajo de Fin de Grado se ha centrado en dos áreas fundamentales en el desarrollo de videojuegos:

- **Mantenimiento de un videojuego:** Se ha llevado a cabo una identificación y corrección de errores, así como una optimización del rendimiento. Se han implementado estrategias de mantenimiento preventivo, perfectivo y correctivo, garantizando así la sostenibilidad y la mejora continua del producto a lo largo de su ciclo de vida.
- **Desarrollo de nuevas características:** El trabajo también se ha enfocado en el diseño e integración de nuevas funcionalidades. Se ha buscado mejorar la jugabilidad, perfeccionando mecánicas y dándoles utilidad, haciendo el videojuego más entretenido.

3.1. Especificación de requisitos

Al comienzo del desarrollo de un proyecto software se realiza la especificación de requisitos, que se dividen principalmente en requisitos funcionales y requisitos no funcionales.

Los requisitos funcionales son aquellos que definen el comportamiento del sistema y las funcionalidades que debe proporcionar. Estos requisitos describen lo que el sistema debe hacer para cumplir con las expectativas y necesidades de los usuarios. Incluyen descripciones de las interacciones entre el sistema y sus usuarios, así como los procesos internos y las operaciones que el sistema debe realizar.

Los requisitos no funcionales son aquellos que describen las características de calidad, restricciones y condiciones que deben cumplir el sistema o el entorno en el que se desarrollará el proyecto de software. Estos requisitos no se enfocan en las funciones o comportamientos específicos del sistema, sino en cómo debe ser el sistema para ser efectivo y cumplir con las expectativas del usuario y del entorno en el que operará.

A continuación, se enumerarán los diferentes requisitos tanto funcionales como no funcionales que se aplicarán a este proyecto software:

3.1.1. Requisitos funcionales

Los siguientes requisitos del juego, ordenados de mayor a menor importancia dentro de cada categoría, se definen a continuación:

- **Juego**
  - J1: La interfaz se controla tanto con mandos como con ratón.
  - J2: Una vez dentro de la zona de combate contra el jefe final, no se podrá volver a la zona inicial.
  - J3: Las pistas de audio del juego serán reproducidas al volumen que el usuario haya seleccionado y guardado.
- **Personaje**
  - P1: Cuando un usuario presione cualquier botón de movimiento asignado a una acción del personaje, este reaccionará con un movimiento acorde a la orden de entrada.
  - P2: Cuando un usuario presione el botón de salto, el personaje reaccionará con un salto.
  - P3: Cuando un personaje sea atacado, golpeado o entre en una zona de muerte, se le aplicará una penalización de 3 segundos sin poder hacer ninguna acción.
- **Personaje: Clara**
  - C1: Al pulsar el botón de interacción cerca de un objeto interactivo, se ejecutará la acción que dicho objeto tenga asignada.
  - C2: Al pulsar el botón de habilidad cerca de una superficie adhesiva, Clara se pegará a la superficie, inmovilizándola hasta que se deje de adherir presionando el mismo botón.
- **Personaje: Yema**
  - Y1: Al pulsar el botón de habilidad en el aire, Yema usará su habilidad especial, golpeando contra el suelo y accionando cualquier botón interactivo que haya cerca.
- **Jefe final: Sartén**
  - S1: Debe estar siempre sobre uno de los tres fogones de inducción.
  - S2: Cuando el fogón en el que este posicionado se encienda, si hay otro fogón apagado, se desplazará al apagado.
  - S3: Cuando todos los fogones estén encendidos, mantendrá su posición.
  - S4: Siempre que no esté en movimiento, atacará a Yema con sus espátulas, intentando evitar que accione los botones.
  - S5: La interfaz de la vida del jefe final se mostrará únicamente en la arena de combate.
- **NPC (Non Playable Characters)**
  - N1: Al acercarse cualquier jugador a la zona de interacción del personaje no jugable, se activará el diálogo correspondiente.
- **Botón**
  - B1: Al ser accionado por Yema, se mantendrá presionado durante 0.2 segundos, hasta que se desactive.
- **Interruptor**
  - IN1: Al entrar en contacto con cualquier jugador, se activará, realizando una acción determinada.
- **Elemento interactivo**
  - E1: Cuando Clara interactúe con él mediante proximidad y presionado de teclas, se realizará una acción determinada.

- **Interfaz**
  - I1: Cada menú debe mostrar una o más opciones seleccionables, a excepción del menú de créditos.
  - I2: Cada botón debe realizar la acción correspondiente a su función.
  - I3: Los menús de selección y de opciones deben tener un botón para retroceder.
  - I4: El botón de “Salir” solo debe estar en el menú principal, y debe cerrar la aplicación.
  - I5: El menú de créditos deberá mostrar un texto, y transitar al menú principal a los cinco segundos.

3.1.2. Requisitos no funcionales

A continuación, se enumeran los requisitos no funcionales:

- **RNF1:** La interfaz mostrará el botón seleccionado con un color diferente al resto de botones.
- **RNF2**: El juego debería poder ser ejecutado en un ordenador Windows con al menos un procesador i5 y 4GB de RAM.
- **RNF3**: Las habilidades aprendidas por el mapa serán útiles para la pelea contra el jefe final.
- **RNF4**: Los controles deben ser sencillos. Debe ser una experiencia cómoda para el jugador.
- **RNF5**: El juego debe tener una ambientación animada pero unos diálogos y un trasfondo oscuro.

3.2. Modelado conceptual

Usando uno de los lenguajes de modelado conceptual más comunes, UML (Unified Model Language), podemos visualizar, especificar, construir y documentar los componentes de un sistema de software. El siguiente diagrama muestra las relaciones entre las distintas clases y objetos del sistema en la implementación original. En este contexto, un diagrama de clases bien estructurado es fundamental para la gestión eficiente de la arquitectura del juego, ya que define cómo se interrelacionan los diferentes componentes y facilita la planificación de nuevas características.

Como podemos observar, el diagrama presenta las relaciones entre las clases, muchas de ellas complejas. Esto no solo añade una capa de complejidad al diseño, sino que también genera una dependencia fuerte entre los diferentes componentes, lo que dificulta la identificación y el aislamiento de problemas específicos.

En un entorno como el desarrollo de videojuegos, donde la iteración y la incorporación de nuevas funciones son constantes, una estructura como la presentada puede llevar a problemas significativos en la gestión del código.

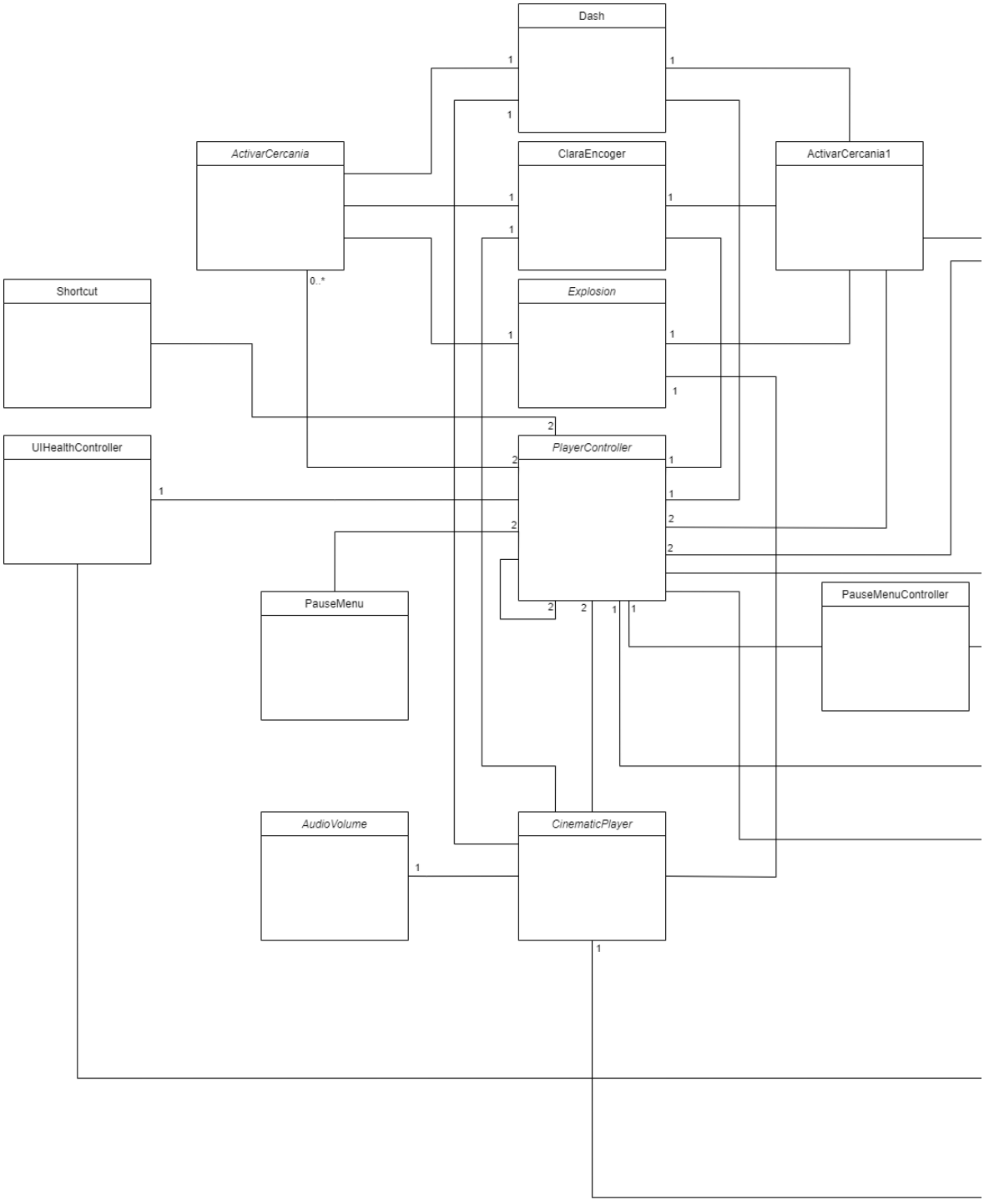


Figura 15 Diagrama de clases previo a la refactorización parte 1

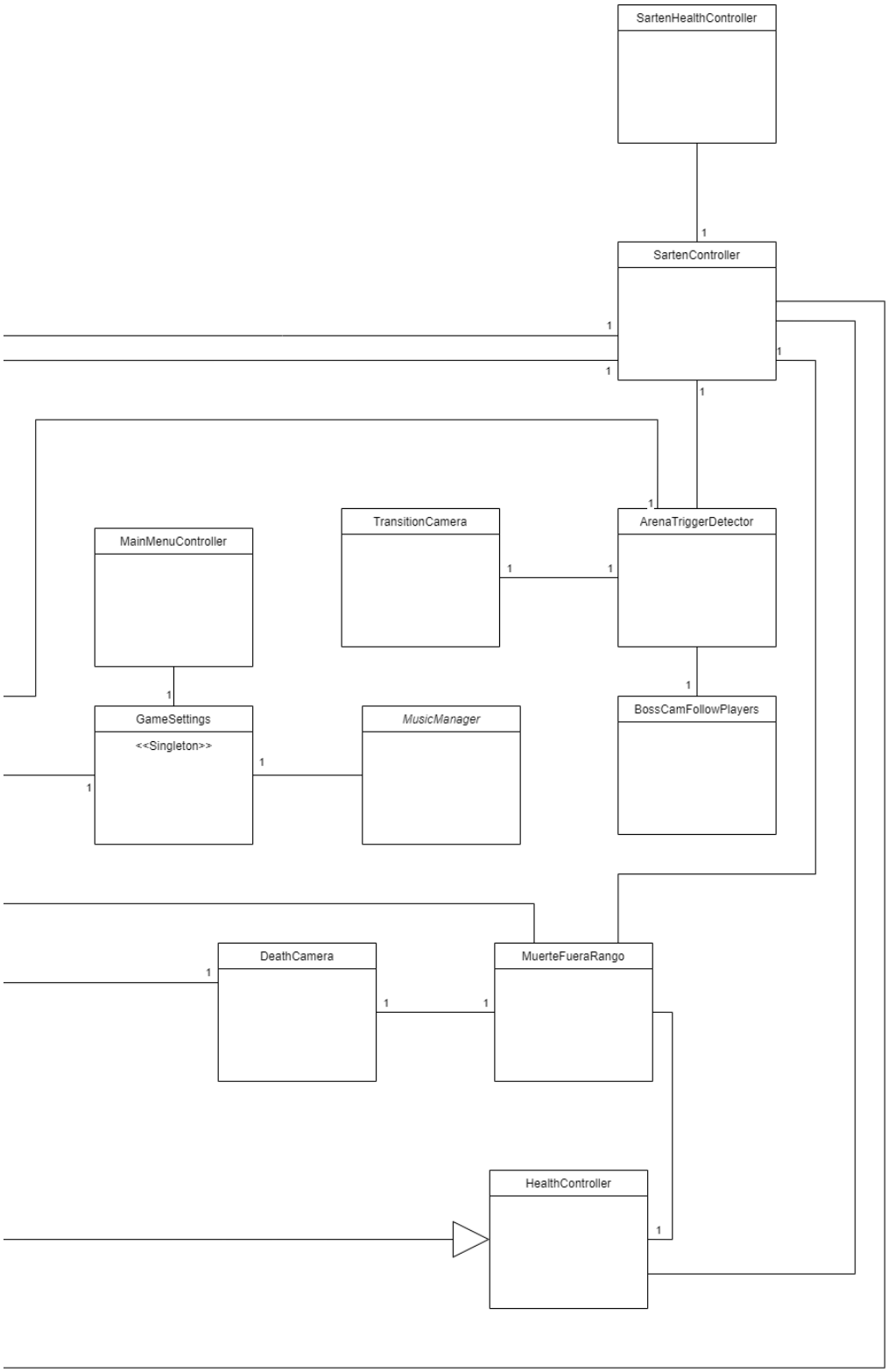
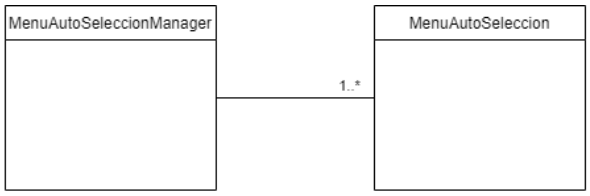


Figura 16 Diagrama de clases previo a la refactorización parte 2



Clases sin relaciones directas con otras clases				
AreaDetector	Atraer	CajonLaberinto	Camara	CamaraCambioDeAngulo
ControladorSeleccionMandos	Hundir	*MandoDialogo*	Moverse	Outline
ParticlePlayer	SplashController	*StopBroom*	TriggerChangeShader	Victoria

Figura 17 Diagrama de clases previo a la refactorización parte 3

Este diagrama no tiene en cuenta las llamadas a las instancias de las clases que siguen el patrón Singleton, ya que complicaría su visibilidad.

La estructura del diagrama de clases de la Figura 15, la Figura 16 y la Figura 17 es tan compleja que añadir nuevas funcionalidades requeriría modificar varias clases y relaciones, lo que aumentaría el riesgo de introducir errores y afectaría a la estabilidad del videojuego.

3.3. Problemas previos a la refactorización

En este apartado, se detallan las métricas recopiladas antes de la refactorización para estudiar el estado de la base del código del videojuego, tales como las líneas de código por clase, líneas de código por función, la complejidad ciclomática (Blanes, 2018) de los métodos y el número de relaciones entre clases.

3.3.1. Cálculo de métricas

El análisis del código fuente revela lo siguiente:

Número de ficheros de código fuente	40
Cantidad total de líneas de código	3650
Número medio de líneas de código por fichero	91,25
Promedio de las líneas de código de las funciones	13,106

Tabla 3 Análisis de líneas totales y medias previos a la refactorización

A continuación, se muestra una tabla con los siguientes datos:

- Nombre de la clase.
- Líneas de código por clase.
- Nombre de la función.
- Líneas de código por función.
- Complejidad ciclomática por función.

Clase/Método	Líneas	C.C.
ActivarCercania	101	
Update	42	8
EndDialogue	16	2
StartTalking	4	1
NoTalk	4	1
ActivarCercania1	80	
StartDialogue	23	3
EndDialogue	16	2
AreaDetector	45	
Update	28	4
OnDrawGizmosSelected	5	1
ArenaTriggerDetector	127	
Update	20	4
SetDialog	23	5
ResetBossFight	21	2
OnTriggerEnter	9	3
OnTriggerExit	13	4
Atraer	112	
SkillSquare	3	1
Update	8	3
NotGrab	4	1
Shoot	16	3
StartAttraction	4	1
MarkObject	9	1
OnTriggerEnter	6	2
StopAttraction	14	2
AttractObject	11	2
AudioVolume	24	
Start	5	1
Update	3	1

Clase/Método	Líneas	C.C.
BossCamFollowPlayers	29	
EnableMovement	5	1
DisableMovement	6	1
CajonLaberinto	32	
OnTriggerEnter	20	5
Camara	25	
LateUpdate	13	2
CamaraCambioDeAngulo	125	
Start	24	4
Update	23	2
OnDrawGizmos	37	2
CinematicPlayer	82	
Awake	5	2
EndReached	22	2
Update	10	2
PlayVideo	22	2
ClaraEncoger	94	
Start	19	3
OnCircle	16	3
Shrink	20	4
ResetSize	10	1
ControladorSeleccionMandos	131	
Awake	14	4
OnMove	28	6
Update	23	12
StartGame	11	5
Dash	51	
Start	4	1
CircleDash	3	3
Update	3	1

DashI	10	4
DeathCamera	24	
ResetPosition	6	2
DestroyMe	3	1
Explosion	52	
SkillSquare	3	1
Explode	10	5
NoEx	4	1
GameSettings	68	
getSFXVolume	4	1
getMusicVolume	4	1
LoadGameSettings	12	1
HealthController	50	
Start	3	1
Update	3	2
ResetHealth	3	1
TakeDamage	3	1
GetHealth	3	1
GetMaxHealth	4	1
GetHealthPercentage	3	1
Hundir	53	
Start	3	1
Update	6	5
OnTriggerEnter	4	2
OnTriggerExit	4	2
MainMenuController	213	
LoadHistoria	4	1
ExitGame	3	1
LoadArcade	4	1
LoadCredits	3	1
AbrirOpciones	4	1
CerrarOpciones	4	1
AbrirSeleccion	4	1
CerrarSeleccion	4	1
Start	35	3
Play	3	1
SetMasterVolume	5	1
SetMusicVolume	5	1
SetSFXVolume	5	1
SetQuality	5	1
SetScreenResolution	10	1
SetTargetFrameRate	6	1
SetFullScreen	11	1
SetVSync	5	3
SaveSettings	11	7
LoadSettings	3	1
QuitGame	3	1
MandoDialogo	82	
Update	35	9
OnMove	3	1
CloseDialog	7	3
OnJump	14	5
MenuAutoSeleccion	138	
Update	25	5
Select	31	5
OpenDropdownAfterDelay	7	3
Volver	6	2
Mover	42	10
MenuAutoSeleccionManager	82	
Update	24	11

OnBack	14	5
OnMove	4	1
OnJump	14	5
Moverse	52	
Start	4	1
Update	25	5
OnCollisionEnter	8	2
MuerteFueraRango	86	
Start	4	1
OnTriggerStay	17	4
Respawn	14	1
ResetPosition	27	5
MusicManager	50	
Start	3	1
Update	3	1
PlayMenuMusic	4	1
PlayGameMusic	4	1
Outline	309	
Awake	17	1
OnEnable	11	2
OnValidate	15	6
Update	6	2
OnDisable	11	2
OnDestroy	5	1
Bake	18	3
LoadSmoothNormals	39	7
SmoothNormals	32	10
CombineSubmeshes	15	3
UpdateMaterialProperties	36	6
ParticlePlayer	21	
PlayIzq	3	1
PlayDer	3	1
PauseMenu	94	
Start	8	2
OnStart	13	3
Update	6	2
PauseGame	19	4
ResumeGame	19	4
PauseMenuController	29	
LoadMainMenu	8	2
Start	3	1
PlayerController	310	
Start	15	5
OnMove	16	2
OnJump	4	1
Grounded	3	1
Update	46	11
OnTriggerEnter	38	12
OnCollisionEnter	15	5
CheckDeath	50	7
MakeInvulnerable	35	6
MakeVisible	6	2
MakeInvisible	6	2
EndInvulnerability	3	1
SartenController	631	
StartFight	4	1
StopFight	23	3
Start	11	2
GoToCredits	5	1
LoadScene2	6	2

LoadScene2E	4	3
Stomp	15	2
EndStomp	5	1
SpawnProp	25	6
LaunchFork	40	3
MoveForkTime	53	10
MoveToPlayerStomp	8	1
MoveTimeStomp	11	4
DashToPlayer	14	2
MoveTime	18	7
Update	24	8
FixedUpdate	68	17
Mix	3	1
Smash	3	1
EndInnerAttack	29	5
CheckDizzy	19	3
GetNearestPlayer	14	3
GetFurthestPlayer	14	3
OnTriggerStay	32	10
OnCollisionStay	9	3
OnCollisionEnter	10	3
OnTriggerEnter	20	5
OnTriggerExit	12	4
OnCollisionExit	8	3
SartenHealthController	25	
Start	3	1
OnTriggerStay	9	3
Shortcut	27	
Update	12	3
SplashController	36	
Update	11	3
OnButton	7	1
StopBroom	31	
OnCollisionEnter	18	4
TransitionCamera	32	
ResetPosition	13	3
DestroyMe	3	1
TriggerChangeShader	19	
OnTriggerEnter	8	3
UIHealthController	22	
Update	3	1
Victoria	56	
OnTriggerEnter	17	4
TeletransportarYDesactivar DespuesDeEspera	24	6

Tabla 4 Lista de clases y métodos previos a la refactorización

3.3.2. Cálculo del número de relaciones

En este subapartado, se contarán el número de relaciones que tiene cada clase con otras:

- ActivarCercania: 4
- ActivarCercania1: 5
- AreaDetector: 0
- ArenaTriggerDetector: 4
- Atraer: 0
- AudioVolume: 1
- BossCamFollowPlayers: 1
- CajonLaberinto: 0
- Camara: 0
- CamaraCambioDeAngulo: 0
- CinematicPlayer: 6
- ClaraEncoger: 4
- ControladorSeleccionMandos: 0
- Dash: 4
- DeathCamera: 2
- Explosion: 3
- GameSettings: 3
- HealthController: 3
- Hundir: 0
- MainMenuController: 1
- MandoDialogo: 0
- MenuAutoSeleccion: 1
- MenuAutoSeleccionManager: 1
- Moverse: 0
- MuerteFueraRango: 4
- MusicManager: 1
- Outline: 0
- ParticlePlayer: 0
- PauseMenu: 1
- PauseMenuController: 2
- PlayerController: 14
- SartenController: 6
- SartenHealthController: 1
- Shortcut: 1
- SplashController: 0
- StopBroom: 0
- TransitionCamera: 1
- TriggerChangeShader: 0
- UIHealthController: 2
- Victoria: 0

3.4. Identificación y análisis de soluciones posibles

Una vez analizados los cálculos de métricas y los cálculos del número de relaciones, se observa que se encuentran una gran cantidad de clases con excesiva complejidad ciclomática, que supera el límite de diez. También hay una alta relación entre clases, como por ejemplo en la clase **PlayerController**.

A partir de la lista previa de problemas, se presenta algunas posibles soluciones:

- **Desarrollo centrado en el mantenimiento**
  - Esta solución se centra en la reestructuración de clases, así como su refactorización para reducir la cantidad de líneas de código y mejorar el rendimiento del videojuego. Aun así, no desaparece la posibilidad de añadir contenido nuevo o mejorarlo.
  - Ventajas:
    - Mejora de la estabilidad y reducción de la cantidad de bugs o errores durante la ejecución del juego y durante la compilación.
    - Reducción de dependencias externas, ya que disponer de código no desarrollado de manera interna aumenta la posibilidad de tener un código no óptimo para las necesidades del juego y además puede realizar más acciones de las que debería.
    - Optimización de rendimiento para permitir que el juego se ejecute en dispositivos menos potentes, aumentando el rango de jugadores.
    - Facilidad de un mantenimiento futuro, reduciendo la complejidad de añadir nuevo contenido y la deuda técnica del proyecto.
  - Desventajas:
    - Falta de nuevo contenido: el enfoque en el mantenimiento puede considerarse una falta de innovación, ya que se dedica menos tiempo a crear nuevas características o contenidos.
    - Modificar y refactorizar el código existente puede introducir nuevos errores o problemas. Requiere un esfuerzo adicional en pruebas y depuración para asegurar que los cambios no afecten negativamente al juego.

- **Desarrollo centrado en el contenido**
  - Esta solución se centra en el desarrollo de contenido nuevo, como niveles o personajes nuevos. Se trata de expandir la experiencia haciendo las mínimas modificaciones a la estructura del juego original.
  - Ventajas:
    - Mejora el juego desde la perspectiva del jugador ya que para el observador externo el cambio es más notorio.
    - Aumenta la duración del juego.
  - Desventajas:
    - Puede aumentar el tamaño del juego, así como afectar al rendimiento de este.
    - La adición de contenido puede complicarse exponencialmente, lo cual requeriría un esfuerzo considerable a largo plazo.

3.5. Solución propuesta

Tras la identificación y el análisis de las posibles soluciones, se ha decidido que la solución propuesta es centrarse en el mantenimiento, reestructurando las clases, eliminando código innecesario y añadiendo nuevas funcionalidades. Esta elección se basa en la necesidad de mejorar la estabilidad, rendimiento y sostenibilidad a largo plazo del proyecto. Esta decisión tiene en cuenta las ventajas y desventajas de las posibles soluciones, priorizando la optimización del código y la reducción de la deuda técnica para permitir una futura expansión más eficiente.

Al reestructurar el videojuego se facilita la introducción de nuevas características sin comprometer su estabilidad, y también permite mejorar la legibilidad y mantenibilidad del código, parte imprescindible para continuar con el desarrollo en el futuro.

3.6. Plan de Trabajo

Este Trabajo de Fin de Grado se dividirá en las siguientes secciones:

1. Análisis del Código y Pruebas:
   a. Identificar las partes del código que necesitan refactorización y las dependencias externas que deben ser revisadas o eliminadas, así como implementar pruebas iniciales que permitan tener contexto de la situación inicial del proyecto.
   b. Pruebas unitarias de las clases del juego, para comprobar que, al refactorizar, el comportamiento del juego sea el mismo que el que había previo a la reescritura del código.

c. Análisis de la relación entre clases, con el objetivo de reducir dependencias cíclicas y aplicar el principio de responsabilidad única
d. Medición de la complejidad ciclomática del proyecto, para descubrir dónde están las mayores debilidades del juego y posteriormente comparar con el resultado final de la refactorización.

2. Implementación de una nueva estructura de clases:
   a. Crear una nueva estructura de clases que facilite el mantenimiento y la expansión futura del juego.
   b. Introducción de un GameManager que se encargue de manejar todas las dependencias de cada clase, reduciendo la cantidad de búsquedas a otros objetos de la escena y reduciendo la cantidad de relaciones entre clases.
   c. Implementar herencias entre clases como por ejemplo las de los personajes.
   d. Eliminación de código redundante.
3. Implementación de contenido nuevo:
   a. Nuevas habilidades de personajes.
   b. Nuevas interacciones con el entorno.
   c. Mejora de la mecánica principal del jefe final.
   d. Nuevos puzles en la escena.
4. MVP (Minimum Viable Product) y Pruebas:
   a. Pruebas Unitarias tras la refactorización.
   b. MVP.
   c. Pruebas de validación con usuarios.

Esta propuesta permite asegurar una mejora en el rendimiento del juego, así como establecer una base sólida para un futuro desarrollo más sostenible y eficiente.

Como se indicó en el apartado 1.4 se ha seleccionado la metodología Kanban para el desarrollo del proyecto. A continuación, se muestra un tablero con las tareas principales que se ejecutarán a lo largo de este proyecto (Figura 18) y un diagrama del plan de trabajo (Figura 19)

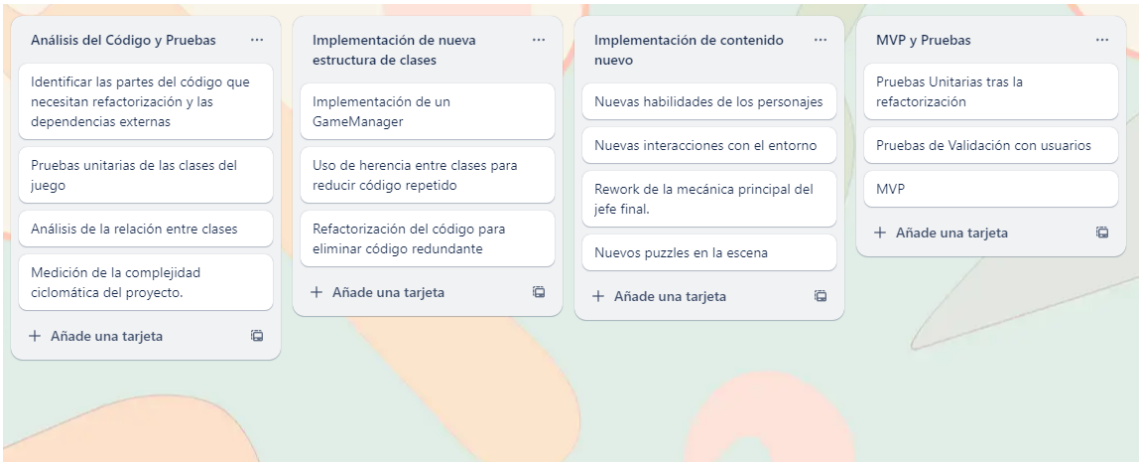


Figura 18 Captura del tablero de Trello, con el plan de trabajo planteado

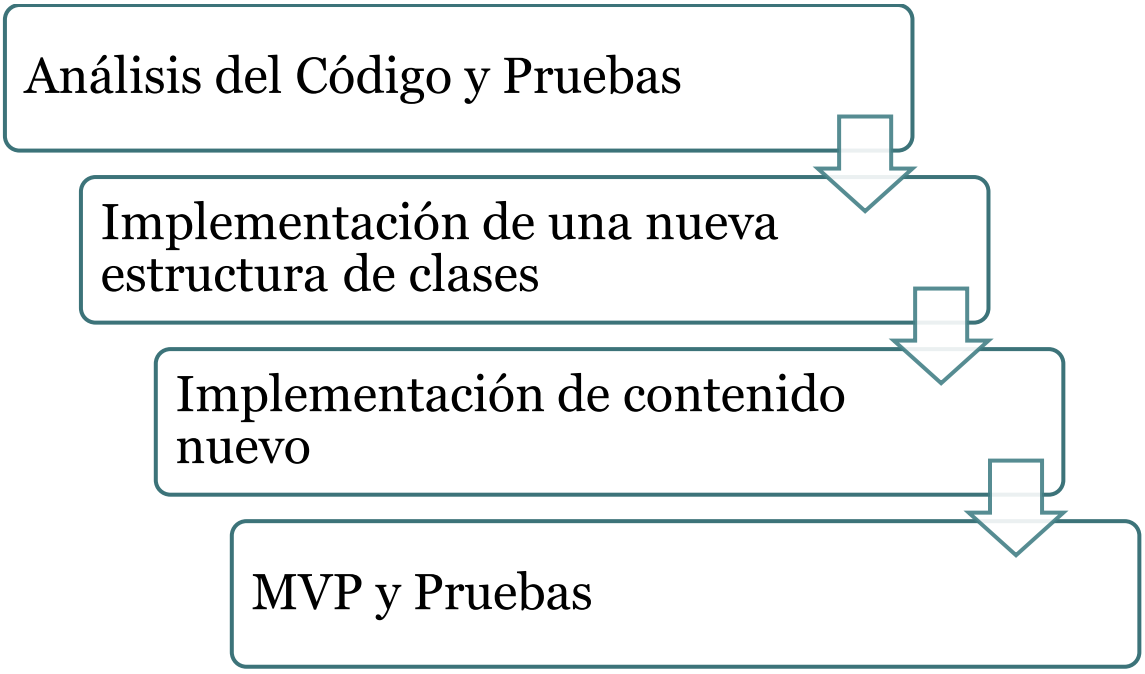


Figura 19 Diagrama del plan de trabajo

4. Diseño de la solución

En este capítulo se explicará la arquitectura del sistema y se detallará el diseño y la tecnología utilizada.

4.1. Arquitectura del sistema

En este TFG se propone implementar una arquitectura dividida en las siguientes tres capas:

- Presentación: En esta capa están contenidas todas las clases relacionadas con la interfaz.
- Lógica: En esta capa se encuentran las clases que controlan el movimiento de los personajes, los eventos de los elementos interactivos y la interacción con personajes no jugables.
- Persistencia: En esta capa se guardan los datos del juego, como la configuración del usuario, el estado de salud de los enemigos o el estado del diálogo entre personajes.

4.1.1. Capa de presentación

En este videojuego, la capa de presentación incluye los siguientes componentes:

- **Menú principal:** Es la pantalla que aparece al iniciar la aplicación (Figura 20). En ella se visualizan cuatro botones. El botón “Historia”, que lleva a la primera escena del juego, la cocina. El botón “Opciones” que carga la escena del menú de opciones. El botón “Créditos” que carga el menú de los créditos. Y por último el botón de “Salir”, el cual cierra la aplicación. (Requisito Funcional J1, I1, I2, I4)
- **Menú de selección:** Es la pantalla en la que cada jugador elige su personaje, con dos iconos representando a dos mandos, en los que el jugador elige una dirección para confirmar su elección (Figura 21). También hay un botón “Atrás” que regresa al menú principal. (Requisito Funcional J1, I1, I2, I3)
- **Menú de opciones:** Esta pantalla muestra tres controles deslizantes o sliders en inglés, que sirven para controlar el nivel de volumen, tanto general como de música y efectos de sonido (Figura 22). También, como en el menú de selección, hay un botón “Atrás” que regresa al menú principal. (Requisito Funcional J1, I1, I2, I3)

- **Menú de créditos:** Esta pantalla muestra un texto con los nombres de los autores del videojuego (Figura 23). No dispone de ningún botón, ya que a los cinco segundos regresa al menú principal automáticamente. (Requisito Funcional I5)
- **Menú de agradecimientos:** Esta pantalla (Figura 24) tiene dos botones, el botón de “Rellenar formulario”, que abre un enlace web para rellenar una encuesta, que se presentará en el apartado 7, y un botón “Volver al menú”, el cual cargará de nuevo la escena del menú principal. (Requisito Funcional J1, I1, I2))
- **Interfaz de la vida del jefe:** Esta interfaz (Figura 25) no es interactiva, ya que muestra en la parte superior de la pantalla una barra de vida del jefe final. (Requisito Funcional S5)
- **Interfaz de los diálogos:** Esta interfaz (Figura 26) es interactiva mediante la pulsación de cualquier tecla, ya que muestra un cuadro de diálogo que va progresando letra a letra a no ser que reciba una entrada del usuario, que hará que el texto aparezca completo, sin la animación descrita anteriormente. Una vez el texto se haya mostrado por completo, el usuario, con cualquier entrada, pasará al siguiente cuadro de diálogo hasta que no quede más texto por mostrar. (Requisito Funcional J1)

4.1.1.1. Menú principal

Figura 20 Captura del Menú Principal

4.1.1.2. Menú de selección

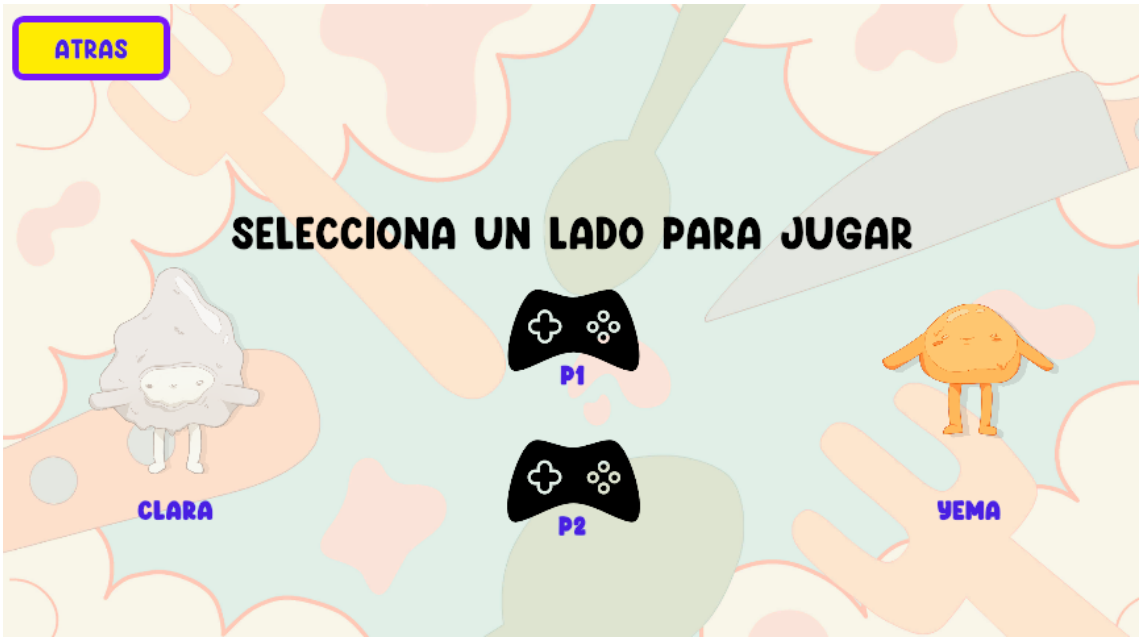


Figura 21 Captura del Menú de Selección

4.1.1.3. Menú de opciones

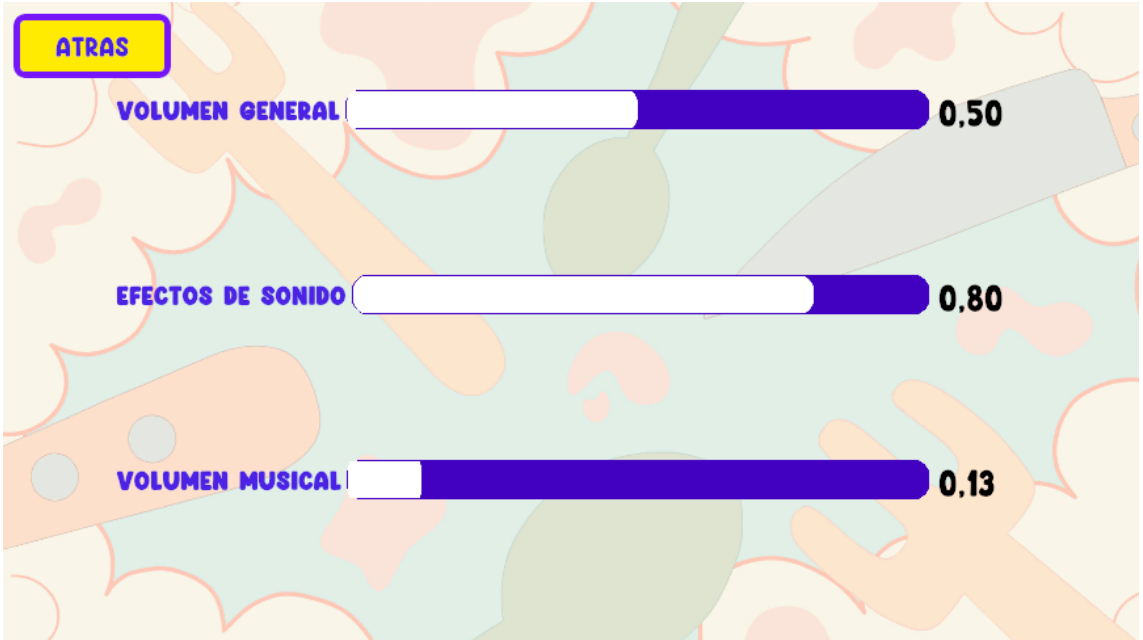


Figura 22 Captura del Menú de Opciones

4.1.1.4. Menú de créditos

Figura 23 Captura del Menú de Créditos

4.1.1.5. Menú de agradecimientos

Figura 24 Captura del Menú de agradecimientos

4.1.1.6. Interfaz de la vida del jefe

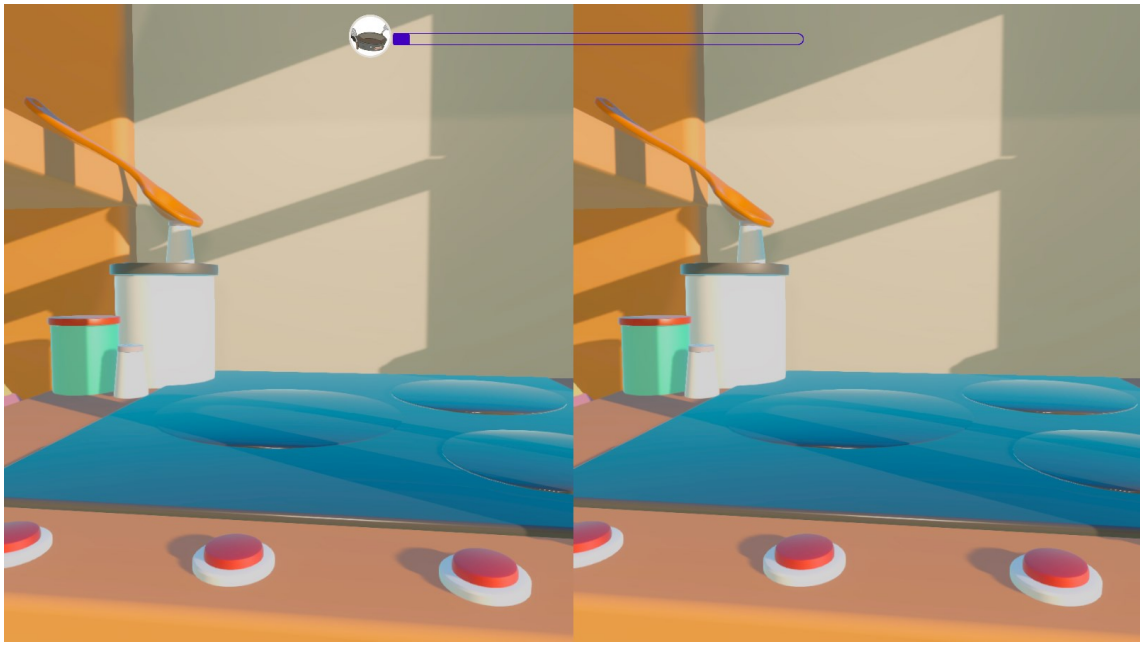


Figura 25 Interfaz de la vida del jefe

4.1.1.7. Interfaz de los diálogos

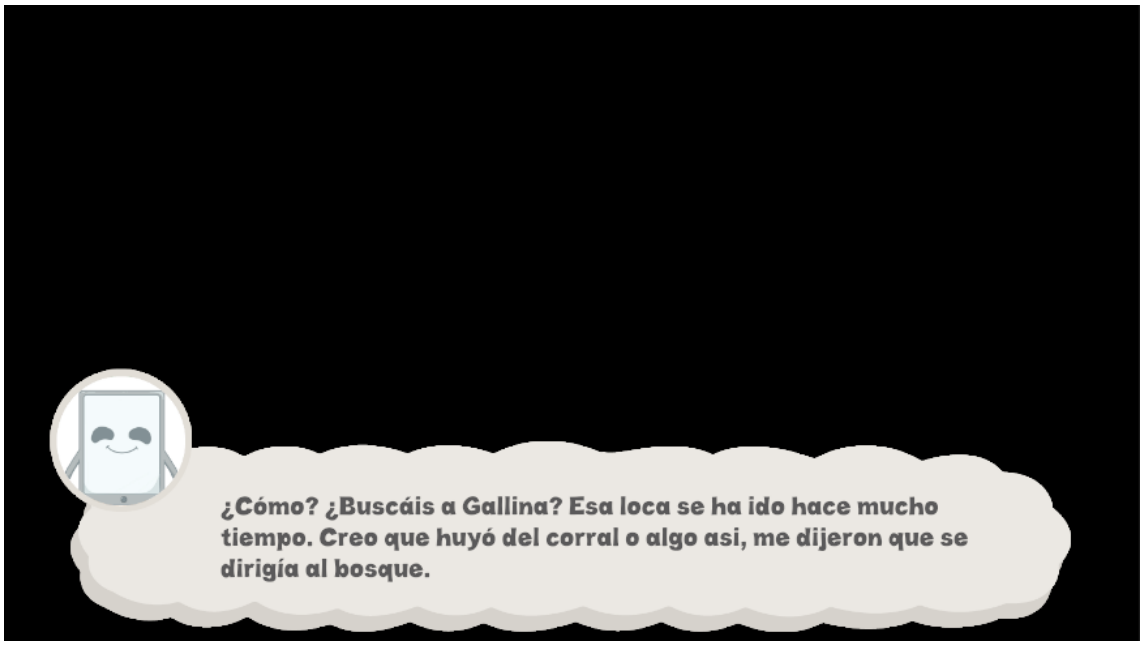


Figura 26 Interfaz de los diálogos

4.1.2. Capa lógica

En este apartado, se detallará la capa lógica, que se encarga de gestionar el estado del juego, el movimiento de los personajes, sus interacciones y el comportamiento del jefe final.

- **Gestión de los personajes:** Este componente se encarga del control de los personajes. Desde la posición inicial de cada personaje hasta su movimiento continuo a través del entorno, este sistema asegura que la mecánica de movimiento, incluyendo caminar, correr, saltar y cualquier habilidad especial, funcione de manera fluida y precisa. Además, gestiona las transiciones entre estados (como estar en reposo, caminando, corriendo o realizando acciones específicas) para cada personaje.
- **Gestión de interacciones:** Este componente se encarga de gestionar todas las interacciones de los personajes con el entorno. Esto abarca desde colisiones físicas básicas hasta interacciones más complejas como las colisiones tipo *trigger* que activan eventos específicos en el juego. También incluye la capacidad de interaccionar con objetos y elementos del escenario, como por ejemplo interruptores o botones.
- **Gestión del jefe final:** Se encarga de coordinar los movimientos y ataques del jefe, gestionar su barra de vida y manejar cualquier fase específica del combate. Además, supervisa las interacciones especiales que pueden surgir entre los personajes principales (Clara y Yema) y el escenario durante la confrontación con el jefe final.

Cada componente cumple un papel fundamental en la experiencia de juego, asegurando la coherencia interna del mundo del juego y la respuesta precisa a las acciones del jugador.

4.1.3. Capa de persistencia

En este videojuego, la capa de persistencia se encarga de que la información relevante se guarde de manera confiable y accesible. Esta capa maneja el guardado y la carga de partidas y la persistencia de las preferencias del usuario, fundamental para ofrecer una experiencia personalizada.

- **Manejo de preferencias del usuario:** La capa de persistencia gestiona las preferencias del usuario, como son las diferentes configuraciones de audio, tanto el volumen maestro como el de efectos de sonido y el de la música. Para esto, se utilizan herramientas como PlayerPrefs, que almacenan de manera eficiente datos simples en un formato accesible los tipos nativos del lenguaje de programación.
- **Uso de ScriptableObjects para la vida del jefe final:** Para gestionar la complejidad del jefe final, se ha implementado un ScriptableObject, que

permite definir y ajustar dinámicamente su salud. Esto no solo facilita la configuración durante el desarrollo, sino que también mejora la flexibilidad para ajustar y equilibrar la dificultad si fuera necesario.

4.2. Diseño detallado

En este apartado, se va a describir y detallar los sistemas que se han diseñado en este videojuego. Estos son: el sistema de los jugadores, el sistema de interacciones con el entorno, el sistema de control de niveles, el sistema del jefe final, el sistema de cámaras y el sistema de interfaces.

Como se puede observar en la Figura 27 al final de este apartado, se está ante un sistema complejo, por lo que se detallarán y explicarán aquellas clases que sean más importantes.

4.2.1. Sistema de los jugadores

El sistema de los jugadores se encarga de gestionar todas las entradas de los controladores de los usuarios, incluyendo el movimiento, el uso de habilidades, el salto y la reaparición al ser golpeado.

La clase **PlayerController** se encarga de recibir los datos de la clase **CharacterInput**, específicamente de la clase **ControllerInput**, ya que el juego se controla con mandos. Gracias a estas clases, los jugadores podrán moverse por el nivel, saltar y usar sus habilidades.

Dependiendo del botón que se presione, se usará una habilidad especial del personaje o se interactuará con los objetos interactivos del mapa.

La clase **PlayerController** es heredada por dos clases hijas, **ClaraController** y **YemaController**, los cuáles implementan las habilidades específicas de cada personaje. Al usar el botón de la habilidad, Clara se adherirá a una pared pegajosa al usar el botón de la habilidad y Yema aplastará todo lo que tenga debajo al usar el botón de habilidad al estar en el aire.

La clase **AnimatorController** se encarga de recibir datos del **PlayerController** y comunicarse con el Animator para ejecutar una animación u otra.

4.2.2. Sistema de interacciones con el entorno

El sistema de interacciones con el entorno se encarga de gestionar el comportamiento de los elementos interactivos de la escena al ser activados por un jugador.

La clase **Interactable** es una clase con un método abstracto, que sobrescriben las demás clases hijas para realizar una acción concreta dependiendo del tipo de **Interactable** que sea. Además, la clase **StickyInteractable** es únicamente activable con el personaje de Clara, mientras que la clase **ExplosionInteractable** es únicamente activable con el personaje de Yema. Las clases **IroningBoardInteractable** y **SinkInteractable** no son exclusivas de ningún personaje, siendo interactivas para ambos jugadores.

4.2.3. Sistema de control de niveles

El sistema de control de niveles está principalmente compuesto por **GameManager** y **SceneController**, siendo **GameManager** el controlador general de la partida y **SceneController** el controlador específico de cada nivel, el cual determina las posiciones de inicio de los personajes y del jefe final.

GameManager se encarga de dar a conocer a **SceneController** a todos los componentes del juego. **GameManager** implementa el patrón **Singleton**, que hace accesible una instancia única de GameManager desde cualquier parte del código. A su vez, **GameManager** también controla el nivel de volumen del componente **MusicVolume**, ya que al ejecutarse una cinemática, el volumen de la música deberá bajar a cero.

Además de indicar las posiciones iniciales, **SceneController** también se encarga de dar a conocer algunos componentes vitales del nivel, como el controlador de la cámara que se esté usando en cada momento o el contenedor de la vida del jefe final, con el objetivo de comunicárselo a la interfaz que muestra su porcentaje de vida.

4.2.4. Sistema del jefe final

El sistema del jefe final está compuesto por cinco clases: **CottonController**, **SartenController**, **FirePoint**, **SartenButtonRandomizer** y **SartenButton**. Se encarga de poder completar de principio a fin la batalla contra el jefe final.

La clase **SartenController** se encarga del comportamiento de Sartén, y depende únicamente de tres instancias de la clase **FirePoint**, las cuales se utilizarán como puntos de referencia para que Sartén se mueva.

La clase **SartenButtonRandomizer** se encarga de aleatorizar la lista de **FirePoint** que se activan con cada una de las interacciones de Yema con cualquier objeto de la clase **SartenButton**.

Cuando cualquier **FirePoint** tenga un valor activado, se marcará como punto no válido y Sartén saltará a otro punto que sea válido hasta que no le queden más sitios válidos. En ese momento, Clara deberá usar un objeto de la clase **CottonController** para reducir la vida del jefe.

4.2.5. Sistema de cámaras

El sistema de cámaras está compuesto por las clases: **CameraController**, **CamaraCambioDeAngulo**, **CameraBoss**, **CameraFadeController**, **VideoRunner** y **CinematicPlayer**.

CameraController es una clase abstracta de la cual heredan **CamaraCambioDeAngulo** y **CameraBoss**. Esta clase implementa métodos para añadir jugadores a la lista que permite a las diversas cámaras encuadrarse para visionar a todos los jugadores correctamente.

CamaraCambioDeAngulo permite el seguimiento de ambos jugadores de una manera conjunta, aunque no estén cerca, siguiendo una vía determinada previamente, sin control por parte del usuario.

CameraBoss permite el control de la cámara al jugador, ya que durante el combate contra el enemigo, se necesita más control de la situación.

CinematicPlayer es el encargado de reproducir en pantalla los videos de cinemáticas que se le soliciten a lo largo de la partida. A su vez, **VideoRunner** es el encargado de comunicarle a **CinematicPlayer** esa necesidad de reproducir una cinemática y le indica cuál debe reproducirse.

4.2.6. Sistema de interfaces

El sistema de interfaces está compuesto por una clase principal, **UIController**, de la cual heredan cuatro clases más, **MainMenuController**, **SelectorController**, **SliderController** y **ThanksUI**. La clase **CreditsController** también está contenida dentro de este sistema, pero es más independiente, ya que no posee controles, solo muestra información de los créditos.

UIController tiene la función de guardar una lista de elementos interactivos, que no conoce previamente y dependiendo del tipo de elemento, determina un comportamiento u otro.

MainMenuController, **SelectorController**, **SliderController** y **ThanksUI** se encargan de determinar qué elementos interactivos deberá conocer su clase padre.

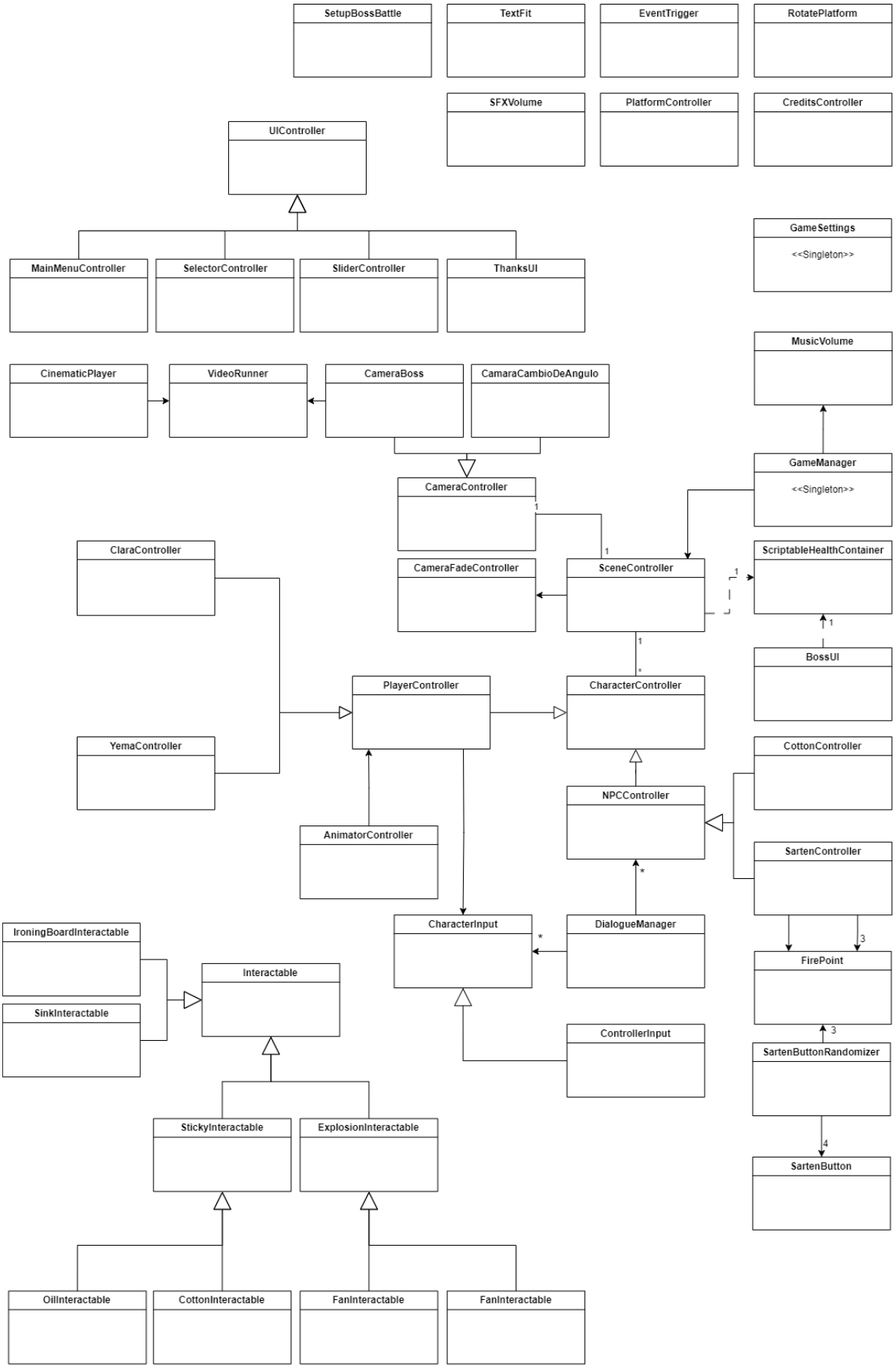


Figura 27 Diagrama de clases de la arquitectura del sistema

4.3. Tecnología utilizada

En el siguiente apartado se comentará y listará las diferentes tecnologías utilizadas para la mejora y el desarrollo de este proyecto.

4.3.1. Unity

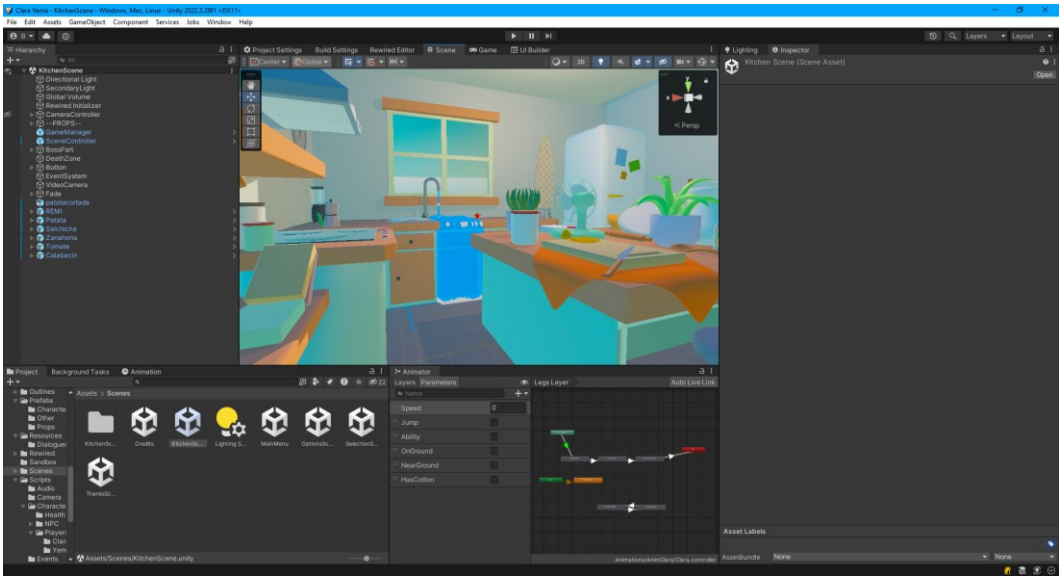


Figura 28 Captura del Motor Unity

Unity (Figura 28) es una herramienta de desarrollo que permite crear juegos y aplicaciones en múltiples plataformas como PC, consolas, móviles y realidad aumentada/virtual. Ofrece una versión gratuita con funcionalidades básicas y varias opciones de suscripción con características avanzadas. Este motor de juegos ayuda a los desarrolladores y diseñadores a trabajar en un entorno visual amigable, lo que resulta fundamental para la creación de experiencias interactivas y atractivas.

Una de las grandes fortalezas de Unity es su capacidad para facilitar la creación de juegos tanto en 2D como en 3D mediante potentes herramientas y recursos. Utiliza un sistema de componentes que permite añadir funcionalidades específicas a los objetos del juego de manera flexible, adaptándose a las necesidades de cada proyecto. La programación en Unity se realiza en C#, con un entorno robusto y eficiente para desarrollar scripts. Además, incorpora motores de física y herramientas de animación avanzadas que simplifican la creación de interacciones realistas y movimientos complejos, lo que es crucial para ofrecer experiencias de juego inmersivas. Unity permite crear Prefabs, que facilitan la instanciación de objetos referencias. Estos objetos permiten clonar un objeto, que, mediante la modificación del original, se propagan los cambios a los clonados.

El ecosistema de Unity incluye una tienda en línea, conocida como la Asset Store, donde se pueden comprar y vender recursos como modelos 3D, scripts, texturas y herramientas que aceleran el desarrollo de los proyectos. Además, Unity cuenta con una comunidad amplia y activa que ofrece soporte, recursos educativos y tutoriales, lo que facilita a los desarrolladores encontrar ayuda y aprender nuevas técnicas. La plataforma también proporciona una documentación detallada que cubre todos los aspectos del motor y su uso, asegurando que los desarrolladores tengan acceso a la información necesaria para maximizar su eficiencia.

Su interfaz intuitiva y la abundancia de recursos facilitan la curva de aprendizaje, permitiendo a los usuarios iterar y prototipar rápidamente gracias a sus herramientas visuales y su modularidad.

4.3.2. Rider

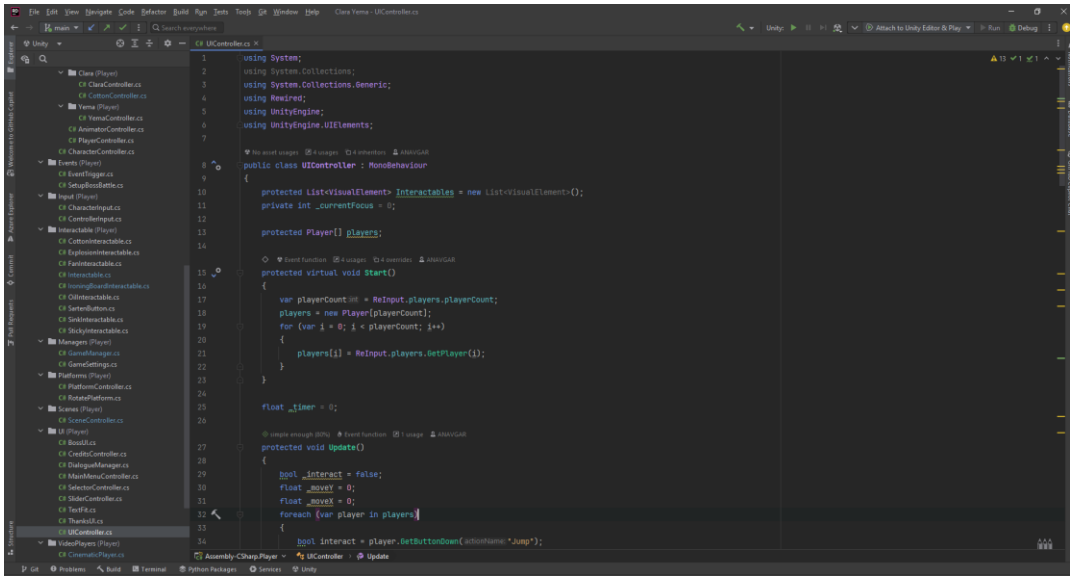


Figura 29 Captura del IDE Rider

Rider (Figura 29) es un potente entorno de desarrollo integrado (IDE) creado por JetBrains, diseñado específicamente para la programación en .NET y compatible con múltiples plataformas como Windows, macOS y Linux. Este IDE destaca por su capacidad para proporcionar una experiencia de desarrollo fluida y eficiente, integrando una gran cantidad de herramientas avanzadas que facilitan el trabajo de los desarrolladores. Gracias a su soporte para C#, VB.NET, ASP.NET, Unity y más, Rider se convierte en una opción versátil para una variedad de proyectos.

Una de las características más notables de Rider es su editor de código inteligente, que ofrece una amplia gama de ayudas contextuales como autocompletado avanzado, refactorización automática y análisis de código en tiempo real. Estas herramientas ayudan a los desarrolladores a escribir código más limpio y eficiente, reduciendo los errores y acelerando el proceso de desarrollo. Además, Rider

se integra de manera impecable con sistemas de control de versiones como Git, proporcionando un flujo de trabajo más coherente y colaborativo.

En resumen, Rider es un IDE excepcional que combina un conjunto completo de herramientas de desarrollo avanzadas con un rendimiento eficiente y una integración sólida con tecnologías clave como .NET y Unity.

4.3.3. Rewired

Figura 30 Logo de Rewired

Rewired (Figura 30) es un plugin para Unity que ofrece una solución avanzada y flexible para la gestión de entradas de usuario en juegos y aplicaciones. Este plugin destaca por su capacidad para soportar una amplia variedad de dispositivos, incluyendo gamepads, teclados, ratones y controladores específicos como volantes o joysticks.

Una de las principales ventajas de Rewired es su sistema de mapeo de controles altamente personalizable, que permite a los desarrolladores asignar funciones a los botones y ejes de cualquier dispositivo de entrada. Además, Rewired ofrece soporte multiplataforma, funcionando de manera consistente en Windows, macOS, Linux, y consolas, lo que facilita el desarrollo de juegos que requieren compatibilidad con múltiples sistemas operativos y dispositivos.

Rewired también proporciona herramientas avanzadas para la configuración y gestión de controles, incluyendo la detección automática de dispositivos y la capacidad de manejar múltiples perfiles de entrada. Su integración con Unity es fluida, permitiendo una implementación rápida y fácil en cualquier proyecto.

En resumen, Rewired es un Asset esencial para los desarrolladores de Unity que buscan mejorar la gestión de entradas de usuario, ofreciendo una solución potente y flexible que soporta un alto número de dispositivos y plataformas.

4.3.4. Adaptive Split Screen

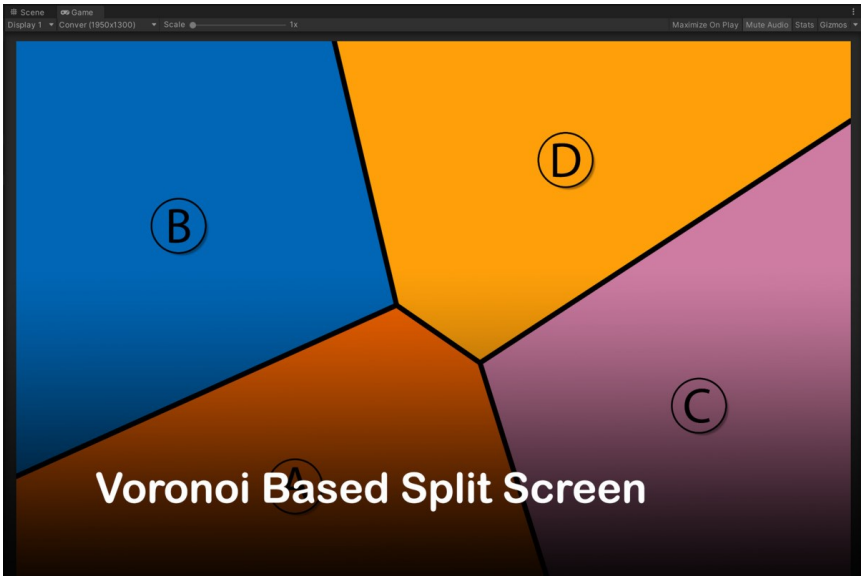


Figura 31 Captura ejemplo de Adaptive Split Screen

Adaptive Split Screen (Figura 31) es un plugin para Unity que facilita la creación de pantallas divididas dinámicas en juegos multijugador local. Este plugin permite ajustar automáticamente la división de la pantalla según la posición y el número de jugadores, proporcionando una experiencia de juego más fluida y atractiva.

Una de las características más destacadas de Adaptive Split Screen es su capacidad para cambiar de manera inteligente el layout de la pantalla utilizando regiones de Voronoi. Esto garantiza que cada jugador mantenga una visibilidad óptima del entorno del juego, mejorando la jugabilidad y la interacción.

En resumen, Adaptive Split Screen es un Asset muy valioso para desarrolladores de Unity que buscan implementar pantallas divididas adaptativas en juegos multijugador, ofreciendo una solución sencilla y eficiente para desarrollar juegos de pantalla compartida.

5. Desarrollo de la solución

En este apartado, se va a explicar y desarrollar la solución propuesta, mostrando cómo se ha llegado desde la propuesta inicial a la solución. A lo largo del apartado, se indicará el requisito que resuelva cada parte.

Se mostrará el desarrollo del nivel, las mecánicas básicas de los personajes, los elementos interactivos, el sistema de diálogos, el jefe final y la interfaz visual.

5.1. Desarrollo del nivel

Al ser un juego en tres dimensiones, el desarrollo de niveles ha sido mediante el posicionamiento manual de objetos por el escenario.

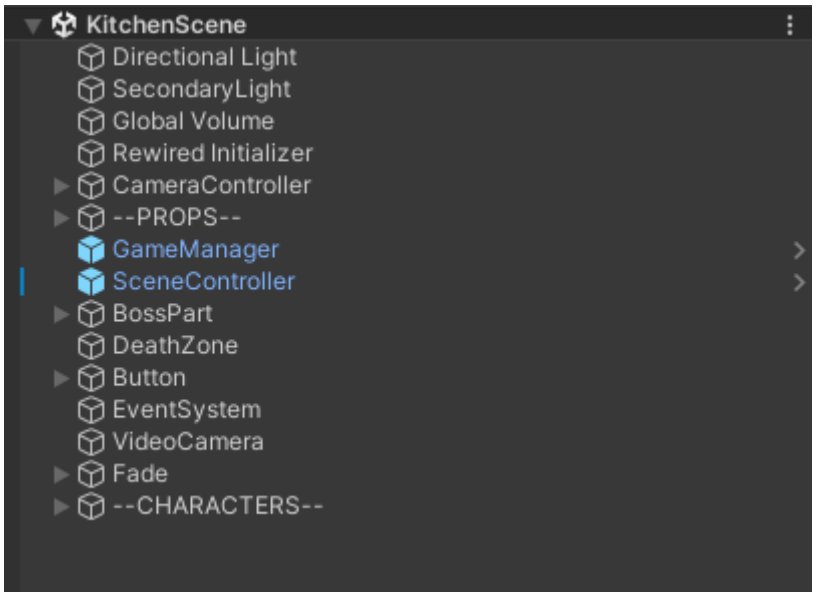


Figura 32 Captura de la jerarquía de la escena

En la Figura 32, observamos la jerarquía de la escena "KitchenScene". La escena se ha dividido en diferentes objetos, que se pueden categorizar de la siguiente forma:

5.1.1. Controladores

- **CameraController:** Este objeto contiene el script **CamaraCambioDeAngulo** (*Figura 35*), y controla la rotación y la posición de la cámara en la escena. Su función principal es asegurar que la cámara

siga de manera fluida la acción dentro del juego, centrando su vista en la posición media de los jugadores.

- **GameManager:** Este objeto es el núcleo de la gestión del juego. Su responsabilidad principal es coordinar y controlar todos los aspectos del juego, desde el inicio hasta el final de la partida. Además, interactúa con otros controladores y componentes para asegurar el correcto funcionamiento del juego. Este componente contribuye a la resolución del requisito funcional J2, J3

- **SceneController:** Este objeto gestiona los aspectos específicos de la escena actual. En "KitchenScene", el SceneController (*Figura 33*) encarga de hacer aparecer a los personajes jugables y no jugables, así como determinar qué cámara se asigna a cada jugador. También maneja la lógica de transición entre escenas y tiene una referencia al objeto que controla el desvanecimiento de la cámara. Este componente contribuye a la resolución del requisito funcional J2.

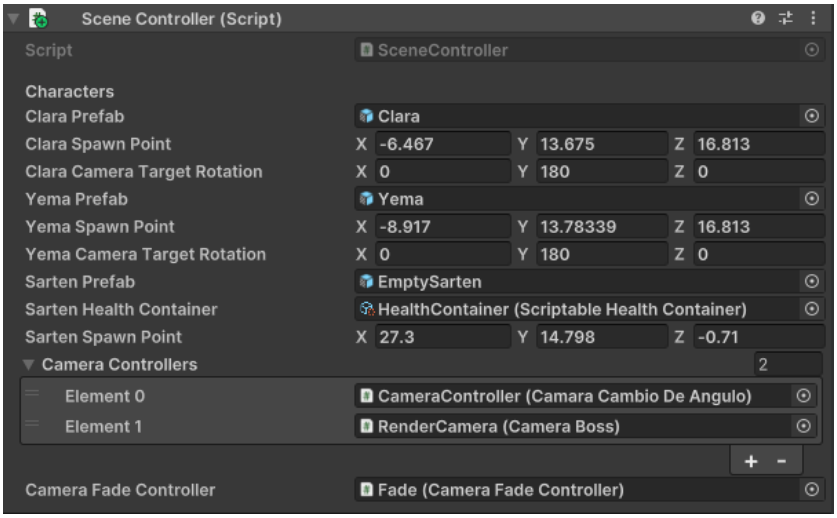


Figura 33 SceneController en el inspector

- **Fade:** Este objeto controla el efecto de desvanecimiento mediante el script **CameraFadeController** (*Figura 34*). Se encarga de asignar que cámara debe mostrar el efecto de desvanecimiento dependiendo del jugador que solicite la acción. Este componente contribuye a la resolución del requisito funcional P3.

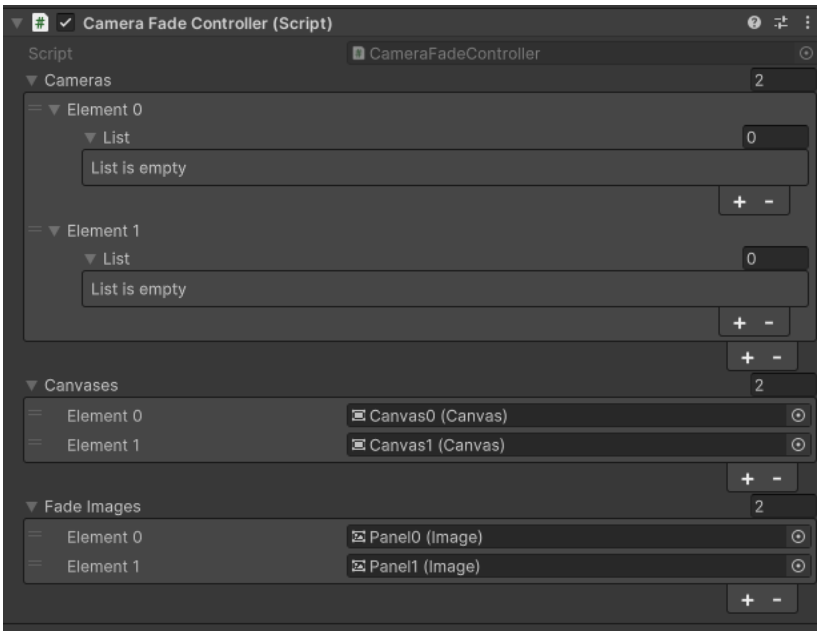


Figura 34 CameraFadeController en el inspector

5.1.2. Organizadores:

- **CameraController:** Este objeto (*Figura 35*) se encarga de guardar las cámaras utilizadas en la partida para que el SceneController pueda acceder a ellas. Además, controla las cámaras para que se posicionen en el punto medio entre los dos jugadores.

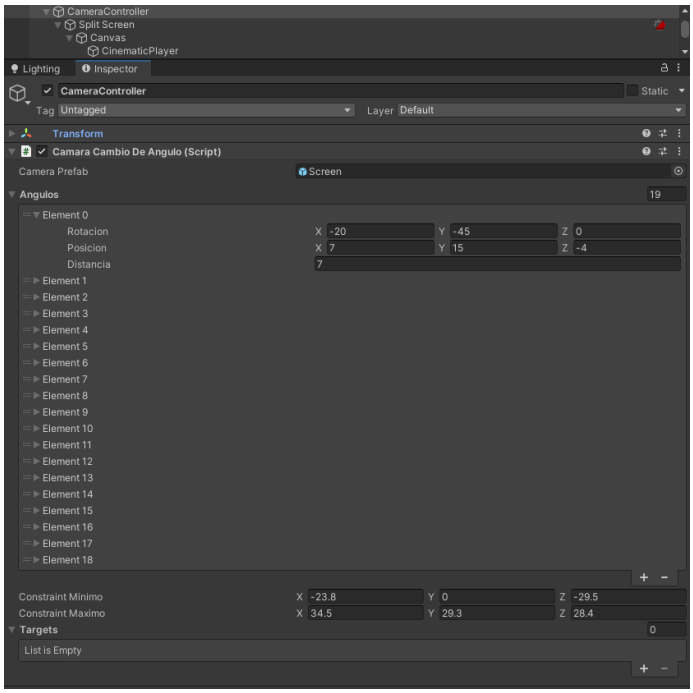


Figura 35 CameraController en el inspector

- **Props:** Este objeto (Figura 36) guarda todos los objetos con los que los jugadores pueden interactuar, desde el suelo hasta los elementos interactivos.

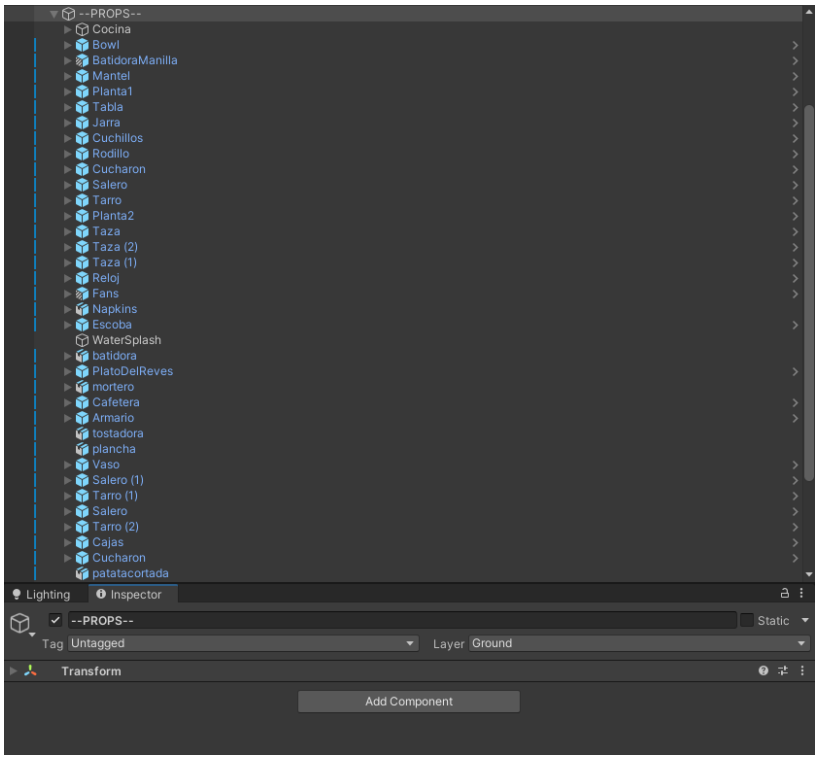


Figura 36 Objeto Props en el inspector

- **BossPart:** Este objeto (Figura 37) organiza todos los elementos del combate final contra el jefe y se encarga de preparar la batalla y habilitar el personaje del jefe. También ejecuta una animación al comenzar la batalla y habilita la cámara del combate, que es diferente a la usada anteriormente. Este componente contribuye a la resolución del requisito funcional J2, P3, S5.

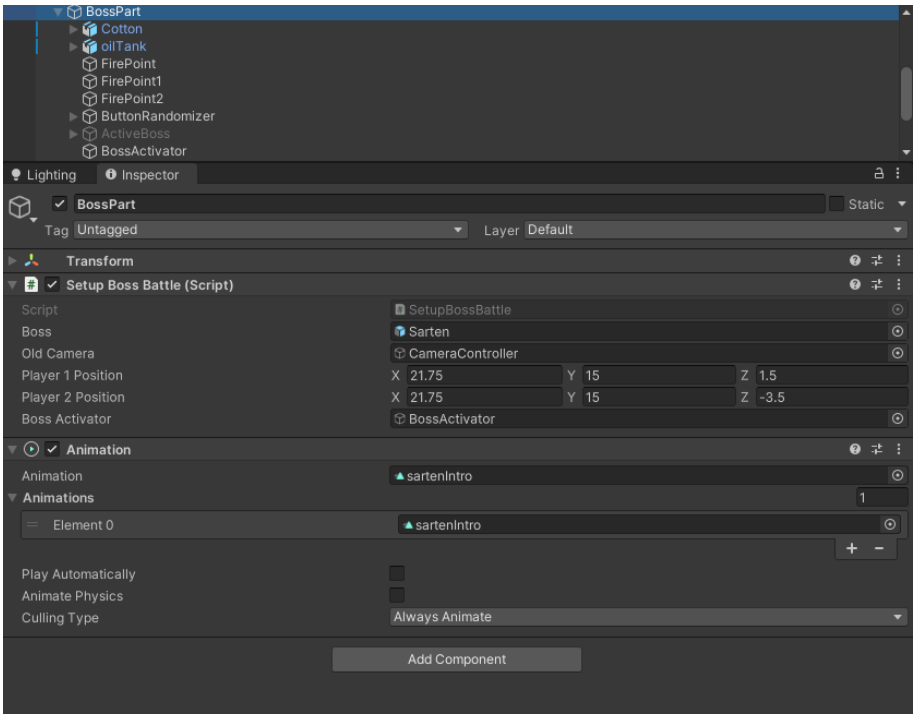


Figura 37 BossPart en el inspector

- **Characters:** Este objeto (Figura 38) se encarga de agrupar todos los personajes no jugables pasivos o los que muestran un diálogo cuando el jugador interacciona con ellos. No tiene ningún comportamiento extra más que la agrupación de los personajes.

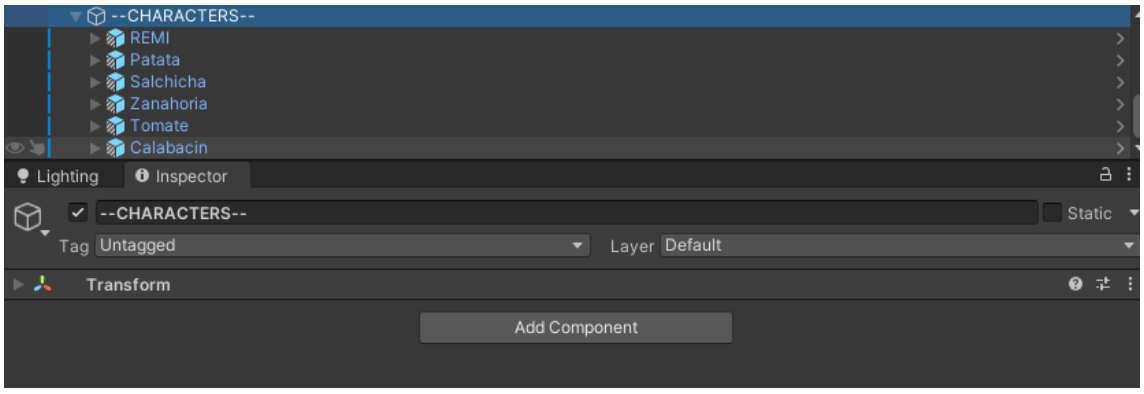


Figura 38 Characters en el inspector

5.2. Mecánicas básicas de los personajes

En la *Figura 39* podemos ver los componentes del objeto Clara y en la *Figura 40* los del objeto Yema. En estos componentes podemos ver algunos parámetros o propiedades configurables, como velocidad de movimiento, velocidad de rotación o de salto, y elementos del juego referidos por el jugador.

En ambos casos se utiliza el mismo controlador del jugador base, con la diferencia principal en las habilidades dependiendo del personaje. Estos componentes contribuyen a la resolución del requisito funcional P1, P2, P3, C1, C2 y Y1.

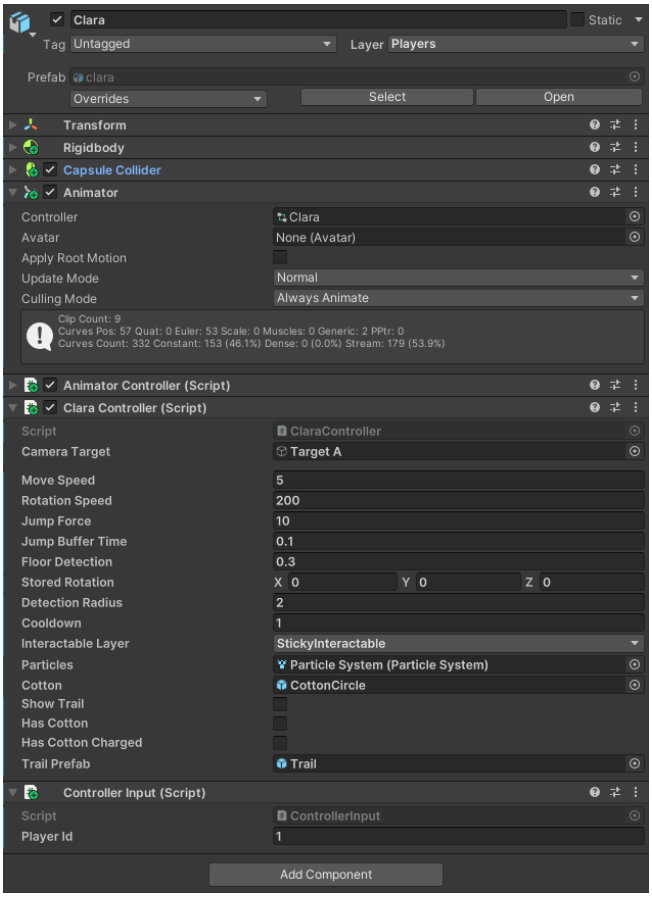


Figura 39 Captura del Prefab Clara

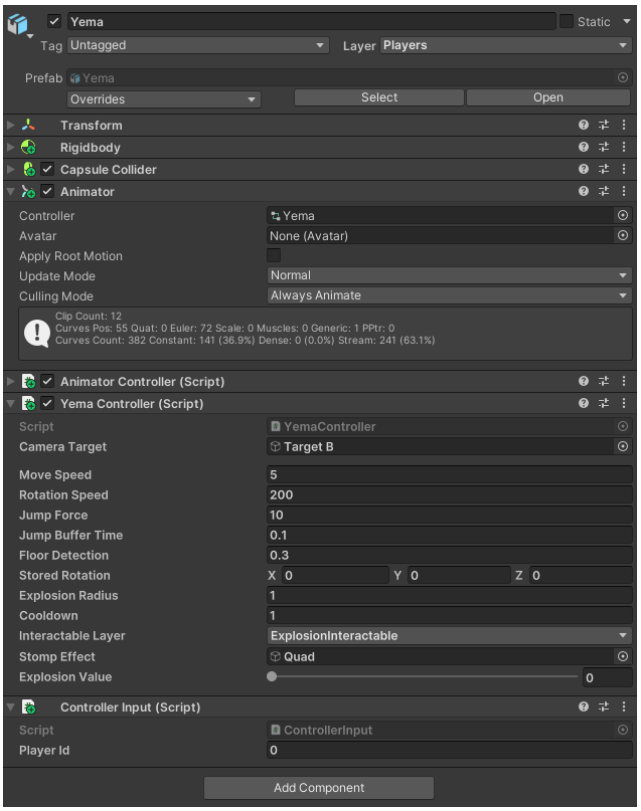


Figura 40 Captura del Prefab Yema

Las acciones que es capaz de realizar el jugador son:

- **Movimiento:** Los jugadores pueden moverse en el plano horizontal a una velocidad constante.
- **Salto:** Los jugadores pueden saltar siempre que estén en contacto con una superficie determinada como suelo. El salto siempre será a la misma velocidad.
- **Explosión (Yema):** Yema podrá interactuar con objetos si activa su habilidad especial estando en el aire.
- **Pegarse (Clara):** Clara puede adherirse a una superficie activando su habilidad especial, lo que le permitirá al jugador llegar a lugares antes inaccesibles.
- **Lanzar (Clara):** Clara puede lanzar un disco de algodón una vez haya sido empapado, lo que permitirá dañar a Sartén.

5.2.1. Diagramas de estado

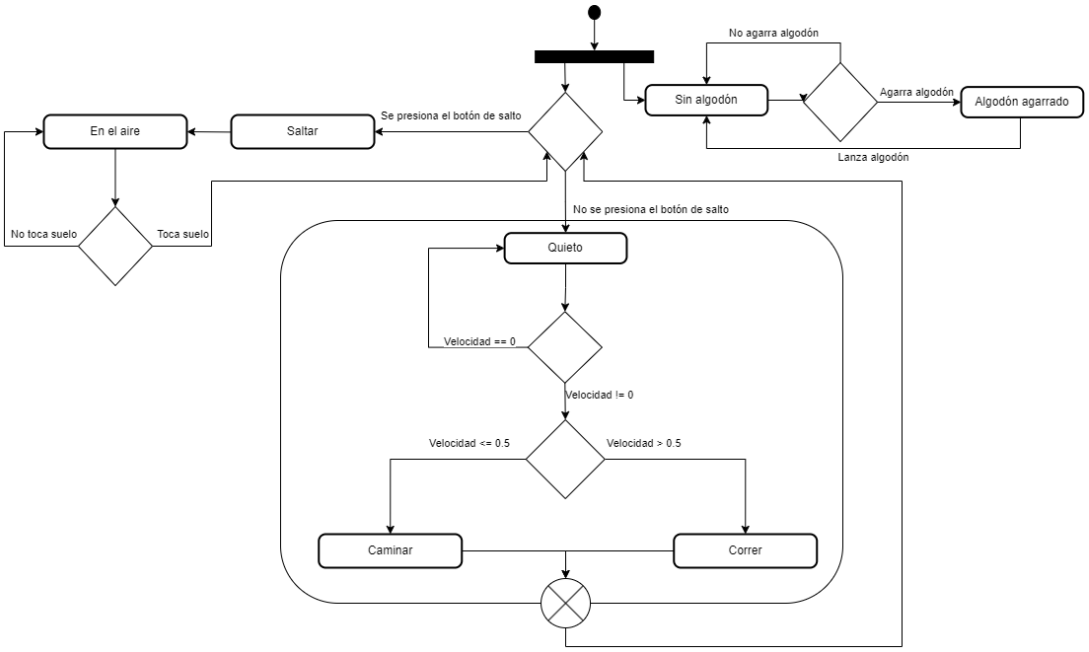


Figura 41 Diagrama de estados de Clara

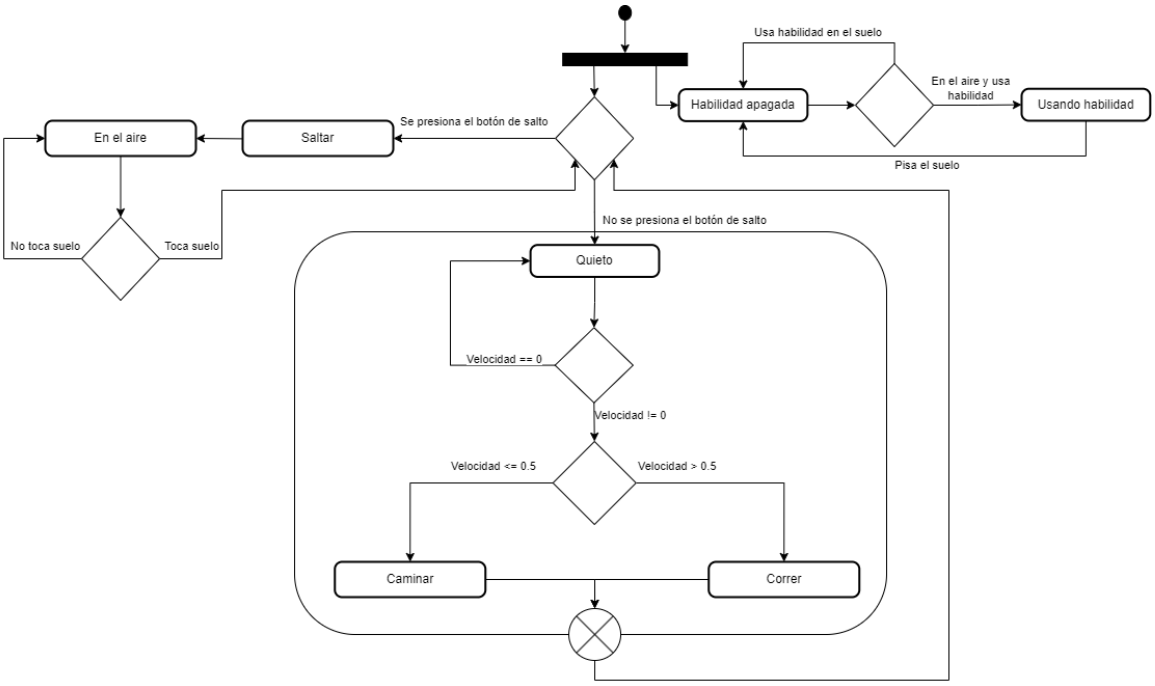


Figura 42 Diagrama de estados de Yema

Como se puede observar en los diagramas de estado de la Figura 41 y la Figura 42, ambos personajes

En Unity, existe un sistema de animaciones que permite crear pseudo diagramas de estado. Este sistema, mediante el uso de *Blend Trees* y de parámetros, permiten comunicarse con el código del juego para decidir cuándo hacer una transición al siguiente estado.

Los *Blend Trees* son unos estados especiales que permiten la mezcla de múltiples animaciones, realizando una transición suave entre ellas mediante un parámetro llamado *blending parameter*. (Blend Trees (Árboles de Mezcla), s.f.)

Los parámetros son variables que pueden modificarse mediante el código del juego, permitiendo el control de las animaciones y de las transiciones desde el código.

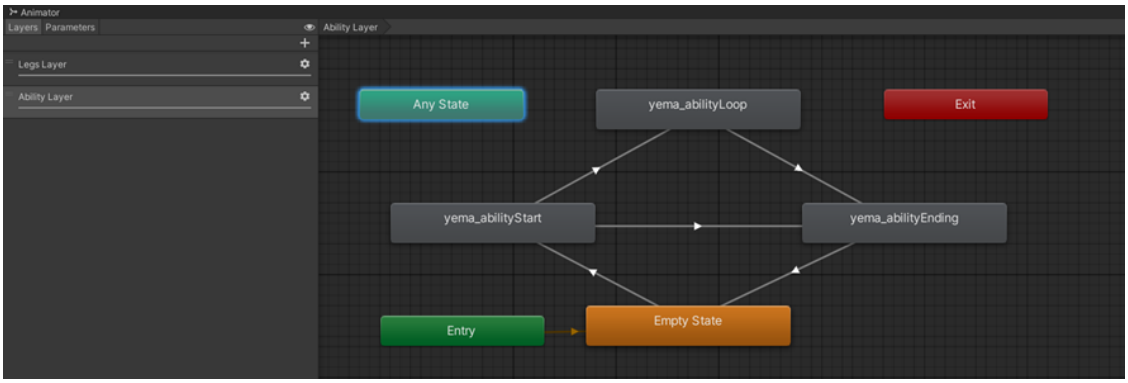


Figura 43 Ejemplo de diagrama de estado en Unity

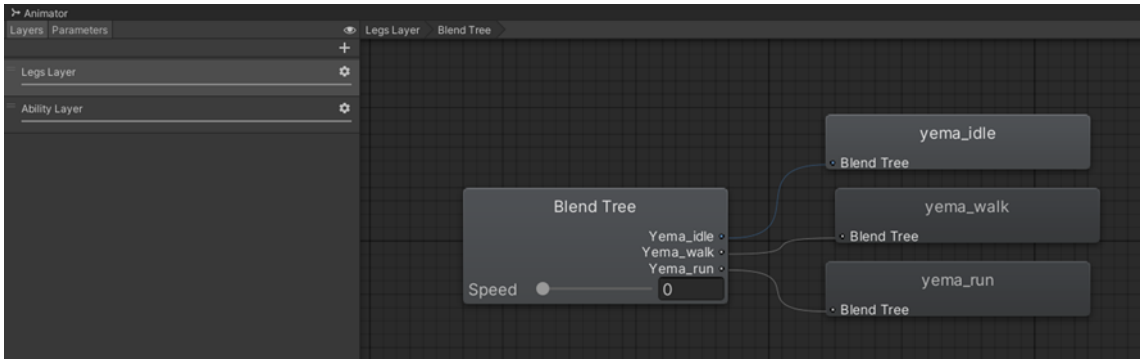


Figura 44 Ejemplo de Blend Tree del animador

Figura 45 Ejemplo de parámetros del animador

5.3. Los elementos interactivos

En la partida, hay repartidos por el escenario elementos con los que hay que interactuar para resolver puzles que permitirán acceder a otras zonas bloqueadas. A continuación, se muestra los elementos con los que se puede interactuar a lo largo de la partida y qué personaje será el encargado de interactuar con el mismo.

5.3.1. Ventilador

El ventilador (Figura 46) es un objeto que, al ser activado por Yema, genera una corriente de aire que moverá un grupo de servilletas, las cuales ayudarán a Clara a cruzar de la isla de la cocina a la encimera. Utilizando su habilidad de "explosión", Yema puede activar el ventilador cuando se encuentra en el aire. Este componente contribuye a la resolución del requisito funcional B1.

Figura 46 Captura del ventilador

5.3.2. Interruptor

El interruptor (Figura 47) es un mecanismo que, al ser accionado por Clara, activa la plancha y la hace bajar, permitiendo el paso de Yema. Este interruptor no necesita de una interacción del usuario, ya que se activa al colisionar con el mismo, detectando únicamente las colisiones con objetos de tipo **Players**. Este componente contribuye a la resolución del requisito funcional IN1.

Figura 47 Captura del interruptor

5.3.3. Maneta del grifo

La maneta del grifo (Figura 48) es un elemento accionado por Clara, que controla el flujo de agua de la pila. Clara puede girar la maneta, que activará una animación para detener el flujo del agua y permitir así el paso de Yema hacia el otro lado. Este componente contribuye a la resolución del requisito funcional E1.

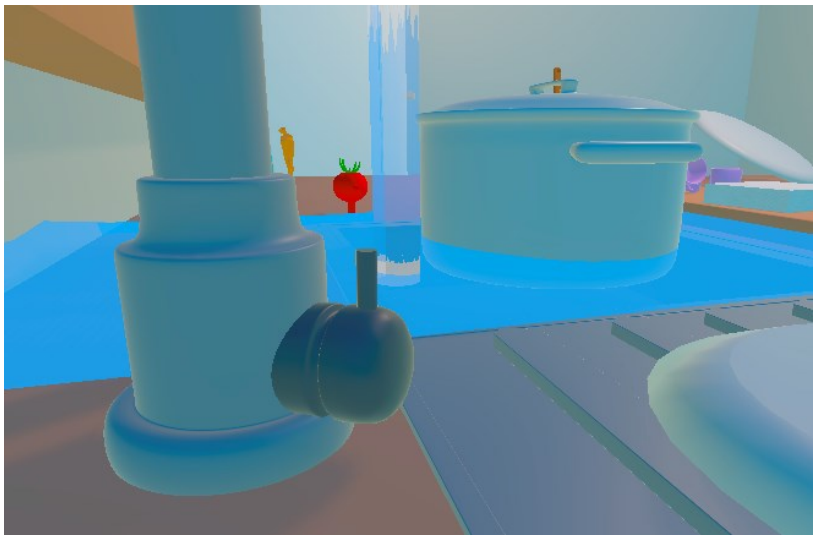


Figura 48 Captura de la maneta

5.3.4. Botones en la pelea final

Durante la batalla final contra el jefe, hay una serie de botones (Figura 49) accionables por Yema, asignados aleatoriamente a uno o más de los fogones que hay en la vitrocerámica. Yema puede interactuar con estos botones utilizando su habilidad de "explosión" mientras está en el aire, encendiendo o apagando los fogones que estén asignados al botón presionado. Este componente contribuye a la resolución del requisito funcional B1.

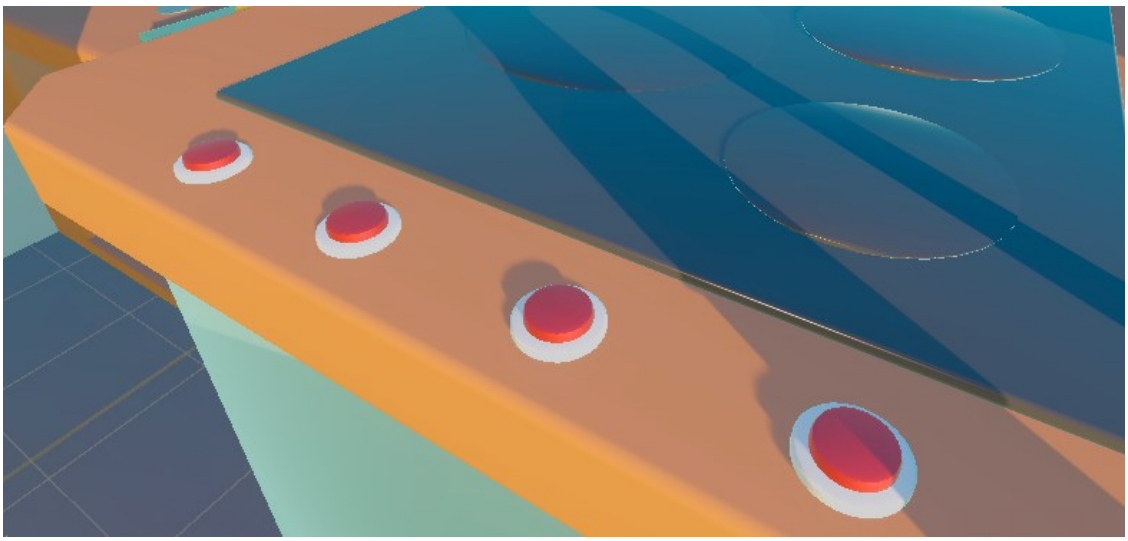


Figura 49 Captura de los botones de la pelea final

5.3.5. Algodón y aceite

Durante la batalla final contra el jefe, Clara deberá acercarse a un objeto (Figura 50) con el script **CottonInteractable**, interactuar con él para recoger una pieza de algodón y transportarla hacia un charco de aceite (Figura 51) con el script **OilInteractable**, el cual se usará para empapar el algodón y posteriormente lanzárselo a Sartén gracias a la habilidad especial de lanzamiento de Clara. Este componente contribuye a la resolución del requisito funcional E1.

Figura 50 Captura del algodón

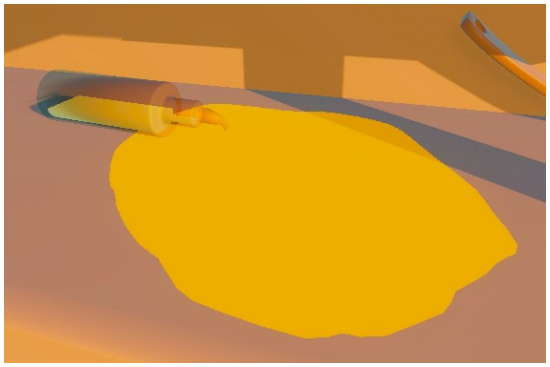


Figura 51 Captura del aceite

5.4. El sistema de diálogos

El sistema de diálogos se activará siempre que cualquiera de los dos jugadores colisione con un *trigger* que esté situado cerca de la ubicación de un personaje no jugable. Se mostrará una interfaz donde se podrá leer el dialogo correspondiente a cada personaje, como se puede ver en la Figura 26. Este componente contribuye a la resolución del requisito funcional J1 y N1.

5.5. El jefe final

El jefe final solo se activará cuando se detecte a los dos jugadores dentro del *trigger* que hay en la zona de la vitrocerámica. Al activarse comienza una cinemática y al acabar comienza la batalla. Este componente contribuye a la resolución del requisito funcional J2.

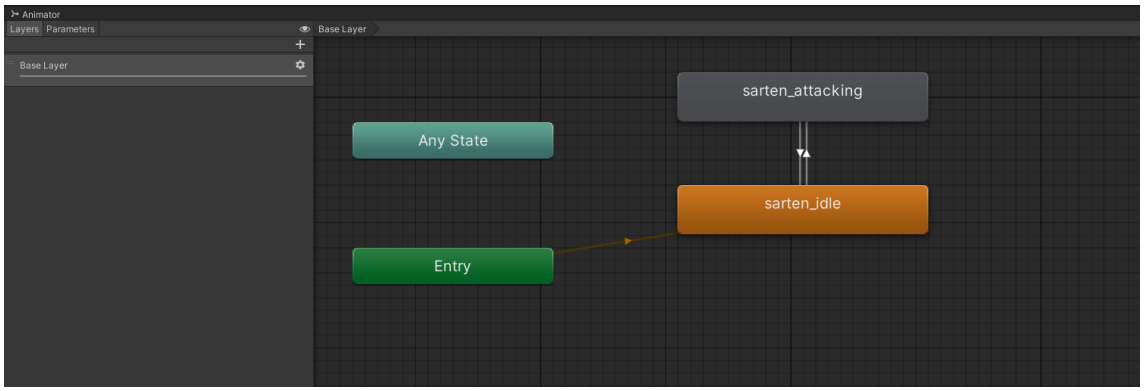


Figura 52 Animator de Sartén

Como podemos observar en la *Figura 52*, las animaciones de Sartén no son muy complejas, ya que la mayoría se ejecutan en el código de **SartenController** como podemos ver en la Figura 53, ya que para realizar el ataque de Sartén se necesita conocer la ubicación de Yema, lo cual hay que hacer mediante código.

```
void Update()
{
  if (!IsEnabled())
    return;

  if (SartenState.Attacking == _currentState && !yema.IsEnabled())
  {
    _currentState = SartenState.Idle;

    StopCoroutine(_attackCoroutine);
    _attackCoroutine = null;
    StartCoroutine(ResetEspatulas());
  }
  _animator.SetBool("Attacking", _currentState == SartenState.Attacking);

  if (_currentFirePoint.isOnFire)
  {
    MoveToFirePoint();
  }
  else if (fireParticles != null && fireParticles.isPlaying)
  {
    fireParticles.Stop();
    _audioSource.Stop();
  }

  if (attackCooldown <= 0 && _currentState == SartenState.Idle)
  {
    Attack();
    attackCooldown = Random.Range(3.5f, 6f);
  }
  else if (_currentState == SartenState.Idle)
```

```
    {
        attackCooldown -= Time.deltaTime;
    }
}
```

Figura 53 Fragmento de código de SartenController

Cuando el jefe está atacando, el Animador de Sartén se detiene pasando al estado a “sarten_attacking”. Este cambio de estado permite que Sartén ataque con las espátulas. Si no se detuviera el Animador, Sartén no podría atacar. Este componente contribuye a la resolución del requisito funcional S4.

Si el jefe está posicionado en un fogón en activo, se cancela el ataque que esté realizando en ese momento y Sartén se desplaza a otra posición donde no hay fuego, reiniciando las animaciones y volviendo al estado “sarten_idle”. Este componente contribuye a la resolución del requisito funcional S1, S2, S3.

5.6. La interfaz visual

Como hemos podido observar en la Figura 20, el menú principal tiene cuatro botones:

- **Historia**: Al seleccionar este botón, los jugadores serán redirigidos a la pantalla de selección de personaje (Figura 21), donde cada jugador elegirá qué personaje utilizar entre Clara y Yema. Este componente contribuye a la resolución del requisito funcional I2.
- **Opciones**: Al seleccionar este botón, los jugadores serán redirigidos al menú de opciones, donde podrán ajustar el volumen maestro, el de la música y el de los efectos de sonido. (Figura 22) Este componente contribuye a la resolución del requisito funcional I2.
- **Créditos**: Al seleccionar este botón, se mostrará la pantalla de créditos, la cual redirigirá al menú automáticamente a los cinco segundos. (Figura 23) Este componente contribuye a la resolución del requisito funcional I2 e I5.
- **Salir**: Al seleccionar este botón se cerrará la aplicación. Este componente contribuye a la resolución del requisito funcional I4.

Al seleccionar los dos personajes en la Figura 21, al pasar tres segundos, cargará la escena de la cocina. Este componente contribuye a la resolución del requisito funcional I1 e I3.

En la pelea contra el jefe final, se muestra un interfaz que indica su vida tal y como podemos ver en la Figura 25. Este componente contribuye a la resolución del requisito funcional S5.

6. Implantación

6.1. Puesta en marcha

Para la puesta en marcha del videojuego desarrollado, se debe crear una *build* desde la pestaña *Build Settings* del editor, como se puede observar en la Figura 54.

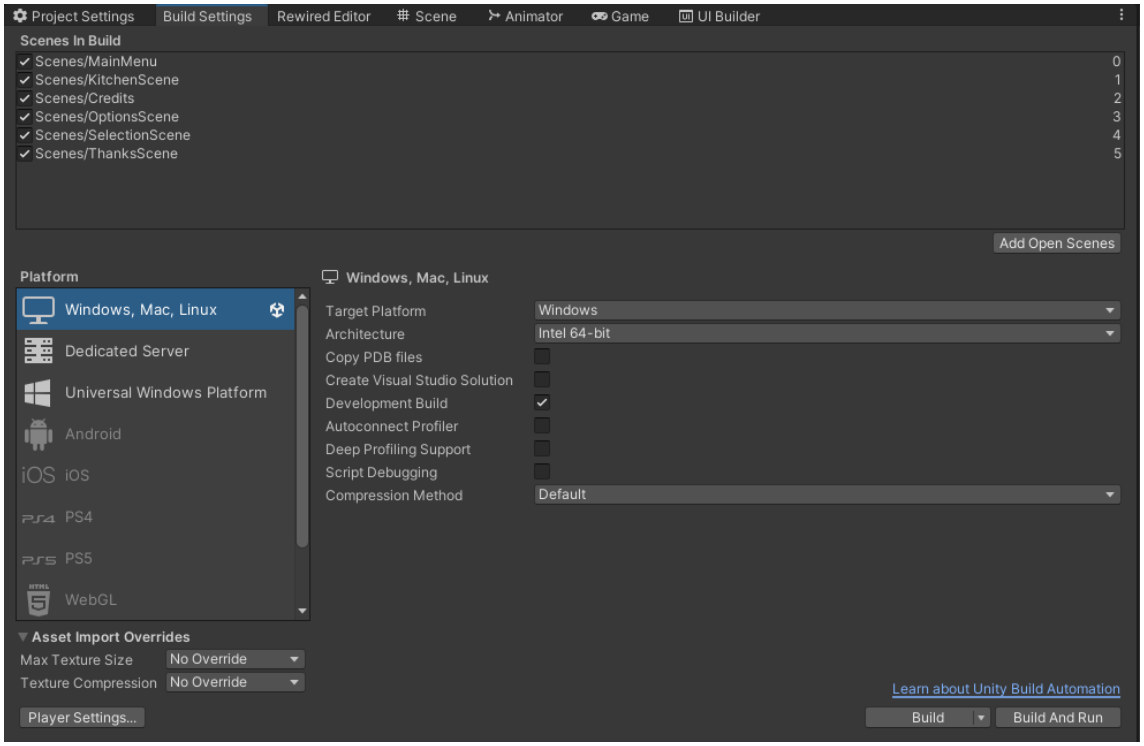


Figura 54 Captura de la pestaña Build Settings

Esta *build* genera un ejecutable del juego, que se puede ejecutar para empezar a jugar, sin necesidad de ningún programa externo.

Posteriormente, el archivo ejecutable y las demás carpetas (Figura 55) se exportan a una plataforma de videojuegos indie como itch.io (https://itch.io)

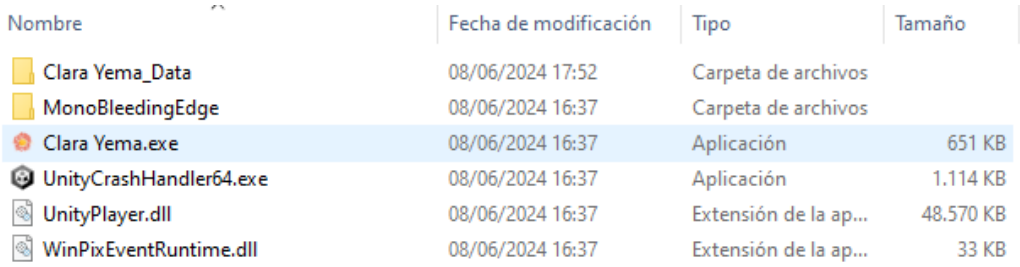


Figura 55 Captura de las carpetas del ejecutable

6.2. Pruebas de rendimiento

En este apartado se analizarán algunas pruebas realizadas manualmente por el desarrollador. Se ha ejecutado el juego para comprobar el rendimiento del juego en un sistema con un procesador I7-9700K y una tarjeta gráfica RTX 3080. Cuando el juego se ejecuta, en el procesador se observa un aumento de entre el 5-15% de carga, lo cual no supone un problema para el procesador. Sin embargo, la tarjeta gráfica tiene un mayor consumo (73%) ya que el juego debe renderizar una imagen con resolución 1080p cada 7 milisegundos.

Cuando se ejecuta el juego removiendo las limitaciones de la sincronización vertical, es decir, permitiendo el renderizado de más imágenes por segundo, el juego llega a renderizarse a aproximadamente 200 *frames* por segundo.

Figura 56 Captura del panel de sensores del PC

7. Pruebas

Tras finalizar el juego y subirlo a Itch.io, se puede dar por concluido el MVP1. En este apartado se analizarán los nuevos resultados de las mismas pruebas que se realizaron al inicio del proyecto, es decir, antes de la refactorización, para compararlas y evidenciar una mejora del proyecto.

Además, se realizará una evaluación de las encuestas hechas a los usuarios del juego.

7.1. Pruebas en Unity

En este apartado se presentarán y comentarán las pruebas unitarias y de regresión que se han aplicado antes y después de la refactorización. Estas pruebas se han hecho gracias a la herramienta *Test Runner* que está disponible en Unity y ayuda al desarrollador a ejecutar pruebas unitarias y ver el estado de estas.

7.1.1. Pruebas previas a la refactorización

Como se puede observar en la Figura 57, para el proyecto previo a la refactorización, se han realizado veinte pruebas, las cuales son ejecutadas correctamente.

Estas pruebas son importantes, ya que nos permiten comprobar si la refactorización del código se realizó correctamente, o sea, si el contenido implementado sigue funcionando igual en la nueva versión del juego.

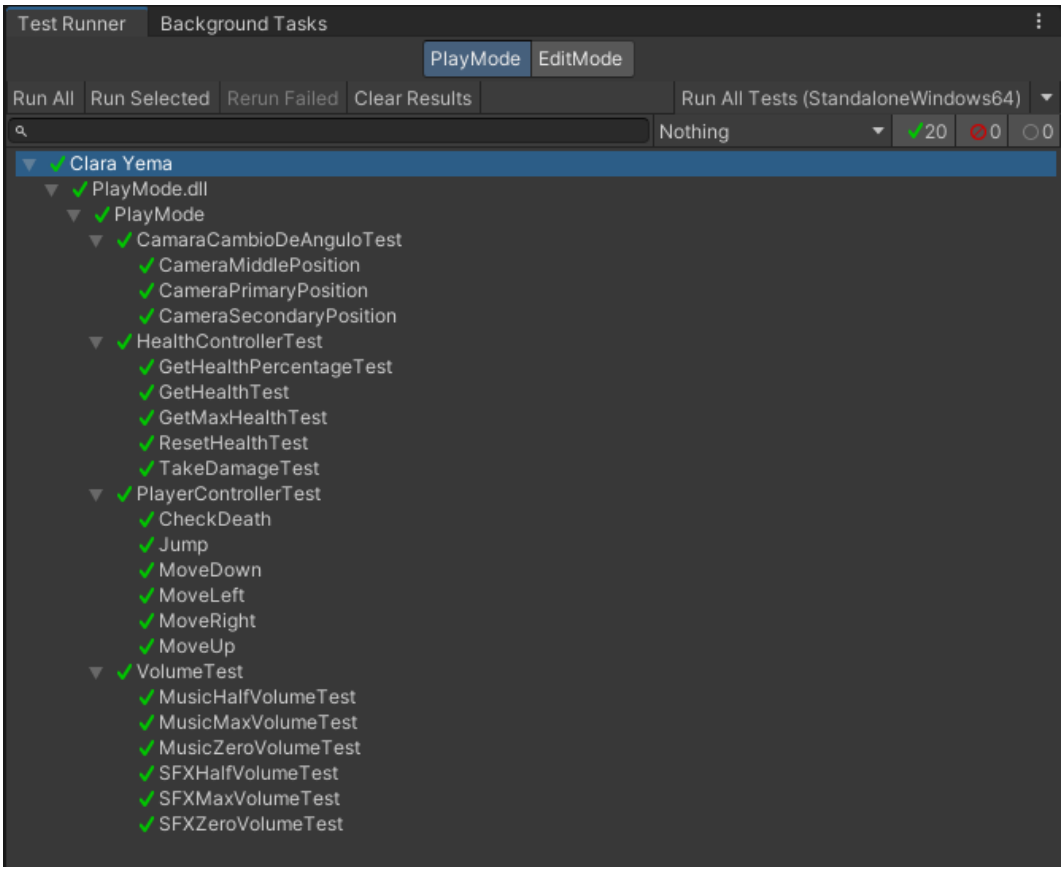


Figura 57 Captura de la herramienta de testeo Test Runner de Unity

7.1.2. Pruebas posteriores a la refactorización

Tras completar la refactorización y el desarrollo de nuevas características, se han ejecutado las pruebas de regresión (Figura 57), con adición de seis pruebas nuevas (Figura 58), que ayudan a comprobar que las características nuevas han sido aplicadas correctamente.

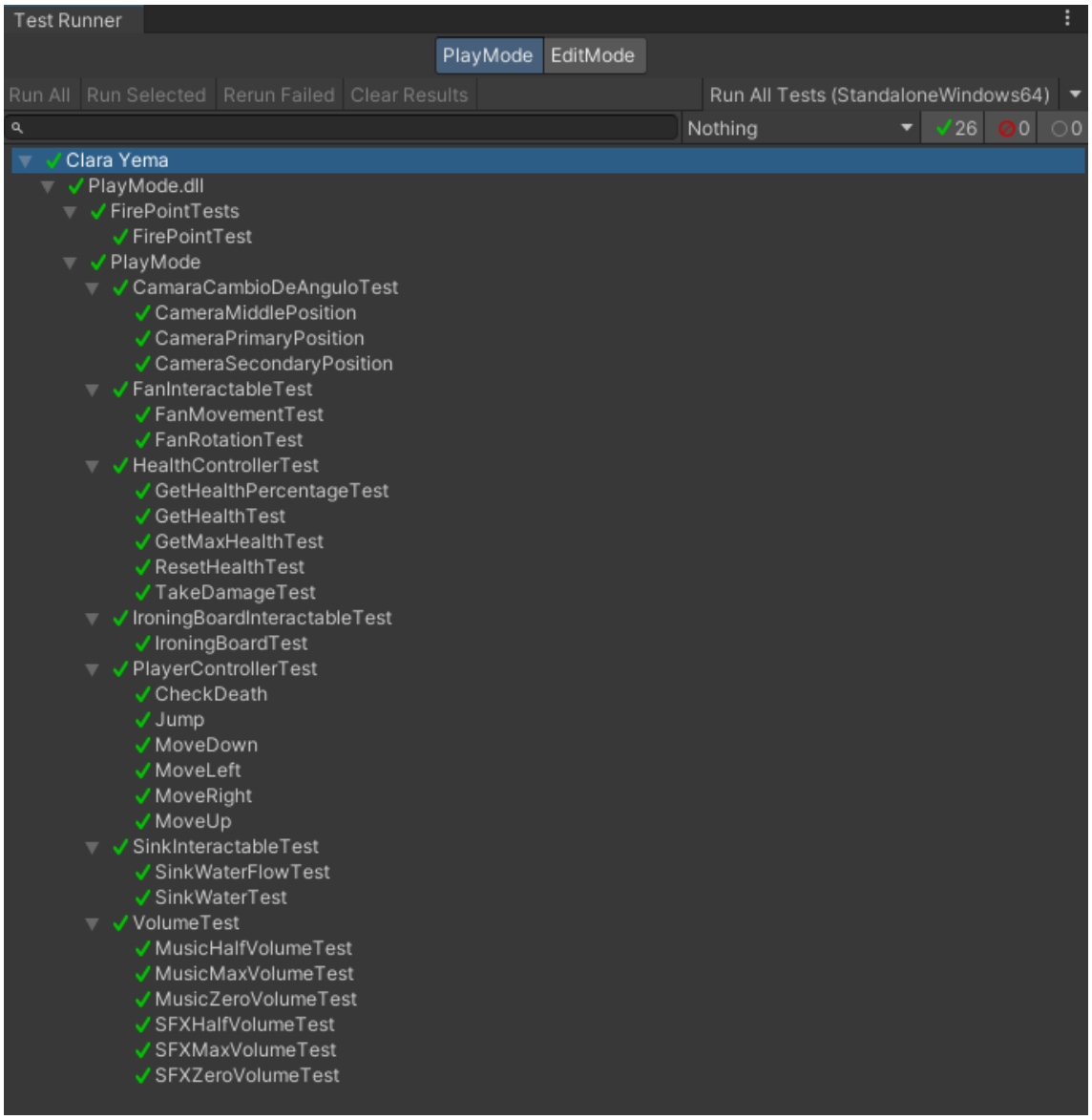


Figura 58 Captura de las pruebas unitarias y de regresión

7.2. Cálculo de las nuevas métricas y relaciones

7.2.1. Cálculo de las nuevas métricas

A continuación, se realizará un análisis del código fuente similar al realizado en el apartado 3.3.1:

El análisis del código fuente revela la siguiente comparativa:

	Datos actuales	Datos previos
Número de ficheros de código fuente	46	40
Cantidad total de líneas de código	2749	3650
Número medio de líneas de código por fichero	59,7	91,25
Promedio de las líneas de código de las funciones	8,871	13,106

Tabla 5 Análisis de líneas totales y medias posteriores a la refactorización

A continuación, se muestra una tabla similar a la Tabla 4:

Clase/Método	Líneas	C.C.
MusicVolume	39	
Start	6	1
Update	4	1
Mute	4	1
Unmute	4	1
IsMuted	4	1
SetVolume	5	1
GetVolume	4	1
SFXVolume	39	
Start	6	1
Update	4	1
Mute	4	1
Unmute	4	1
IsMuted	4	1
SetVolume	5	1
GetVolume	4	1
CamaraCambioDeAngulo	163	
Awake	4	1
StartTarget	30	4
Update	12	2
GetData	20	1
AddTarget	13	2
ReplaceTarget	6	1
GetTargets	4	1
GetCamera	5	1
OnDrawGizmos	35	1
CameraBoss	48	
Update	8	2
AddTarget	6	2
StartTarget	11	3
GetCamera	4	1
PlayVideo	6	1
CameraController	10	
CameraFadeController	74	
Start	8	1
Update	17	3
FadeIn	6	2
FadeOut	6	2
FadeInCoroutine	8	2
FadeOutCoroutine	8	2
AddCamera	4	1
ScriptableHealthContainer	54	
SetHealth	4	1
SetMaxHealth	4	1
GetHealth	4	1
GetMaxHealth	4	1
Heal	8	2
ResetHealth	4	1
Damage	8	2
Kill	4	1
GetHealthPercentage	4	1
FirePoint	33	
SetOnFire	6	1

Clase/Método	Líneas	C.C.
SetOffFire	6	1
ToggleFire	11	2
SartenButtonRandomizer	42	
Start	4	1
RandomizeFirePoints	29	6
SartenController	249	
Start	21	5
Update	31	10
Attack	9	4
AttackCoroutine	10	1
SendEspatulas	29	5
ResetEspatulas	9	1
ResetEspatulaPosition	14	3
ResetEspatulaRotation	14	3
MoveToFirePoint	31	8
MoveToPosition	4	1
MoveToPositionCoroutine	18	2
RemoveHealth	21	11
OnDrawGizmos	10	1
NPCController	6	
ClaraController	224	
Awake	12	3
SetEnabled	4	1
Update	14	3
LateUpdate	7	2
GetLaunch	19	7
AdjustCamera	5	2
AdjustCameraCoroutine	15	2
ResetCamera	5	2
ResetCameraCoroutine	15	2
LaunchCotton	5	1
ChargeCotton	20	5
CalculateCottonTrajectory	33	7
UseAbility	27	5
StickToSurface	9	1
SetMovementEnabled	5	1
OnDrawGizmos	7	1
CottonController	36	
Awake	4	1
SetOilMaterial	4	1
SetCottonMaterial	4	1
OnTriggerEnter	13	4
YemaController	61	
Awake	5	1
Update	9	2
UseAbility	8	3
WaitUntilGrounded	20	3
OnDrawGizmos	6	1
AnimatorController	41	
Start	5	1
Update	27	6
PlayerController	190	
Awake	5	1

Update	12	2
LateUpdate	4	1
CheckCamera	13	5
SetInput	4	1
GetInput	4	1
HandleInput	9	2
CheckGrounded	7	3
ProcessInput	34	10
GetAbility	8	2
GetInteraction	14	3
CheckDeath	17	4
Revive	13	3
GetCharacterID	4	1
SetCharacterID	4	1
OnDrawGizmos	8	1
CharacterController	19	
DisableCharacter	4	1
EnableCharacter	4	1
IsEnabled	4	1
EventTrigger	85	
InvokeMethod	8	2
OnTriggerEnter	16	5
OnTriggerExit	11	2
OnTriggerStay	7	2
OnCollisionEnter	7	2
OnCollisionExit	7	2
OnCollisionStay	7	2
SetupBossBattle	59	
Start	4	1
StartBossBattle	5	1
PositionPlayers	20	5
StartBoss	8	1
FadeIn	5	3
FadeOut	5	3
CharacterInput	93	
EnableInput	4	1
DisableInput	4	1
GetHorizontalInput	6	2
GetHorizontalLookInput	6	2
GetVerticalInput	6	2
GetVerticalLookInput	6	2
GetJump	6	2
GetAbility	6	2
GetAnyInput	6	2
GetAnyInputDown	6	2
GetAnyInputNoPause	4	1
GetAnyInputDownNoPause	4	1
GetInteract	4	1
GetLaunch	4	1
ControllerInput	54	
setPlayerID	5	1
getHorizontalInput	4	1
getHorizontalLookInput	4	1
getVerticalInput	4	1
getVerticalLookInput	4	1
getJump	4	1
getAbility	4	1
getAnyInput	4	1
getAnyInputDown	4	1
getInteract	4	1

getLaunch	4	1
CottonInteractable	10	
Interact	6	2
ExplosionInteractable	10	
Interact	4	1
FanInteractable	49	
Awake	5	1
Update	9	2
Interact	10	2
DisableButton	9	1
Interactable	9	
Interact	3	1
IroningBoardInteractable	21	
OnCollisionEnter	7	2
Interact	5	1
OilInteractable	15	
Interact	9	2
SartenButton	17	
Interact	8	1
SinkInteractable	13	
Interact	4	1
StickyInteractable	10	
Interact	4	1
GameManager	49	
Awake	12	2
Start	4	1
GetBGMusic	4	1
SetControllers	5	1
EndGame	4	1
GameSettings	58	
getSFXVolume	5	1
getMusicVolume	5	1
LoadGameSettings	6	1
SetProperty	18	4
GetProperty	15	4
PlatformController	25	
Update	9	2
OnTriggerStay	4	1
OnTriggerExit	3	1
RotatePlatform	16	
Start	4	1
Update	4	1
SceneController	99	
Awake	31	1
RegisterCharacter	4	1
foreach	7	2
foreach	7	2
GetActiveCameraController	11	2
LoadMenu	4	1
GetCameraFadeController	4	1
BossUI	19	
Start	5	1
Update	4	1
CreditsController	17	
Start	4	1
BackToMainMenu	4	1
DialogueManager	169	
Start	12	1
ActivateDialogue	4	1
WaitToActivateDialogue	11	3

AreAllPlayersReady	4	1
Update	31	3
UpdateDialogue	14	4
CheckDialogueInput	25	5
DisplayDialogue	4	1
DisableDialogue	19	3
SetText	5	1
MainMenuController	46	
Start	18	1
StartGame	4	1
OpenOptions	4	1
OpenCredits	5	1
ExitGame	5	2
SelectorController	159	
Start	12	1
Update	7	1
UpdatePlayersMovement	16	4
UpdatePlayerTimer	10	2
UpdateControllersAlignment	5	1
AlignController	10	1
CheckAndStartGame	12	3
AnyControllerInCenter	4	2
BothControllersNotInCenter	4	2
StopGameCoroutine	8	2
StartGameCoroutine	7	2
StartGame	15	3
MoveController	23	10
BackToMainMenu	4	1
SliderController	69	
Start	12	1
SetupSlider	21	2

BackToMainMenu	4	1
OnMasterSliderValueChanged	4	1
OnSFXSliderValueChanged	4	1
OnMusicSliderValueChanged	4	1
UpdateSliderValue	6	1
TextFit	26	
Start	6	1
Update	8	2
ThanksUI	29	
Start	12	1
BackToMainMenu	4	1
OpenForm	4	1
UIController	114	
Start	9	2
Update	26	5
MoveY	8	3
MoveX	8	3
PaintButtons	13	4
PaintSlider	19	4
ChangeFocus	9	3
TryChangeValue	9	2
CinematicPlayer	59	
Start	4	1
EndReached	16	6
Update	11	2
PlayVideo	15	7
VideoRunner	22	
Awake	7	2
PlayVideo	4	1

Tabla 6 Lista de clases y métodos posteriores a la refactorización

Como podemos observar tanto en la Tabla 3 como en la Tabla 5, hay seis clases más que en el programa original, aunque este número no es representativo, ya que la estructura del programa y la mayoría de las clases han cambiado por completo.

Los datos más representativos son el número de líneas, reducido de 3650 a 2749, el promedio de líneas de código por fichero, que ha bajado de 91,25 a 59,7 y el promedio de líneas por funciones, que ha decrecido significativamente, de 13,106 a 8,871.

7.2.2. Cálculo del número de relaciones

A partir de la *Figura 39 Captura del Prefab Clara*Figura 27, se contarán el número de relaciones que tiene cada clase con otras:

- MusicVolume: 1
- SFXVolume: 0
- CamaraCambioDeAngulo: 1
- CameraBoss: 1
- CameraController: 2
- CameraFadeController: 1
- ScriptableHealthContainer: 2
- FirePoint: 2
- SartenButtonRandomizer: 2
- SartenController: 2
- NPCController: 4
- ClaraController: 1

- CottonController: 1
- YemaController: 1
- AnimatorController: 1
- PlayerController: 5
- CharacterController: 3
- EventTrigger: 0
- SetupBossBattle: 0
- CharacterInput: 3
- ControllerInput: 1
- CottonInteractable: 1
- ExplosionInteractable: 3
- FanInteractable: 1
- Interactable: 4
- IroningBoardInteractable: 1
- OilInteractable: 1
- SartenButton: 1
- SinkInteractable: 1
- StickyInteractable: 3
- GameManager: 2
- GameSettings: 0
- PlatformController: 0
- RotatePlatform: 0
- SceneController: 5
- BossUI: 1
- CreditsController: 0
- DialogueManager: 2
- MainMenuController: 1
- SelectorController: 1
- SliderController: 1
- TextFit: 1
- ThanksUI: 1
- UIController: 4
- CinematicPlayer: 1
- VideoRunner: 2

Como podemos ver, el número de relaciones entre clases se ha reducido. Esto refleja que el código actual es mucho más escalable, ya que las dependencias entre clases ahora son menores. Un ejemplo muy notorio es la clase PlayerController, que ha pasado de tener 14 clases relacionadas a tener solo 5.

7.3. Encuestas con usuarios

A continuación, se evaluarán las encuestas con los usuarios para evaluar la satisfacción del usuario con el juego Clara Yema. Para ello se han realizado varias preguntas, tanto antes de la refactorización y desarrollo como después:

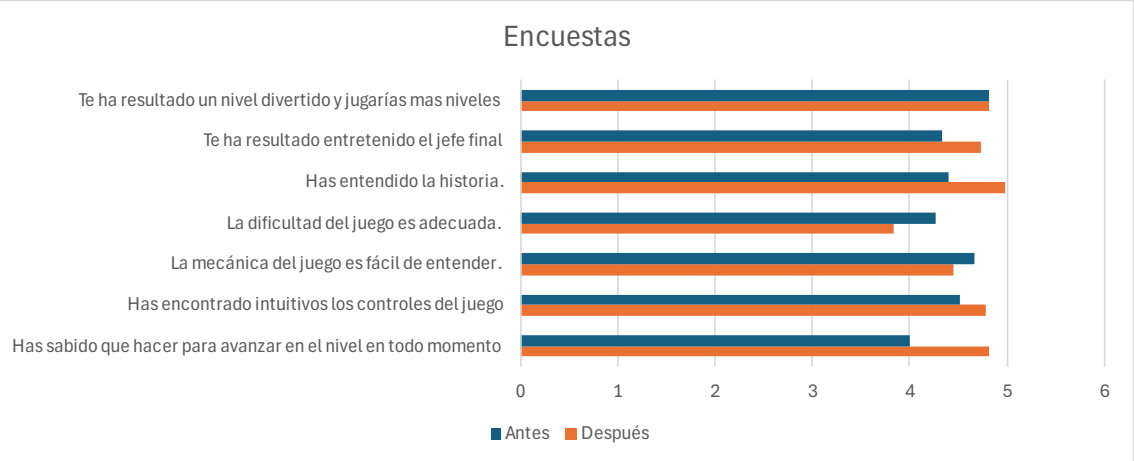


Figura 59 Resultado encuestas

Como se observa en la gráfica donde se compara los resultados de los encuestados antes y después de la refactorización (Figura 59), se ve cómo los resultados, en general, mejoran. Al ser las medias muy positivas antes de la refactorización, no todos los resultados han mejorado. Este es el caso de la pregunta relacionada con la mecánica del juego, donde la disminución puede ser debida al aumento de puzles y complejidad del juego, pues el escenario tiene más variables.

En cambio, cabe destacar dos aumentos destacables. Por una parte, se encuentra el entendimiento de la historia. Los usuarios han podido entender mejor la historia, ya que los diálogos han sido perfeccionados. Y por otra parte, cabe destacar, la capacidad de los usuarios para saber cómo avanzar en el juego. Esto es debido a que los personajes van ayudando al usuario a avanzar.

8. Conclusiones

Con respecto a los objetivos marcados en el apartado 1.2, podemos afirmar que se han completado casi todos los objetivos, a excepción del cuarto objetivo.

Se ha utilizado la metodología ágil Kanban. Además, se han utilizado Trello y GitHub para el control de versiones, lo que ha permitido tener tanto una copia de seguridad como un seguimiento de los cambios y mejoras al proyecto.

El primer objetivo era la optimización del código y la mejora del rendimiento. Como podemos ver en el apartado 7.2, se ha cumplido satisfactoriamente, ya que el código ha sido optimizado, se han reducido gran cantidad de errores y se ha creado una estructura del proyecto mucho más escalable.

En cuanto al segundo objetivo, mejora de la jugabilidad, se han introducido con éxito nuevas mecánicas, habilidades y puzles. Además, se ha rehecho por completo la pelea contra el jefe final, lo que mejora por completo la experiencia de juego.

También se han aplicado pruebas de regresión, unitarias y con el usuario, lo que satisface el tercer objetivo al implementar un sistema de control de calidad.

El cuarto objetivo era desarrollar al menos dos versiones jugables en las cuales los usuarios dieran sus opiniones de mejora. Este objetivo se ha cumplido parcialmente, ya que solo se ha realizado una versión en la cual se ha hecho una encuesta a los usuarios.

Para concluir, es importante subrayar el valioso aprendizaje que ha representado la experiencia de desarrollar un videojuego como proyecto final. Este proceso ha requerido la familiarización con tecnologías nuevas para la parte técnica y además, ha implicado la realización de numerosas pruebas. A pesar de contar con experiencia previa en asignaturas como Mantenimiento y Evolución de Software, ha sido la primera vez que se han aplicado estos conocimientos en un entorno real, lo que ha supuesto un reto significativo y una oportunidad invaluable de crecimiento profesional. Asimismo, este proyecto abre nuevos caminos al desarrollo de este videojuego.

9. Trabajos futuros

Este proyecto ha concluido de forma satisfactoria, pues se dispone de un videojuego completamente funcional. Es interesante conocer las futuras líneas de trabajo que se pueden llevar a cabo para continuar el juego.

Para empezar, se comentará la principal y más interesante línea de futuro. Se trata de ampliar el juego a muchos más niveles o pantallas. Para ello, se necesitaría tiempo para desarrollar las nuevas partes y además se necesitaría la colaboración de una o dos personas mínimo. Este número es tan elevado debido a la necesidad de personas con conocimientos de software y con habilidades en diseño de personajes, escenarios, etc.

Por otra parte habría que mejorar el *game feel*. El *game feel* consiste en mejorar la calidad y la sensación del jugador al jugar el videojuego. Ejemplos de *game feel* serían añadir elementos como efectos de partículas o una retroalimentación auditiva placentera.

Finalmente, se plantea añadir compatibilidad con el teclado y la posibilidad de que dos jugadores puedan jugar desde diferentes ordenadores a través de una conexión a internet.

10. Referencias

Adams, E. W. (2013). *Fundamentals of Game Design.* New Riders.

Blanes, R. G. (2018). *El Libro Práctico del Programador Ágil.* Rafa G. Blanes.

Blend Trees (Árboles de Mezcla). (s.f.). Obtenido de Unity Documentation: https://docs.unity3d.com/es/530/Manual/class-BlendTree.html

Carabaña, C. (27 de 3 de 2020). *Doom, el videojuego que creó el mundo gamer y que sigue siendo un éxito casi 3 décadas después*. Obtenido de Revista GQ: https://www.revistagq.com/noticias/articulo/doom-videojuego-historia

Chandler, H. (2020). *The Game Production Toolbox.* CRC Press.

del Dedo, R. d., García, A. Á., & Gómez, C. L. (2018). *Métodos Ágiles. Scrum, Kanban, Lean.* ANAYA MULTIMEDIA. Obtenido de Right People Group: https://rightpeoplegroup.com/es/blog/los-7-tipos-de-metodologias-agiles

Fernández, E. C. (14 de 08 de 2020). *Space Invaders: un videojuego de marcianitos con mucha historia*. Obtenido de Tokio School: https://www.tokioschool.com/noticias/space-invaders-videojuegos-marcianitos-historia/

Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code.* Pearson Education.

Gallego. (13 de 6 de 2008). *Especial: repasamos la saga 'Alone in the Dark'*. Obtenido de Vida Extra: https://www.vidaextra.com/pc/especial-repasamos-la-saga-alone-in-the-dark

Historia de los videojuegos. (s.f.). Obtenido de Universidad Politécnica de Cataluña: https://www.fib.upc.edu/retro-informatica/historia/videojocs.html

Jones, D. T., & Womack, J. P. (14 de 02 de 2012). *Lean Thinking.* Grupo Planeta. Obtenido de APD: https://www.apd.es/metodologia-lean-que-es/#:~:text=El%20Lean%20es%20un%20m%C3%A9todo,el%20tiempo%20y%20el%20esfuerzo

Keith, C. (2010). *Agile Game Development with SCRUM.* Addison-Wesley Professional.

Linares, M. (8 de 2 de 2019). *Videojuegos de coches retro: cuando ir a toda velocidad solo te costaba cinco duros*. Obtenido de El Economista: https://www.eleconomista.es/ecomotor/motor/noticias/9687679/02/19/Videojuegos-de-coches-retro-cuando-ir-a-toda-velocidad-solo-costaba-cinco-duros.html

McCarthy, R. (2020). *El Método Agile.* Obtenido de BBVA: https://www.bbva.com/es/innovacion/metodologia-agile-la-revolucion-las-formas-trabajo/

Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación. (s.f.). Obtenido de Naciones Unidas: https://www.un.org/sustainabledevelopment/es/infrastructure/

Perazo, C. (10 de 6 de 2024). *La historia del Pac-Man, el videojuego japonés que conquistó al mundo*. Obtenido de La Nación: https://www.lanacion.com.ar/juegos/historias/la-historia-del-pac-man-el-videojuego-japones-que-conquisto-al-mundo-entero-nid10062024/

Radice, P. (s.f.). *Spacewar!* Obtenido de Proyecto IDIS: https://proyectoidis.org/spacewar/

Ruiz, E. (14 de 4 de 2023). *Descubre la increíble historia de Pong, el primer videojuego rentable de la historia*. Obtenido de La Razón: https://www.larazon.es/tecnologia/descubre-increible-historia-pong-primer-videojuego-rentable-historia_202304146439379f1b5f5b000148d9f3.html

Trilnick, C. (s.f.). *Tennis for Two*. Obtenido de Proyecto IDIS: https://proyectoidis.org/tennis-for-two/

Velasco, J. (15 de 7 de 2011). *Historia de la tecnología: OXO, un videojuego para uno de los primeros computadores de la historia*. Obtenido de Hipertextual: https://hipertextual.com/2011/07/oxo-un-videojuego-para-uno-de-los-primeros-computadores-de-la-historia

Villar, E. (31 de 8 de 2016). *Aquel «loco» que llevó el videojuego a la TV*. Obtenido de La Razón: https://www.larazon.es/tecnologia/aquel-loco-que-llevo-el-videojuego-a-la-tv-JE13429356/

11. Anexos

Anexo 1: OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

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

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

El 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años.

1. Fin de la pobreza

2. Hambre Cero
3. Salud y Bienestar
4. Educación de Calidad
5. Igualdad de género
6. Agua limpia y saneamiento
7. Energía asequible y no contaminante
8. Trabajo decente y crecimiento económico
9. Industria innovación e infraestructura
10. Reducción de las desigualdades
11. Ciudades y comunidades sostenibles
12. Producción y consumos responsables
13. Acción por el clima
14. Vida submarina
15. Vida de ecosistemas terrestres
16. Paz, justicia e instituciones
17. Alianzas para lograr objetivos

De los anteriores objetivos de desarrollo sostenibles mencionados, el proyecto está relacionado con:

Relación con el ODS 8: Trabajo Decente y Crecimiento Económico

El desarrollo del videojuego "Clara Yema" y su mejora continua a través de técnicas de refactorización y mantenimiento, no solo representa una práctica significativa en el ámbito educativo y profesional, sino que también contribuye directamente al ODS 8, que busca promover el trabajo decente y el crecimiento económico. Al centrarse en mejorar la calidad del código y añadir funcionalidades nuevas, se fomenta la adquisición de habilidades avanzadas en programación y desarrollo de software, lo cual es esencial en el mercado laboral actual.

Relación con el ODS 9: Industria, Innovación e Infraestructura

La refactorización y expansión del videojuego "Clara Yema" está alineada con el ODS 9, que busca fomentar la innovación. En este contexto, el proceso de mejorar la estructura del código y de añadir nuevas funcionalidades a un software existente no solo mejora la eficiencia y la calidad del producto, sino que también promueve la innovación en el ámbito de los videojuegos y la tecnología digital.

Anexo 2: GDD

2023-2024

GAME DESIGN DOCUMENT

creado por

Alejandro Navarro
Clara García
Adrián Soriano
Ana Paula Moreno
Marta Colomer
Celia Veiga

última fecha de modificación

30/11/2023

TABLA DE CONTENIDOS

Análisis del videojuego

Clara y Yema es un juego multijugador en el que una clara y una yema se enfrentan a diferentes puzzles y combates para lograr resolver un dilema existencial. Con una temática de aventura y comedia, este juego pertenece a los géneros de plataformas y puzzles.

Cada jugador escogerá un personaje y deberán colaborar entre ellos para sortear los distintos obstáculos a través de sus habilidades. También deberán vencer a un boss final a través de diferentes técnicas al terminar cada nivel, lo que dará pie al siguiente.

El juego que trata siempre un tono divertido, con toques de humor que hacen la experiencia del jugador más divertida. Sin embargo, en ciertos puntos se vuelve algo anticlimático con diálogos un tanto oscuros, tapados siempre con una estética alegre y adorable.

Género

El juego pertenece al género de plataformas y puzzles ya que para avanzar en la aventura tienes que superar obstáculos a base de saltar, dashear, meterte por huecos pequeños y resolver puzzles aprovechando las diversas habilidades de los personajes.

Plataformas

La plataforma a las que está orientado el videojuego es la PlayStation 4 y la 5. Creemos que en cuanto a rendimiento y tipo de videojuego será la mas adecuada, pues nos dará libertad de desarrollo y comodidad a la hora de testeo.

Público objetivo

El público objetivo sería la gente joven a partir de 13 años, ya que a esa edad se puede analizar mejor el humor del videojuego. Aunque el estilo y el diseño tenga un aspecto infantil, consideramos no más de un PEGI 7 debido a la violencia no realista y posibles sonidos o escenas que podrían asustar a los más pequeños.

Referentes y análisis competitivo

Para la elaboración e ideación del videojuego Clara y Yema se han tenido en cuenta varios referentes. Para el apartado estético veremos influencias del videojuego OlliOlli World, un videojuego en 3D que a su vez utiliza los toon shaders y el outline en sus modelos, para dar así una estética más cartoon y amigable al videojuego. De hecho, también nos hemos basado en el estilo de sus personajes para el diseño de los de Clara y Yema. También podríamos hablar estéticamente como referente de la serie Hora de Aventuras, con sus personajes generalmente divertidos y sencillos pero, sin embargo, muy carismáticos.

Por otro lado, en cuestión a mecánicas y tipo de videojuego, sin duda hay una clara influencia del It Takes Two y la saga de Super Mario Bros, pues se trata de un videojuego cooperativo multijugador en el que se deben avanzar los niveles para profundizar en la historia.

Estos videojuegos mencionados son nuestros referentes, mas a su vez son nuestros competidores. Nuestro videojuego se diferenciará de ellos en la historia que, con un tono humorístico y unos personajes especialmente carismáticos, aportará al jugador una experiencia más divertida.

Captura del videojuego It Takes Two

Personajes "chuches" de la serie Hora de Aventuras

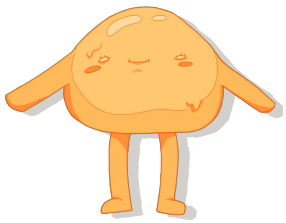
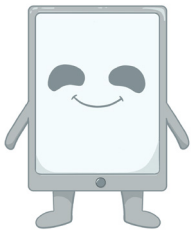
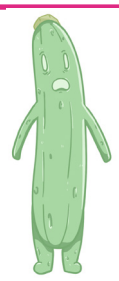
Captura del videojuego OlliOlliWorld

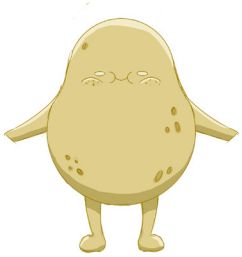
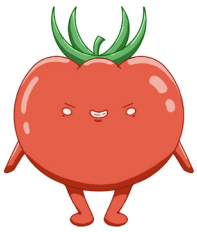
Trama y personajes

Clara y Yema están a punto de ser cocinadas cuando entran en un conflicto existencial: ¿quiénes son? ¿de dónde vienen? ¿Quién es su creador? de modo que inician una aventura con intención de resolverlo. Irán encontrándose con algunos de los compañeros de la cocina, quienes están también a la espera de ser cocinados o algunos incluso ya son alimento para humanos. Sin embargo, todos conocen a la madre de nuestras protagonistas, a quien buscarán para entender el motivo de su existencia.

Sin embargo, los utensilios de cocina lucharán para cocinarlos y cumplir los deseos de sus amos, los humanos. No pueden permitir que un alimento se desperdicie, por ello se encargan personalmente de evitar que cumplan su sueño de huir al exterior y conocer a su madre la gallina, quien huyó hace tiempo de la cocina.

Los alimentos que se encuentran en la cocina harán lo posible por contarles todo lo que saben y ayudarles a cumplir su objetivo, pues están del mismo bando.

PERSONAJE	DESCRIPCIÓN	CARACTERÍSTICAS	DATOS EXTRA
Clara 	Una de las protagonistas que será controlada por uno de los jugadores	Puede pegarse a cualquier superficie, pues es bastante viscosa. Su personalidad es bastante seria y tranquila. Es la inteligente y responsable del dúo	Está harta de que hagan bromas con su nombre, no le hacen ningún tipo de gracia
Yema 	Una de las protagonistas que será controlada por uno de los jugadores	Puede explotar para atacar a los enemigos. Su personalidad es mucho más alegre y alocada, llegando a ser irritablemente despistada	Tiene facilidad para perderse, Clara siempre tiene que ir detrás de ella
Sartén 	Antagonista, primer boss final del respectivo primer nivel, guardián de la cocina	Sus brazos son espátulas. Tiene la capacidad de usarlas para remover a sus enemigos y cocinarlos cuando se introducen en su cuerpo, la sartén. Tiene una personalidad irritante y es bastante estúpido	Si se araña demasiado a sí mismo sin tocar el alimento dentro de la sartén, se raya y se estropea.
R.E.M.I. 	Una tableta que les hace de personaje guía (tutoriales, indicación de caminos…)	Una actitud muy técnica y friki, le encanta todo lo relacionado con la tecnología. Es muy inteligente y almacena información de utilidad para nuestras protagonistas.	Comienza muchas de sus frases con “técnicamente”
Gallina	Una tableta que les hace de personaje guía (tutoriales, indicación de caminos…)	Es un personaje bastante rebelde, pues cree en la libertad y la posibilidad de huir de la cocina algún día	Ha logrado huir de la cocina, a pesar de haber abandonado a sus hijas. Sin embargo, puede vivir con ello.
Bacín 	Personaje secundario. Este forma parte de uno de los obstáculos del nivel. Se encuentra dentro de la batidora mientras entras está enchufada.	Es un calabacín asustadiizo y con ansiedad, por una parte justificada porque están a punto de triturarlo. No para de suplicar ayuda y no puede pensar con claridad, solo quiere salir de ahí antes de que no quede nada de él	Es un llorica y pone nervioso a todo el que esté cerca

PERSONAJE	DESCRIPCIÓN	CARACTERÍSTICAS	DATOS EXTRA
Nahoria 	Personaje secundario. Personaje con el que interactuar para conseguir informacion sobre como avanzar en el nivel y obtener información sobre la historia principal del juego.	Es una zanahoria muy vivaracha y positiva. Todo las conversaciones que tiene están llenas de energía y siempre ve el lado bueno de las cosas. Conocía a la madre de Clara y Yema e intenta ayudarlas al principio de su viaje.	Es muy olvidadiza entonces la información que da no es de mucha ayuda
Tata 	NPC. Personaje con el que interactuar y mantener una conversacion divertida	Es una patata con una actitud negativa pero muy tranquila. Ha visto como matan a sus hermanas y sabe que es la siguiente, y aun así esta serena. Habla con mucha paz a las protagonistas sobre el horrible futuro que le espera a todos los alimentos de la cocina.	Habla con mucha sinceridad, sus conversaciones no tienen filtro
Chicha 	NPC. Personaje con el que interactuar y mantener una conversacion divertida	Es una salchicha que iban a usar para una recerta hace ya tiempo, pero se olvidaron de ella. Esta un poco reseca, pero ya no tiene de que preocuparse de que cocinen. Se coló detras de un elemento de la cocina y ahí es donde espera a que la encuentren.	Es la última salchicha que queda
Tomi 	Personaje secundario. Personaje con el que interactuar para conseguir informacion sobre como avanzar en el nivel y obtener información sobre la historia principal del juego.	Es un tomate un poco canalla y mala persona. No te puedes fiar de todo lo que te diga porque su modo de entretenimiento es confundir y meter en lios a otros.	A todo el mundo le cae mal y esperan que lo cocinen pronto, pero no ha habido suerte

Cómo se juega

Experiencia del jugador

Tras la pantalla de carga, te aparecerá el menú con la opción del modo historia en el que, con un simple clic, te llevará a una cinemática que te presentará el contexto del videojuego. Esta dará lugar al comienzo de la aventura en la que Clara y Yema tienen un mismo objetivo: averiguar de donde vienes, buscar a tu madre la gallina.

A continuación, se te presentará el escenario del primer nivel, una cocina amplia y repleta de utensilios y aparatos, donde te encontrarás con una tablet, R.E.M.I. nada más empezar, que te hará de guía y te ayudará a dar tus primeros pasos para familiarizarte con los controles y habilidades de cada personaje y así poder enfrentarte a los diferentes obstáculos y jefes.

Una vez superes la fase de plataformas y dialogues con los distintos personajes para avanzar en la historia, te encontrarás al boss final, la sartén, que tratará de derrotarte. Para evitarlo, tendrás que meterte dentro de ella y esquivar sus espátulas cada vez que se aturda tras chocarte con una taza.

Cuando vences a la sartén, comenzará una cinemática que contextualizará el salto al segundo nivel, en la que Clara y Yema son metidas en un tupper y llevadas a la nevera.

Pautas de juego

Debido a que el juego está permitido para menores de edad, la violencia que se muestra no debe ser realista ni muy gráfica. El lenguaje debe ser leve y se puede jugar con el doble sentido de las palabras para fines humorísticos.

La estética entrañable y divertida del juego debe contrastar con el transfondo conflictivo de los personajes, creando situaciones anticlimáticas durante el videojuego.

Objetivos y recompensas del juego

El objetivo principal del juego es descubrir el porqué de la existencia de Clara y Yema. Esto se consigue mediante divertidos puzzles y bosses que fomentan la cooperación mediante el uso de mecánicas específicas de cada personaje, que permiten avanzar en la historia y en los escenarios. También mediante interacciones con otros personajes que se encuentran en la encimera, dando importancia así al diálogo en el videojuego para comprender mejor el contexto y desarrollando así adecuadamente un modo historia.

RECOMPENSAS	PENALIZACIONES	NIV. DE DIFICULTAD
Avanza la historia de los personajes. Se desbloquean nuevas interacciones y nuevas zonas del mapa.	No avanzar la historia, volver a un punto de control anterior cuando te vence un boss o caes del escenario	Nivel de dificultad de los bosses y de los puzzles progresivo.

Mecánica de juego

Ambos jugadores poseen la posibilidad de moverse horizontalmente y verticalmente, además de poder acelerar la velocidad mediante un botón. Cada jugador posee habilidades específicas de cada personaje, asociadas a las características

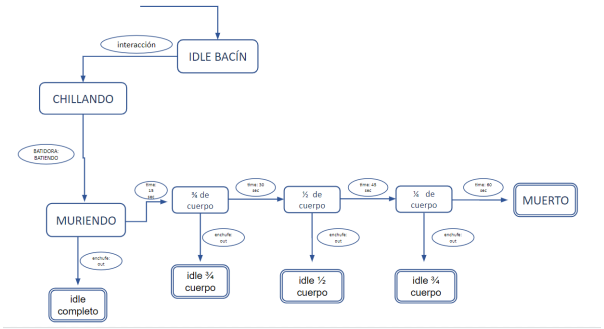
ATRIBUTOS DEL PERSONAJE	
PERSONAJE	**HABILIDADES PERSONALES**
CLARA	Habilidad de movimiento: Elasticidad *para empequeñecer su altura y entrar en zonas pequeñas* Habilidad especial: Pegajosidad *que permite adherirse a algunas superficies*
YEMA	Habilidad de movimiento: Resbalación *para resbalar por zonas que requieran inercia* Habilidad especial: Explosión *permite hacer daño a algunas estructuras o enemigos*

MODOS DE JUEGO	
MODO HISTORIA	El objetivo es completar la historia de los personajes y descubrir sus orígenes pasando por diferentes niveles y enemigos.
MODO ARCADE	El objetivo del modo arcade es poder competir entre los jugadores a través de minijuegos que se desbloquean en el modo historia sin afectar a la historia principal. Solo se puede acceder desde dentro del modo historia

Diagramas de estados

Para el primer nivel hemos desarrollado los diagramas de estados de algunos de los personajes. En primer lugar, lo hicimos del personaje “Batidora” y “Bacín”, que a pesar de formar parte de un mismo conjunto lo hemos diferenciado en 3 personajes para comprender mejor el funcionamiento del mismo. De momento no hemos podido implementarlo en la demo, sin embargo lo hemos adaptado a una simple conversación con Bacín para no entorpecer la historia.

Diagrama de Batidora

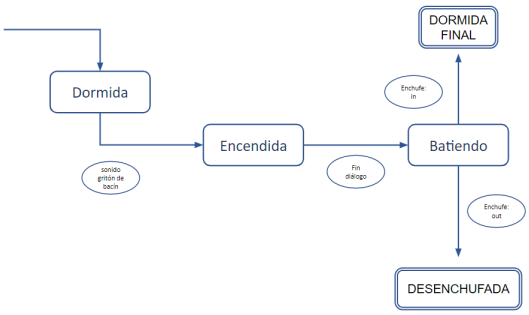


Estados	
IDLE BACÍN	Un idle Bacín sin ser batido, esperando en la batidora
Chillando	Un idle de Bacín hablando con un sonido chillón asustado
Muriendo	El timer comienza y bacín empieza a ser batido
¾ de cuerpo	Bacín está siendo cortado a un 75% de su cuerpo
½ de cuerpo	Bacín está siendo cortado a un 50% de su cuerpo
¼ de cuerpo	Bacín está siendo cortado a un 25% de su cuerpo
Muerto	Bacín se ha convertido en una masa verde
idle ¾ cuerpo	Bacín te agradece con un 75% de su cuerpo cortado
idle ½ cuerpo	Bacín te agradece con un 50% de su cuerpo cortado
idle ¼ cuerpo	Bacín te agradece con un 25% de su cuerpo cortado
idle completo	Bacín te agradece con su cuerpo al 100%

Eventos	
Interacción	Clara y yema interaccionan con el personaje
Batidora: batiendo	La batidora pasa al estado de batiendo
Time: 15 sec	El temporizador es mayor de 15 segundos
Time: 30 sec	El temporizador es mayor de 30 segundos
Time: 45 sec	El temporizador es mayor de 45 segundos
Time: 60 sec	El temporizador es mayor de 60 segundos
Enchufe: out	El enchufe se encuentra en el estado OUT

Eventos/Estados	IDLE BACÍN	Chillando	MURIENDO	¾ DE CUERPO	½ DE CUERPO	1/4 DE CUERPO	Muerto	idle ¾ cuerpo	idle ½ cuerpo	idle ¼ de cuerpo	idle completo
Interacción	Chillando										
Batidora: batiendo		Muriendo									
Time: 15 sec			¾ de cuerpo								
Time: 30 sec				½ de cuerpo							
Time: 45 sec					1/4 de cuerpo						
Time: 60 sec						Muerto					
Enchufe: out			idle completo	idle ¾ cuerpo	idle ½ cuerpo	idle ¼ de cuerpo					

Diagrama de Batidora



Estados	
Dormida	Un idle de la batidora durmiendo
Encendida	Diálogo de la batidora enfadada
Batiendo	La batidora empieza a batir a Bacín, que se pone a dar vueltas y va desapareciendo gradualmente
Desenchufada	La batidora deja de batir
Dormida final	La batidora se duerme de forma permanente

Eventos	
Sonido de grito de Bacín	Grito de socorro de Bacín
Fin diálogo	La batidora acaba de decir su diálogo
Enchufe in	El enchufe está en el estado in cuando se acabe el tiempo
Enchufe out	El enchufe está en el estado out

Eventos/Estados	Dormida	Encendida	Batiendo	Desenchufada	Dormida final
Sonido de grito de Bacín	Encendida				
Fin diálogo		Batiendo			
Enchufe in			Dormida Final		
Enchufe out			Desenchufada		

Diagrama del enchufe de Batidora

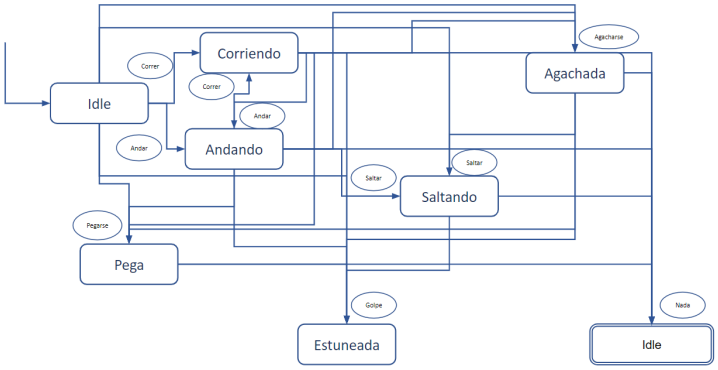
Estados	
IN	Enchufe conectado a la luz
OUT	Enchufe desconectado de la luz

Eventos	
Explosión Yema	El enchufe se encuentra dentro de la zona de colisión de la habilidad de explotar del personaje yema

Eventos/Estados	IN	OUT
Explosion yema	Out	

También hemos hecho el diagrama de estados de las protagonistas de la historia, Clara y Yema, y del boss final, Sartén.

Diagrama de estados de Clara

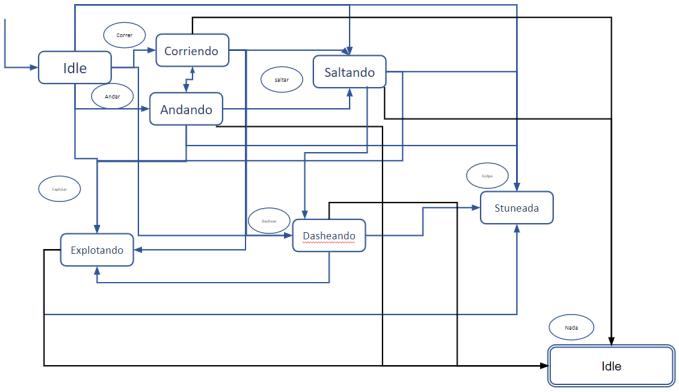


Estados	
Idle	Un idle de Clara
Andando	Clara anda
Corriendo	Clara corre
Saltando	Clara salta
Pega	Hace uso de su habilidad y pega algún objeto
Estuneada	Clara ha sido golpeada o se ha hecho daño
Agachada	Clara se agacha

Eventos	
Correr	Corre en cualquier dirección
Andar	Anda en cualquier dirección
Saltar	Salta
Agacharse	Se agacha
Pegarse	Usar la habilidad para pegar un objeto
Golpe	Recibe un golpe o daño
Nada	El jugador deja de realizar una acción

Eventos/Estados	Idle	Andando	Corriendo	Saltando	Pega	Estuneada	Agachada
Nada		Idle	Idle	Idle	Idle		Idle
Correr	Corriendo	Corriendo					
Andar	Andando		Andando				Andando
Saltar	Saltando	Saltando	Saltando		Saltando		Saltando
Agacharse	Agachada	Agachada	Agachada				
Golpe	Estuneada	Estuneada	Estuneada	Estuneada	Estuneada		Estuneada
Pegarse	Pega	Pega	Pega				Pega

Diagrama de estados de Yema

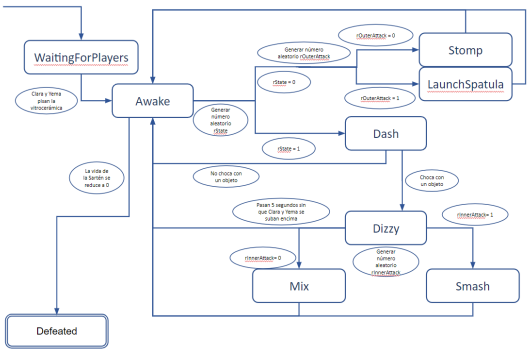


Estados	
Idle	Un idle de Yema
Andando	Yema anda
Corriendo	Yema corre
Saltando	Yema salta
Explotando	Hace uso de su habilidad y aplica una fuerza a los objetos interactuables
Dasheando	Yema dashea
stuneada	Yema ha sido golpeada o se ha hecho daño

Eventos	
Correr	Corre en cualquier dirección
Andar	Anda en cualquier dirección
Saltar	Salta
Dashear	Dashea
Explotar	Usar la habilidad para impulsar objetos
Golpe	Recibe un golpe o daño
Nada	El jugador deja de realizar una acción

Eventos/Estados	Idle	Andando	Corriendo	Saltando	Explotando	Estuneada	Dashear
Nada		Idle	Idle	Idle	Idle		Idle
Correr	Corriendo	Corriendo					
Andar	Andando		Andando				
Saltar	Saltando	Saltando	Saltando				Saltando
Dashear	Dashear	Dashear	Dashear	Dashear			
Golpe	Estuneada	Estuneada	Estuneada	Estuneada	Estuneada		Estuneada
Explotar	Explotar	Explotar	Explotar	Explotar			

Diagrama de estados de Sartén



Estados	
WaitingForPlayers	Espera a que Clara y Yema entren en la vitrocerámica
Awake	Estado que redirige a los demás estados
Stomp	Golpea el suelo con su cuerpo
LaunchSpatula	Lanza sus brazos en dirección a los dos jugadores
Dash	Se desliza hacia el jugador más cercano, puede chocar con elementos de la escena
Dizzy	Se queda quieta durante 5 segundos
Mix	Mueve sus espátulas dentro de la sartén para acabar con Clara y Yema
Smash	Golpea sus espátulas dentro de la sartén para acabar con Clara y Yema
Defeated	Es derrotada

Eventos	
Clara y Yema pisan la vitrocerámica	Clara y Yema pisan la vitrocerámica
Generar número aleatorio rState	Genera un número para decidir la acción a realizar
Generar número aleatorio rOuterAttack	Genera un número para decidir la acción a realizar
Choca con un objeto	Al dashear choca con un objeto
No choca con un objeto	Al dashear no choca con un objeto
Generar número aleatorio rInnerAttack	Genera un número para decidir la acción a realizar
La vida de la Sartén se reduce a 0	Pierde toda la salud
Pasan 5 segundos sin que Clara y Yema se suban encima	Pasan 5 segundos sin que Clara y Yema se suban encima de la sartén cuando esté en estado Dizzy

Diseño de nivel

De momento los niveles que se han pensado son 5: La encimera, la nevera, la despensa, el establo y el bosque. En este ultimo nivel nos encontraremos a la Gallina, el objetivo final del juego. Para este proyecto, de momento, solo elaboraremos el primer nivel a modo de Demo y el salto al segundo nivel, la nevera.

PRIMER NIVEL: ENCIMERA

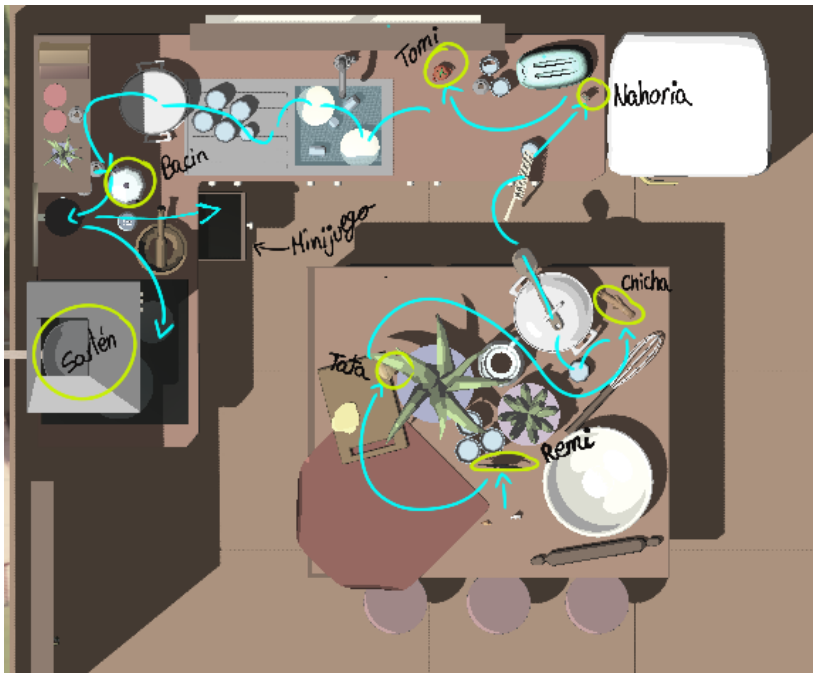
Cuando en el menú haces click al modo historia, el juego comienza con una breve cinemática estilo viñetas que contextualiza la situación.

En ella se muestra a una persona cocinando una tortilla que, de repente, justo antes de batir el huevo, se da cuenta de que se ha olvidado del aceite. Cuando sale a comprarlo, Clara y Yema, que todavía siguen en el Bowl, se separan y cobran vida. Ambas, decididas a averiguar de donde provienen se levantan y salen del bowl iniciando así el modo de juego.

La jugabilidad comienza una vez fuera del bowl, sobre una isla de cocina bastante desordenada. Primero se encontrarán con R.E.M.I, con quien tendrán que interactuar para entender cómo moverse en la cocina y cómo comenzar su investigación.

Una vez han comprendido las mecánicas a través de la explicación con los diálogos, se moverán por toda la isla de cocina aprovechando las mecánicas y esquivando los obstáculos. También podrán interactuar con Tata y Chicha, que no aportarán nada a la historia pero cuentan con diálogos divertidos para dar personalidad al juego. En cierto punto, tendrán que saltar a la encimera para avanzar. Para ello tendrán que subirse a la olla que tiene un cucharón de madera enganchado en la tapa. Para saltar al otro lado, Yema deberá aprovechar la inercia de velocidad para ganar potencia de salto. Para que Clara pueda pasar, Yema usará su habilidad “explosión” para mover la escoba que se encuentra apoyada en la encimera hacia la isla de cocina, y así ayudar a Clara a pasar a la encimera

Una vez en la encimera, se encuentran a los personajes Tomi y Zahoria, quienes parecen conocer bastante bien a su madre, pero no son capaces de darles más pistas. Les recomienda que avancen y hablen con Bacín el calabacín, que parece estar en la batidora a punto de ser triturado. Para ello saltan los obstáculos del fregadero, en el que si caes en el agua debes volver a empezar desde el último checkpoint.



Una vez pasan el fregadero, deben subir por el escurridor hasta la estantería, donde podrán interactuar con Bacín.

*Esta parte todavía está por desarrollar en el videojuego, de momento es una conversación con Bacín sobre como vencer a la sartén

Sin embargo, la batidora, molesta por el ruido de la conversación, se activa y comienza a triturar a Bacín. Clara y Yema tienen tiempo límite para salvarlo. Para ello, Clara debe desenchufarla con su habilidad especial, la pegajosidad. Una vez la desactivan, Bacín les agradece su ayuda y les aconseja que derroten a la sartén, pues cuando vuelva el humano lo untará de aceite y comenzará a freir las patatas, siendo clara y yema las siguientes víctimas. Para ello, ambas bajan a la cocina. Sin embargo, se encuentran un cajón abierto.

Este cajón abierto es el primer arcade que da lugar al primer minijuego competitivo, un laberinto de cubiertos. El que logre primero llegar al final del laberinto, ganará. Sin embargo, no hay ninguna recompensa. Tan solo la satisfacción de ganar a tu compañero.

Una vez salen del arcade, es el momento del boss final. Este boss es la sartén, a quien tendrán que engañar para que se raye a si mismo con las espátulas hasta debilitarse. Sartén empieza a lanzar objetos detrás de los que Clara y Yema se deben esconder, para que al atacarlas se choquen contra ellos y se quede aturdido. Aquí, los jugadores deberán entrar a dentro de la sarten y esquivar las espátulas, que harán que su vida baje poco a poco.

Una vez la sartén pierde toda la vida, se acaba el primer nivel.Entonces comienza la cinemática que da paso al siguiente nivel, la nevera, pues la humana regresa y al ver que la sarten está estropeada y ya no puede hacer la tortilla, guarda a Clara y a Yema en un tupper y las mete en la nevera, el escenario del nivel 2.

ESQUEMA DE CONTROL

BOTÓN	ACCIÓN
JOYSTICK IZQUIERDO	MOVIMIENTO CAMINAR/CORRER
START	MENÚ
B	SALTAR
A	HABILIDAD MOVIMIENTO CLARA / YEMA
Y	HABILIDAD ESPECIAL CLARA/ YEMA
X	INTERACTUAR

Estética del juego

El juego se desarrollará en un entorno 3D, pero estará texturizado con un aspecto cartoon, como si fuera 2D. Usaremos para ello la técnica del Toon Shader para crear texturas planas, sombras duras y figuras con delineado para dar la sensación de dibujo. El escenario es un entorno 3D que te permite moverte libremente por él dentro de un trayecto recomendado para cada personaje, para poder avanzar en la historia con efectividad.

LA CÁMARA

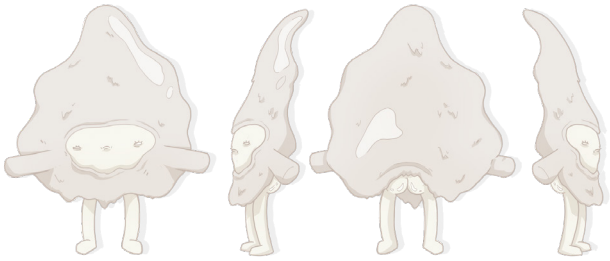
Los personajes deberán cooperar para superar los niveles, por lo que los jugadores deberán comunicarse y apoyarse, lo que lleva a una interacción en la vida real que enriquece la experiencia del juego. La cámara funcionará de manera que cuando los personajes estén juntos en el escenario, la pantalla será única y la misma para ambos jugadores. En el momento en el que los personajes se separen, la cámara también se partirá, permitiendo así la posibilidad de explorar individualmente. Para ello empleamos el plugin de Unity Assets "Adaptative Split Screen".

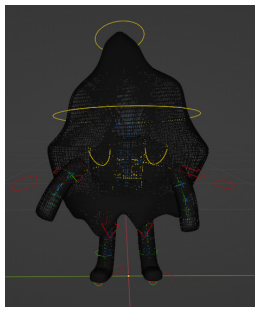
MODELADOS Y ANIMACIONES

Todos los assets, tanto de personajes como de escenarios, han sido modelados y texturizados por el equipo. Para ello se elaboró una primera fase de concept art para los personajes con las distintas vistas de los mismos, para poder modelarlos con la herramienta Blender de la forma más efectiva posible. Después, se pintaron las texturas con los colores indicados también en las ilustraciones.

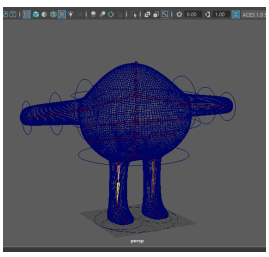
Las animaciones también fueron elaboradas por nuestro equipo de diseño, ya que debían adaptarse a la anatomía de los mismos, de modo que no nos servía cualquier rigeado. La herramienta utilizada para animar ha sido Maya. A continuación, os mostramos en orden de izquierda a derecha el concept, el modelado y el rigeado de cada personaje.

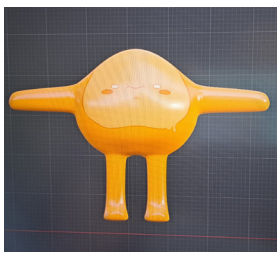
Clara



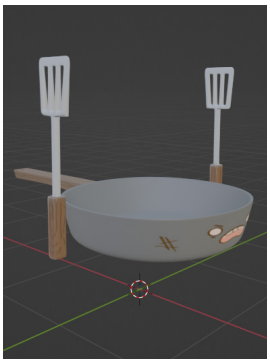


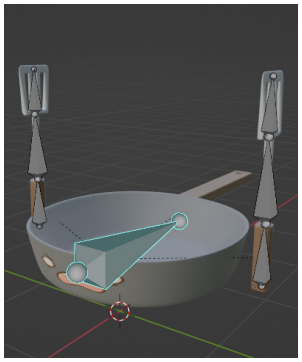
Yema



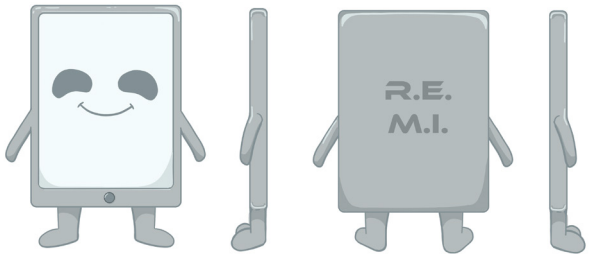


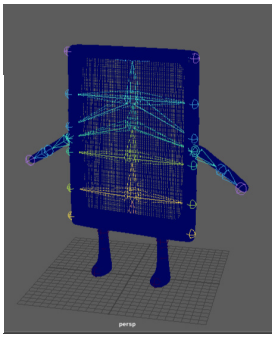
Sartén



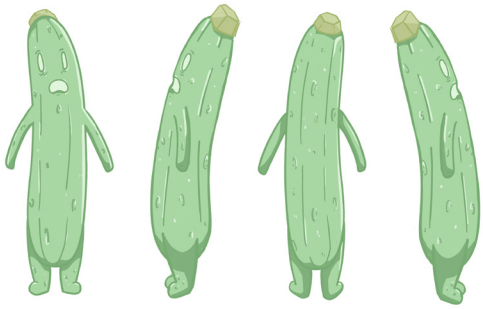


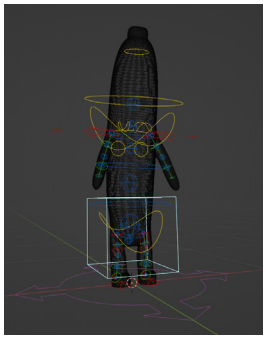
R.E.M.I.



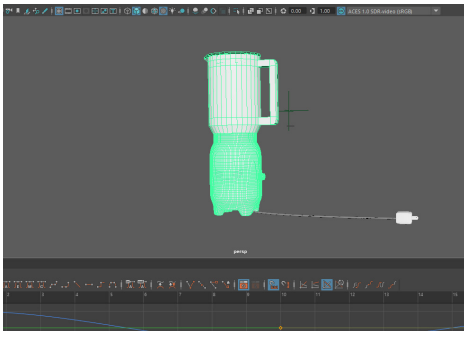


Bacín

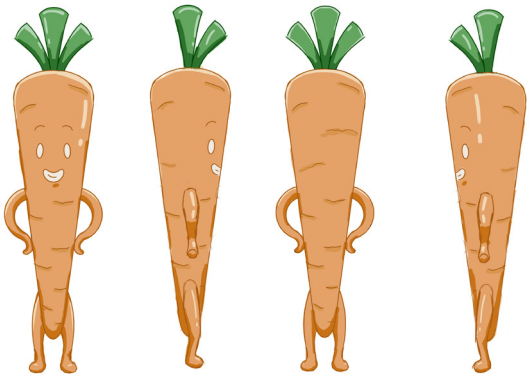


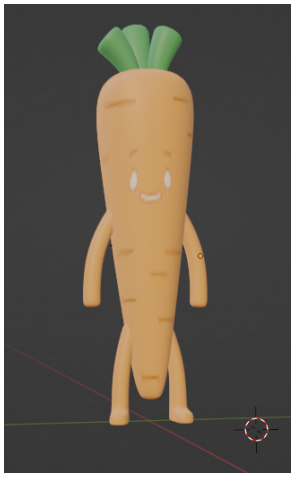


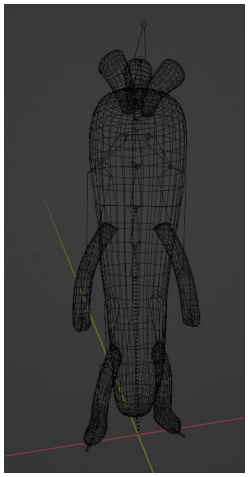
Batidora



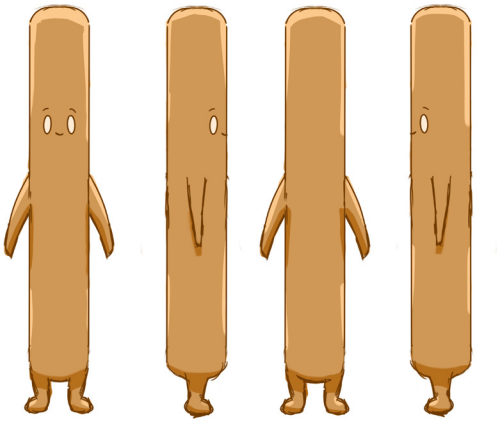
Zahoria

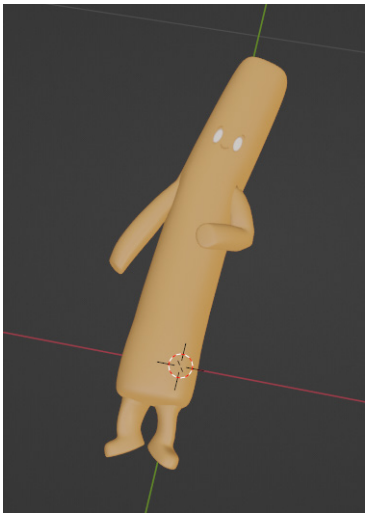


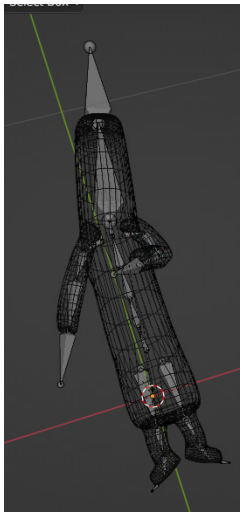




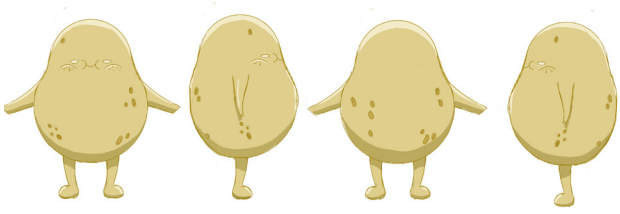
Chicha

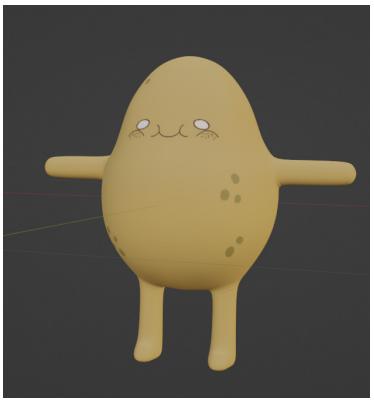


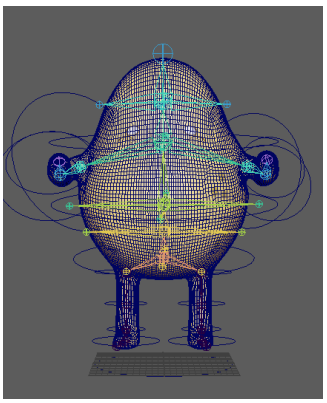




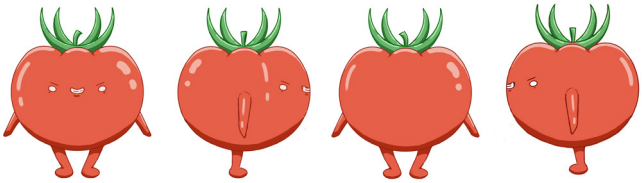
Tata

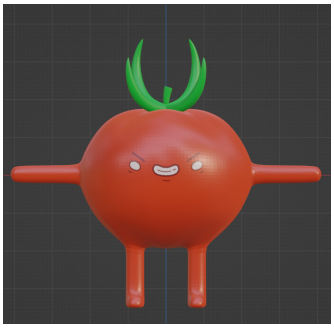


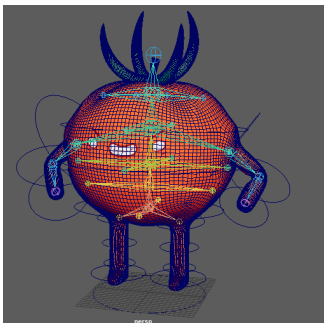




Tomi







Por otro lado, el escenario tuvo un desarrollo más improvisado. Por la falta de tiempo no podíamos hacer un concept para cada objeto, que sería lo ideal. Sin embargo, hemos desarrollado el nivel con bocetos generales del escenario y despues hemos ido modelando los diferentes assets en función de lo que ibamos incluyendo en este boceto, es decir, en función de nuestras necesidades.

Modelado del escenario

PRIMEROS MODELADOS DEL ESCENARIO EN BLENDER CON HIGHPOLY

PRUEBAS DE COLOR DEL ESCENARIO EN BLENDER CON TEXTURAS NO FINALES

Escenario texturizado y montado en Unity

AMBIENTACIÓN

Para la ambientación utilizamos una iluminación que permitiera que los toon shaders se vieran correctamente, para ello utilizamos un punto de luz que proviene de la ventana “directional light” que da color a la escena. También utilizamos la iluminación del skybox.

El skybox utilizado lo conseguimos de un generador de skybox por inteligencia artificial, pidiendo que nos mostrase un exterior cartoon de un pueblo en un mediodía. para que en caso de verse el exterior de la ventana, se viera esa vista.

SONIDO

El sonido implementado en el videojuego tiene como función ambientar. Los requisitos para escoger la música es que fuese tranquila pero divertida. No queríamos que fuese una molestia para el usuario por lo que se buscó una canción con pocos instrumentos, repetitiva y pegadiza. Además al ser un juego cooperativo, los jugadores deben poder hablar entre ellos sin que la música se interponga o sea molesta en las conversaciones.
El estilo musical que más se acercaba a lo que se buscaba era pop, un estilo familiar para todo el mundo, con melodías sencillas y pegadizas que pueden ser más o menos relajadas según se busque.
El autor de todas las obras que conforman la banda sonora del juego son del mismo autor, llamado セカンドスチューデント.Este es un usuario de YouTube que crea obras cortas y las deja a libre disposición para que los usuarios puedan usarlas en sus proyectos.
La primera canción que te encuentras es la que suena durante el menú principal y la animatica inicial. Esta canción se llama 初心者中学生が作曲してみた. Esta es una pieza tranquila que da contexto del ambiente del juego y te introduce en el mundo de estética amigable y divertida.
Durante el juego podemos escuchar 有村架純より可愛い曲. En este caso, es una pieza con un ritmo un poco más animado que el de la cinemática, ya que durante el nivel se crearán situaciones dinámicas. Sin embargo, durante el nivel hay momentos de diálogo con los personajes, por lo que la obra es lo suficientemente tranquila como para no molestar en esos casos.
Finalmente, cuando ocurre el enfrentamiento contra el jefe final, la canción que se puede es cuchar es ばかエモかっこいい曲作ってみたっ. Esta tiene el un ritmo mucho más movido que el resto. El objetivo de esta canción es situar al jugador en un contexto y un ánimo más activo y preparado para un ritmo de juego más rápido. Durante este enfrentamiento no se necesita atender a diálogos, lo único que se pretende es que los jugadores estén más activos y que reaccionen más rápido a los estímulos de la pantalla.
A lo largo del juego las canciones suenan en bucle mientras dure el nivel donde se incluye esta canción. Como ya se ha mencionado esto ayuda a que la música no distraiga al jugador y a la vez que los usuarios sean capaces de recordarlas y asociarlas con los niveles.

Interfaz de usuario

En la pantalla inicial te sale varias opciones a escoger: Historia, Opciones, Créditos o Salir. Dentro del modo historia podrás comenzar una nueva partida. Tras esto pasarás a una pantalla en la que cada jugador elegirá con qué personaje quiere jugar, donde comenzará el juego

El fondo del menú es una ilustración de la cocina con R.E.M.I., que funcionará como gestor del menú inicial.

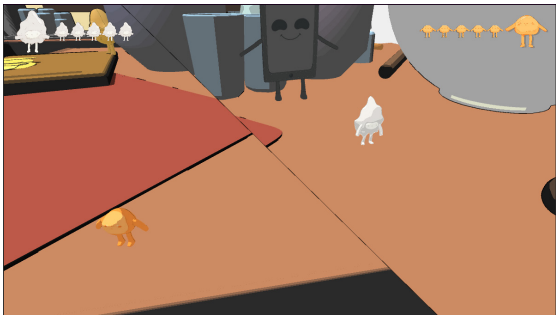
El menú consiste en unos botones de línea y texto que se rellenarán y cambiarán de color cuando estén seleccionados. Cuando entres a una de las opciones las cajas cambiarán para mostrarte el siguiente menú.

Para elegir los personajes cada jugador, usando el joystick de los mandos, seleccionará el personaje con el que juega.

Durante los niveles, el HUD será muy simple.

La vida de Clara y Yema se ven durante todo el juego, pues tienen posibilidad de perder vida durante todo el gameplay. Son 5, y si caen al vacío o son atacadas por el boss final, baja la vida. Por otro lado, está la barra de vida del boss, que también baja progresivamente cuando se raya a si mismo con las espátulas

En cualquier momento, podrás pausar el juego y se abrirá un menú con el aspecto de remi desde el que podrás salir de la partida.



En el minijuego arcade saldrá un elemento en la interfaz que indicará el ganador del laberinto.

Y, para terminar, los diálogos aparecerán cada vez que te acerques a los personajes. Se pasan con la X, e incluyen una imagen del personaje con el que estás hablando. Para próximas versiones hemos considerado añadir también el nombre del personaje.

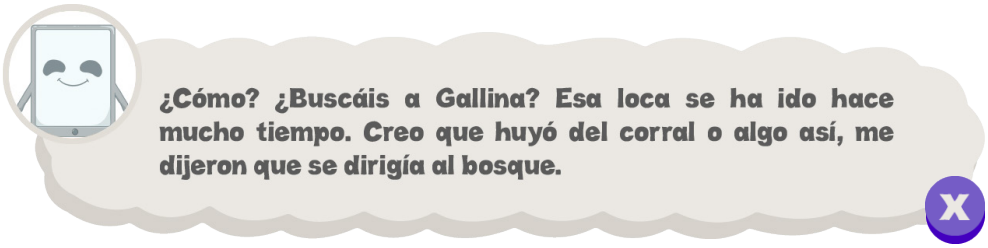
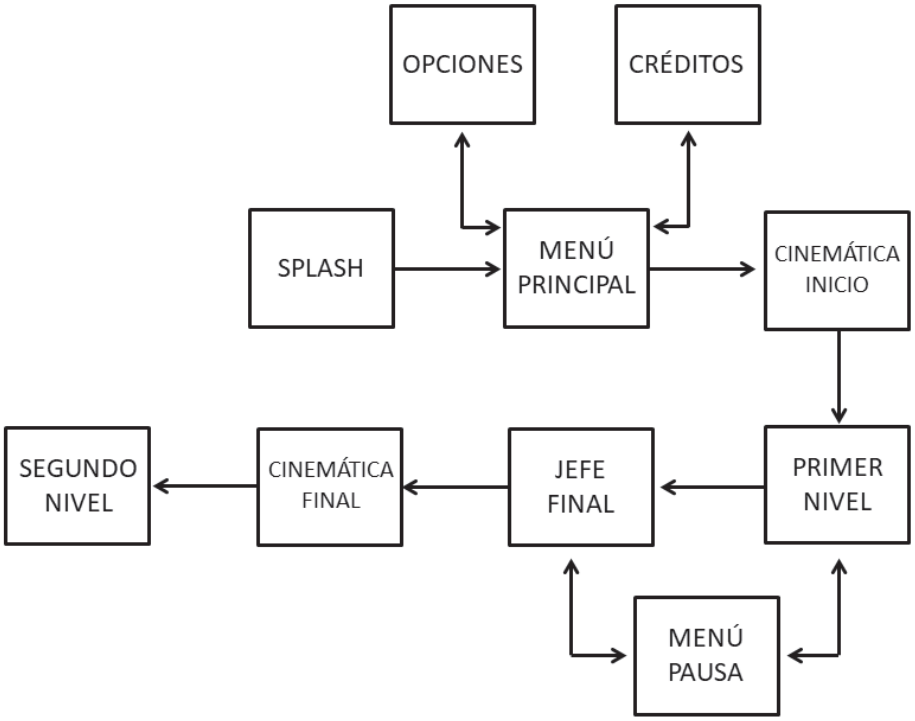
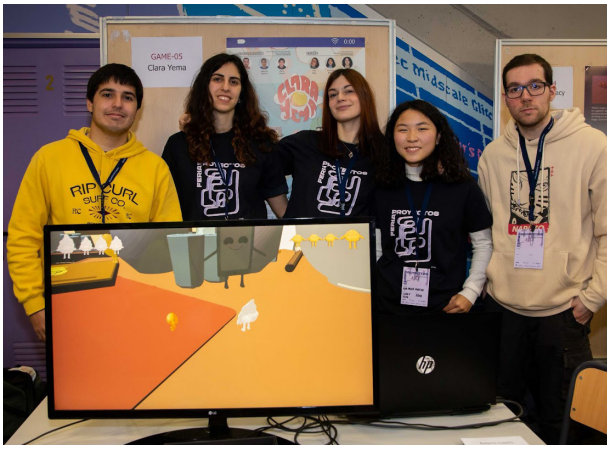


Diagrama de navegación entre pantallas

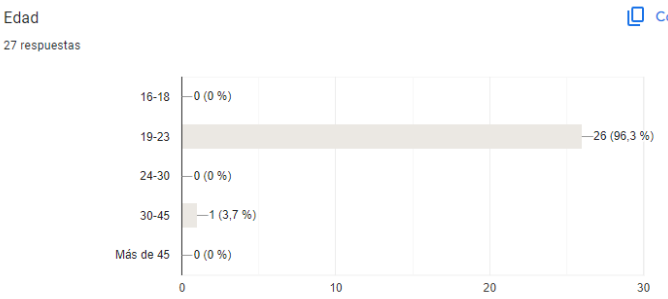


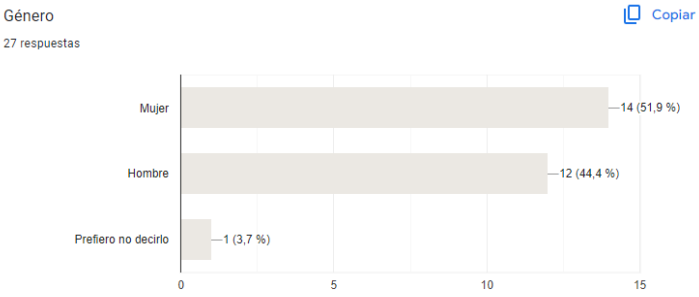
Fase de testeo

Una vez elaborada la Demo del videojuego, tuvimos la oportunidad de mostrarlo en la Feria de Proyectos ETSINF 2023. Esto nos permitió ver cual era la perspectiva de los jugadores, los errores y aciertos del videojuego y dar visibilidad al proyecto. Para ello elaboramos una encuesta, la cual los usuarios podían rellenar al terminar de probar el juego. Para incentivar a la gente a completarla, hemos impreso pegatinas de clara y de yema y las hemos dado a cambio de que la rellenasen.

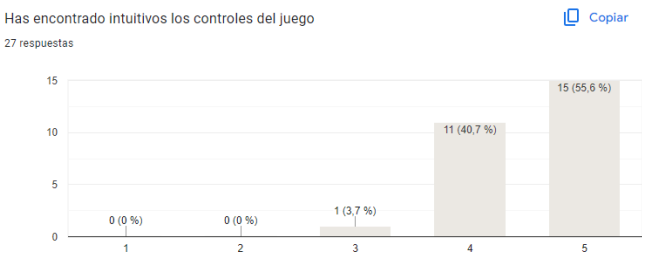


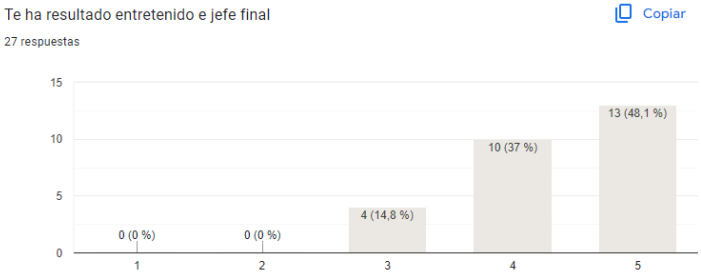
RESULTADOS DE LAS ENCUESTAS

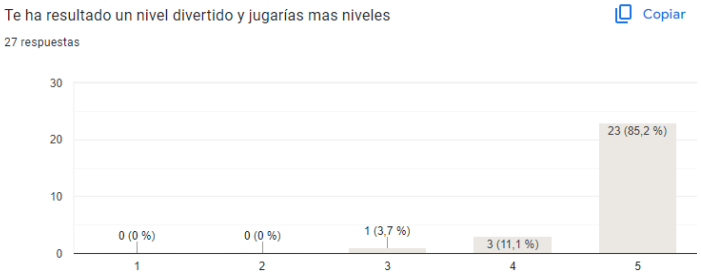


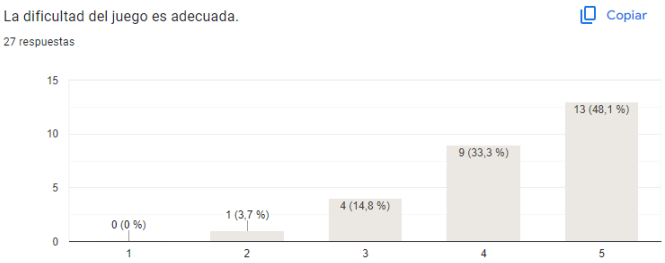


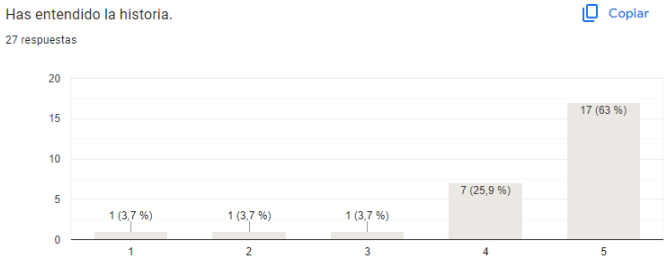
Primero pedimos que nos puntuasen del 1 al 5 las siguientes afirmaciones, siendo 1 nada de acuerdo y 5 muy de acuerdo











Preguntamos si tenian alguna sugerencia para las mecánicas del juego o la jugabilidad, y las respuestas fueron las siguientes:

"El bug de la sartén que atraviesa paredes. Como yema no sabinque debia utilizar mi habilidad especial para que clara psase por la escoba. Creo que en el boss se deberia de mencionar algo de como stunearlo porque no me parecio intuitivo ponerse detras de la taza"

"Cuando tienes que traspasar objetos y sale el circulo alrededor del personaje desorienta mucho, a lo mejor si solo se viese translúcido detrás del objeto se entendería el espacio y hacia donde vas mejor, y el colíder raro que te deja flotando encima del bos tambien"

"Arreglar un pelín la sarten (Y las tacitas)"

"QUIERO VER A LA GALLINAAAA"

"Sería guay que tuvieran algún control en común, o que cuando se juntan hagan alguna habilidad especial"

"Más fluida la pelea"

"Más habilidades y más enemigos contra quien usarlas"

"Más escenarios y con objetivos más claros"

"Flechas que indiquen por dónde ir"

También preguntamos si tenían alguna sugerencia para mejorar la historia o los personajes, y las respuestas fueron estas:

"Que quede más claro el tema de la sartén"

"Cuando hablan los personajes pondría sus nombres, que son muy originales :)"

"El skybox se ve reflejado en la vitrocerámica, la historia una fantasía"

"La historia es turbia pero maravillosa."

"Nada, todo geniaaal"

"Son muuuy graciosos los personajes, el diseño es adorable"

"Nop"

"Más plataformeo"

"Que haya una historia detrás de los dos personajes y un objetivo bonito"

"Los mensajes de diálogo deberían bloquear los movimientos"

CAMBIOS REALIZADOS TRAS LAS ENCUESTAS

Debido a la falta de tiempo hay ciertos cambios que no hemos podido llevar a cabo. Sin embargo, realizamos lo siguiente:

1. Aumentamos el tamaño del circulo para que fuera mas entendible al atravesar la parte de la jarra y los jugadores no se desorientasen tanto

2. Dimos mas vidas a los jugadores para que la dificultad no fuese tan alta

3. Modificamos las velocidades del boss final para que al jugador le diese tiempo a reaccionar, y a la vez no se hiciera aburrido.

4. Arreglamos el minijuego para dar mas jugabilidad de niveles.

5. Añadimos un muro antes de la sartén para que la gente si o sí tuviese que hablar con el calabacín para saber cómo vencer a la sartén, ya que la gente se lo saltaba y llegaba a la sartén sin saber como vencerla.

6. Modificamos la textura de la vitrocerámica para que no saliera el skybox reflejado

Creemos que los cambios realizados facilitaron la jugabilidad y comprensión del juego, además de alargar el gameplay gracias al minijuego que pudimos arreglar tras la feria.

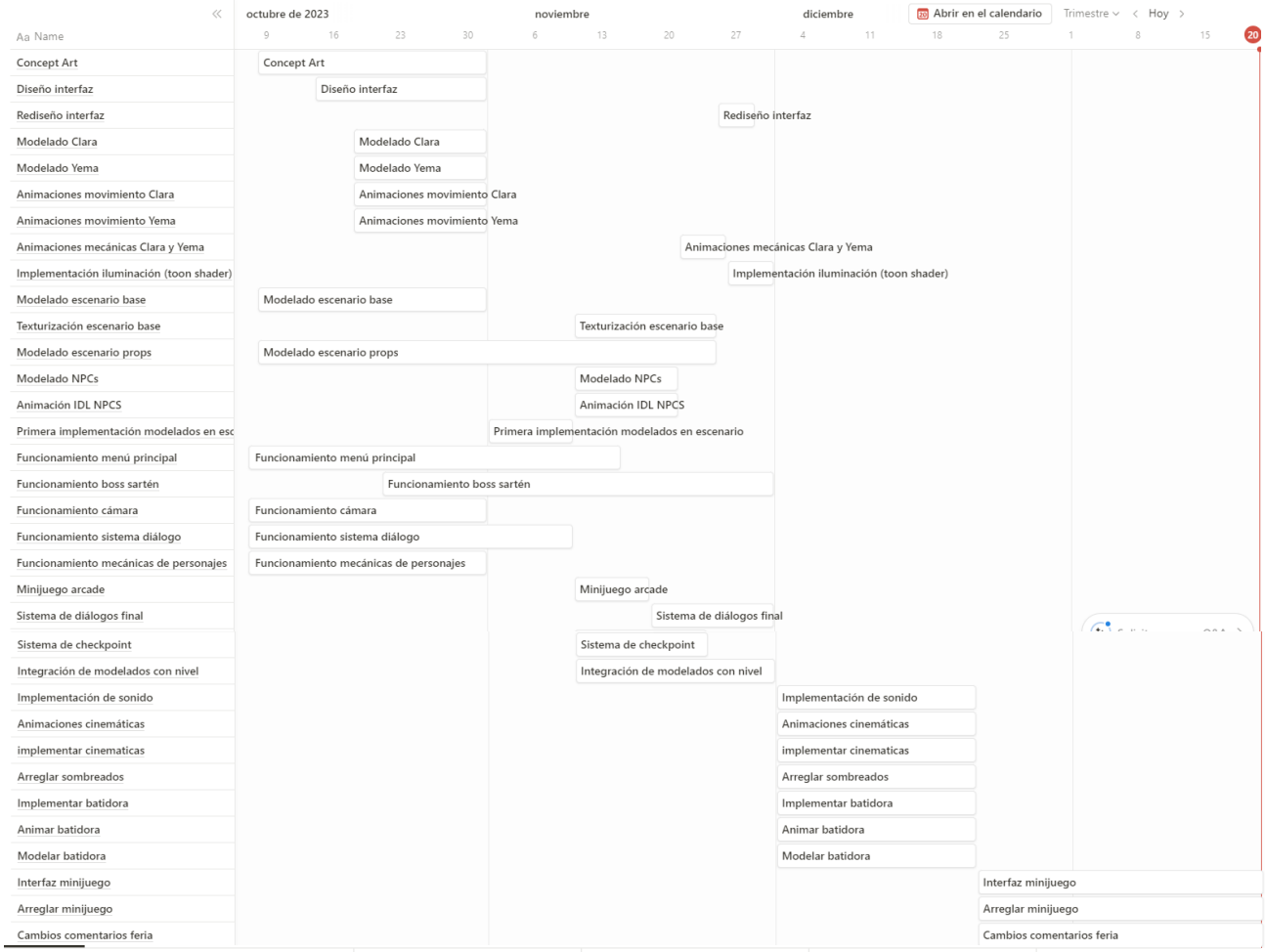
Organización de equipo

Para organizarnos, el equipo utilizó la herramienta Notion, que permite hacer un seguimiento paralelo de las tareas con el estado de la tarea, la persona que lo hace y las fechas en las que se debería hacer.

Nos repartimos los trabajos de concept, diseño, modelado y animacion en el equipo de diseño. Por otro lado, paralelamente, el equipo de informática se encargaba más de la programación de los scripts en el programa. Por otro lado, ambos equipos se encargaron de la implementación de todo en Unity, manteniendonos en constante contacto a través de reuniones y conversaciones en las plataformas Discord y Whatsapp.

Consideramos que la repartición de trabajo ha sido bastante equilibrada y que se ha seguido un buen flujo de trabajo.

DIAGRAMAS DE GANTT Y REPARTICIÓN DE TAREAS



Aa Name	Tags	Status	Person	Date
Concept Art		Done	Celia Veiga López Ana Paula Moreno García Clara Gaher	8 de octubre de 2023 → 31 de octubre de 2023
Diseño interfaz		Done	Clara Gaher Ana Paula Moreno García	14 de octubre de 2023 → 31 de octubre de 2023
Modelado Clara		Done	Clara Gaher	18 de octubre de 2023 → 31 de octubre de 2023
Modelado Yema		Done	Ana Paula Moreno García	18 de octubre de 2023 → 31 de octubre de 2023
Animaciones movimiento Clara		Done	Clara Gaher	18 de octubre de 2023 → 31 de octubre de 2023
Animaciones movimiento Yema		Done	Ana Paula Moreno García	18 de octubre de 2023 → 31 de octubre de 2023
Animaciones cinemáticas		Done	Celia Veiga López Ana Paula Moreno García Clara Gaher	1 de diciembre de 2023 → 21 de diciembre de 2023
Implementación de sonido		Done	Marta Colomer Alejandro Navarro Adrián Soriano Rodriguez	1 de diciembre de 2023 → 21 de diciembre de 2023
Integración de modelados con nivel		Done	Celia Veiga López Ana Paula Moreno García Clara Gaher Marta Colomer Alejandro Navarro Adrián Soriano Rodriguez	10 de noviembre de 2023 → 30 de noviembre de 2023

Sistema de checkpoint		Done	Adrián Soriano Rodriguez Alejandro Navarro	10 de noviembre de 2023 → 23 de noviembre de 2023
Sistema de diálogos final		Done	Marta Colomer	18 de noviembre de 2023 → 30 de noviembre de 2023
Animaciones mecánicas Clara y Yema		Done	Clara Gaher Ana Paula Moreno García	21 de noviembre de 2023 → 25 de noviembre de 2023
Texturización escenario base ABRIR		Done	Celia Veiga López	10 de noviembre de 2023 → 24 de noviembre de 2023
Minijuego arcade		Done	Marta Colomer	10 de noviembre de 2023 → 17 de noviembre de 2023
Implementación iluminación (toon shader)		Done	Celia Veiga López	26 de noviembre de 2023 → 30 de noviembre de 2023
Funcionamiento mecánicas de personajes		Done	Adrián Soriano Rodriguez	7 de octubre de 2023 → 31 de octubre de 2023
Funcionamiento sistema diálogo		Done	Marta Colomer	7 de octubre de 2023 → 9 de noviembre de 2023
Funcionamiento cámara		Done	Alejandro Navarro	7 de octubre de 2023 → 31 de octubre de 2023
Funcionamiento boss sartén		In progress	Alejandro Navarro	21 de octubre de 2023 → 30 de noviembre de 2023
Funcionamiento menú principal		Done	Alejandro Navarro	7 de octubre de 2023 → 14 de noviembre de 2023
Primera implementación modelados en escenario		Done	Adrián Soriano Rodriguez	1 de noviembre de 2023 → 9 de noviembre de 2023
Animación IDL NPCS		Done	Clara Gaher Ana Paula Moreno García	10 de noviembre de 2023 → 20 de noviembre de 2023
Modelado NPCs		Done	Clara Gaher Ana Paula Moreno García	10 de noviembre de 2023 → 20 de noviembre de 2023
Modelado escenario props		Done	Celia Veiga López	8 de octubre de 2023 → 24 de noviembre de 2023
Modelado escenario base		Done	Celia Veiga López	8 de octubre de 2023 → 31 de octubre de 2023
Rediseño interfaz		Done	Celia Veiga López Clara Gaher Ana Paula Moreno García	25 de noviembre de 2023 → 28 de noviembre de 2023
Cambios comentarios feria		Done	Adrián Soriano Rodriguez Marta Colomer Ana Paula Moreno García Alejandro Navarro Celia Veiga López Clara Gaher	22 de diciembre de 2023 → 20 de enero de 2024
Arreglar minijuego		Done	Alejandro Navarro Marta Colomer Adrián Soriano Rodriguez	22 de diciembre de 2023 → 20 de enero de 2024
Interfaz minijuego		Done	Ana Paula Moreno García Marta Colomer Alejandro Navarro	22 de diciembre de 2023 → 20 de enero de 2024
Arreglar sombreados		Done	Adrián Soriano Rodriguez	1 de diciembre de 2023 → 21 de diciembre de 2023
implementar cinematicas ABRIR		Done	Alejandro Navarro	1 de diciembre de 2023 → 21 de diciembre de 2023
Implementar batidora		In progress	Adrián Soriano Rodriguez Alejandro Navarro Marta Colomer	1 de diciembre de 2023 → 21 de diciembre de 2023
Animar batidora		Done	Ana Paula Moreno García Clara Gaher	1 de diciembre de 2023 → 21 de diciembre de 2023
Modelar batidora		Done	Clara Gaher Ana Paula Moreno García	1 de diciembre de 2023 → 21 de diciembre de 2023

Conclusiones

A pesar de estar en una primera fase, estamos orgullosos del resultado de la DEMO. Creemos que hemos trabajado adecuadamente en equipo y que hubo buena comunicación en todo momento. Esto se refleja en los resultados de un proyecto tratado en todo momento con bastante interés por parte de todos y con cada detalle cuidado.

Al ser un videojuego multijugador en la Feria de proyectos de la ETSINF hemos recibido muy buenos comentarios y buenas caras de nuestro público, lo cual nos ha dejado también a nosotros con buenas sensaciones.

Nada de esto hubiera sido posible sin todo el esfuerzo previamente (y posteriormente) realizado, las reuniones y las presentaciones con feedback, que permitieron que fueramos preparados con un buen producto a la misma.

A pesar de quedar cosas por implementar, como el minijuego de la batidora o arreglar algunos de los fallos que nos comentaron en la feria, creemos que el videojuego resulta entretenido y de una facilidad adecuada para el público que buscamos. Esperamos que así mismo lo sientan nuestros jugadores.