UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Melilac: desarrollo de un videojuego de combate en tercera persona con opción de multijugador en línea

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Gustavo Lee Kâm Châk

Tutor: Ramón Pascual Mollá Vayá

Curso 2020-2021

A Ramón, por haberme ayudado a superar una etapa que llevo esperando toda una vida.
A Simone, por la eterna paciencia y el amor infinito de una madre.
A Luca, por llenarnos el estómago con las mejores recetas y el corazón con los mejores versos.
A Miguel, por cuidar tan bien de esta familia y compartirnos siempre un buen licor.
A mis amigos, por las tardes de cervezas y las noches de videojuegos.
Y a María, por el apoyo incondicional de quien quiere mucho.
Gracias.

Resum

El projecte tracta d'un videojoc de lluita fet en Unity amb l'opció de multijugador online.

El motor gràfic Unity servix com a ferramenta principal que actuarà d'interfície visual, el que permetrà crear el videojoc de combat en tercera persona i arran d'això integrar els continguts d'autor necessaris per a dissenyar el videojoc. Complementàriament es realitza una tasca d'anàlisi de l'oferta de motors de comunicació que permetrà integrar un sistema de multijugador online al videojoc i que incloga diversos servicis que faciliten realitzar proves sobre el funcionament d'este sistema multijugador.

Paraules clau: Videojoc, Joc de lluita, Multijugador online, Unity, Online, Combat, C#

Resumen

El proyecto se trata de un videojuego de combate con opción a multijugador en línea desarrollado en Unity.

El motor gráfico Unity sirve como herramienta principal que actuará de interfaz visual para poder crear el videojuego de combate en tercera persona e integrar los contenidos de autor necesarios para diseñar el videojuego. Complementariamente se realiza una tarea de análisis de la oferta de motores de comunicación que permitirá integrar un sistema de multijugador en línea al videojuego y que incluya varios servicios que faciliten realizar pruebas sobre el funcionamiento de este sistema multijugador en línea.

Palabras clave: Videojuego, Juego de lucha, Multijugador en línea, Unity, Online, Combate, C#

Abstract

This project is about a fighting video game with online multiplayer made in Unity.

The Unity game engine serves as the main tool that will act as a visual interface to create the third-person combat video game and integrate the necessary resources to design the videogame. In addition, an analysis of the offer of communication engines that will allow the integration of an online multiplayer system to the videogame and that includes several services that will facilitate testing the operation of this online multiplayer system.

Key words: Video game, Fighting game, Online multiplayer, Unity, Online, Combat, C#

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

1.1 Motivación

La industria de los videojuegos es uno de los sectores económicos con más auge y que más dinero genera actualmente. Con el paso del tiempo, y especialmente este último año, más y más personas se ven inmersas en este medio de entretenimiento. Este crecimiento en la industria tiene sus razones, siendo un par de ellas el amplio catálogo de juegos cuyo precio es cada vez más accesible y también la amplia temática que pueden llegar a abarcar.

A cada jugador le gustan los videojuegos por diferentes motivos. Desde adolescentes que dedican cuatro horas diarias a disparar a otros jugadores por la emoción que supone competir en línea, hasta docentes que emplean treinta minutos de su limitado tiempo juntando caramelos en su teléfono móvil con tal de desbloquear otro nivel y descansar un poco de la reciente jornada laboral.

Este proyecto tiene la intención de describir el proceso técnico y creativo de un videojuego de acción en tercera persona centrado en combates en línea.

1.2 Objetivos

El objetivo de este TFG es desarrollar una primera versión viable de un videojuego multijugador en línea. Una vez terminado el MVP[1], la idea sería distribuirlo de forma gratuita en páginas como Itch.io[2] donde jugadores de diferentes partes del mundo lo puedan probar y expresar sus opiniones acerca del juego.

[1] MVP significa mínimo producto viable
[2] Página web de Itch.io: www.itch.io

CAPÍTULO 2
Estado del arte

Los videojuegos en línea no son ninguna novedad, en el mercado ha habido este tipo de videojuegos desde la década de los 90 con títulos como Doom[1], Ultima Online[2] o Counter-Strike[3]. Poco tuvo que cambiar para que este género conquistara a millones y millones de jugadores por todo el mundo. A día de hoy es inegable el hecho de que la tecnología ha cambiado mucho en los últimos veinte años. Desde el ADSL hasta la fibra óptica, de velocidades como 56Kbps[4] hasta velocidades 300 Mbps[5] o incluso más.

Figura 2.1: Counter-Strike en 1999

[1]Juego de disparos en primera persona creado en 1993
[2]MMORPG creado en 1997 basado en la saga de viodejuegos Ultima
[3]Famoso juego de disparos en primera persona creado por Valve en 1999
[4]Kilobits por segundo.
[5]1 Mbps equivale a 1.000 kilobits por segundo

Eso implica que jugar en línea con personas de todo el globo es mucho más fácil y accesible actualmente que hace veinte años. Además de eso, diariamente se factura una gran cantidad de dinero en la industria de los videojuegos, especialmente cuando se tratan de videojuegos multijugador en línea, basta mirar a juegos como «League of Legends»[6] o Fortnite[7], dos de los videojuego más populares en el mercado, y ambos son «free-to-play»[8].

Para seguir hablando de los videojuegos en línea, primero es importante explicar el concepto de «microtransacciones»:

- Las microtransacciones son el modelo de negocio principal de los videojuegos online free-to-play. Consisten en que el juego es gratuito para todo aquél que quiera jugarlo, y una vez dentro del juego, suelen haber mercados virtuales totalmente **opcionales** donde se venden aspectos, iconos o cualquier tipo de cosmético para los personajes del juego.

Este modelo de negocio es tan popular y tan efectivo que League of Legends recaudó dos mil millones de dólares en el año dos mil diecisiete.

Figura 2.2: Imagen de una partida de League of Legends

Figura 2.3: Diferentes cosméticos para el personaje «Ezreal» de League of Legends

[6] Es un videojuego del género multijugador de arena de batalla en línea y deporte electrónico desarrollado por Riot Games

[7] Fortnite lanzado en 2017 y desarrollado por la empresa Epic Games

[8] Juegos para PC, smartphone y tablet que se ofrece de forma gratuita y que, en principio, se pueden jugar sin coste adicional

2.1 Crítica al estado del arte

En la industria de los videojuegos existen títulos con ideas buenas, mecánicas innovadoras y mucho potencial y, debido a malas decisiones de negocio, no están teniendo el éxito que podrían llegar a tener. Ese es el caso de Ubisoft con For Honor.

For Honor[9] es un videojuego que tuvo mucha popularidad en su fecha de estreno debido a su original mecánica de combate pero que, año tras año, va perdiendo más y más jugadores que deciden probar contenido nuevo. El primer problema del juego está en que no sigue el modelo Free-to-play con microtransacciones que hemos mencionado previamente. De hecho, el precio de este juego suele rondar fácilmente los 30€, salvo cuando hacen rebajas. Personalmente opino que los servidores de For Honor tendrían muchos más jugadores activos a día de hoy si se hubiera adaptado el juego al mercado actual.

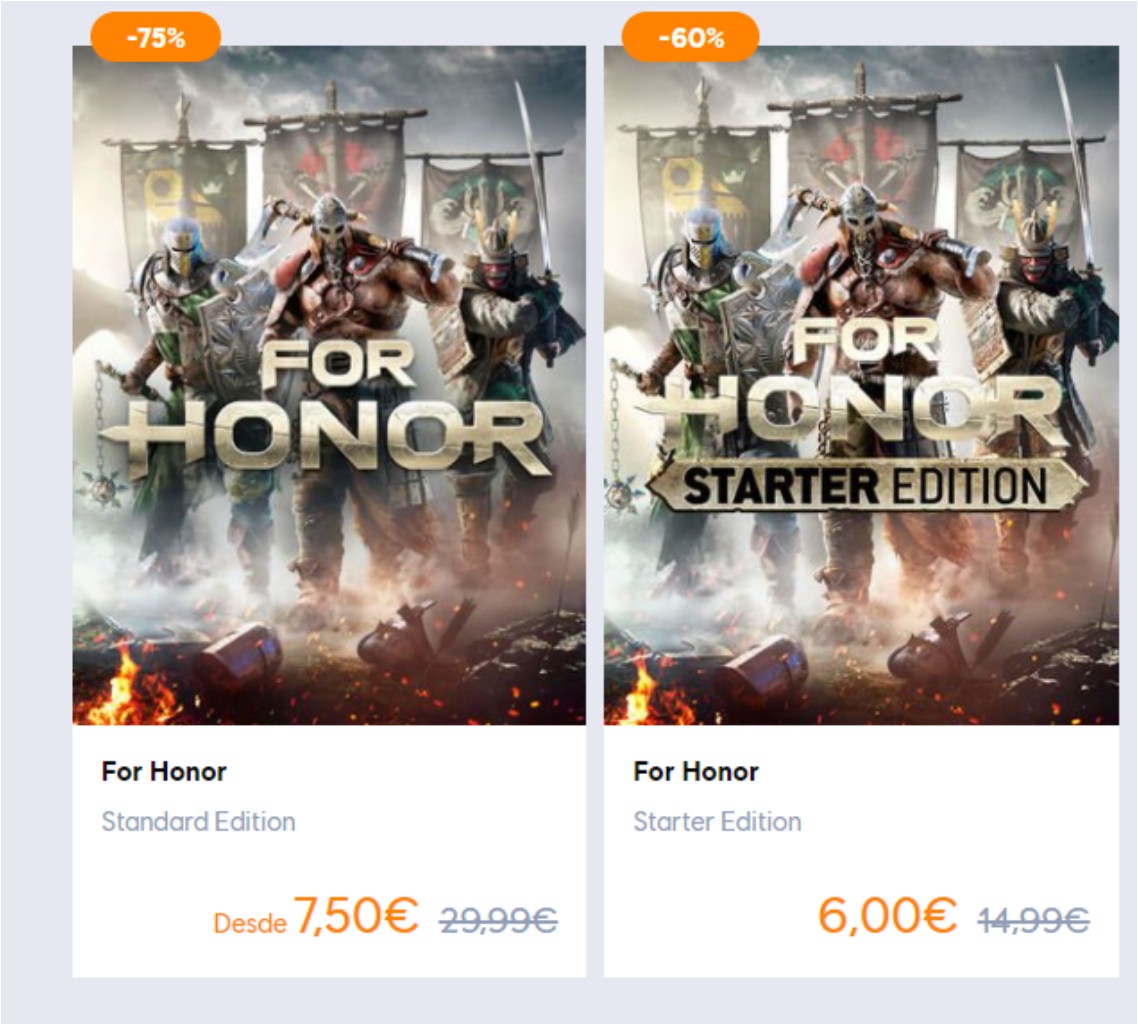


Figura 2.4: Precio de la tienda oficial de Ubisoft

Debido a esto, se ha decidido crear desde cero un videojuego de combate en tercera persona con opción a multijugador en línea usando como mecánica principal el sistema de combate direccional.

[9] Página web For Honor: www.ubisoft.com/es-es/game/for-honor

CAPÍTULO 3
Análisis del problema

Antes de empezar cualquier tipo de trabajo es preciso analizar los requisitos y estudiar las diferentes formas de abordar sus problemas y necesidades. Temas a tener en cuenta son: el tiempo de desarrollo, el presupuesto necesario para llevar a cabo el proyecto, las tecnologías y soluciones que se van a utilizar, entre otras cosas. En este capítulo se expondrá de forma detallada las distintas ideas y soluciones que han habido en el tratamiento de este videojuego.

3.1 Análisis del mercado

A continuación se analizarán los diferentes videojuegos, ideas y mecánicas[1] del mercado que han servido de inspiración para llevar a cabo este proyecto. La idea principal de esta sección es contrastar los puntos y características más importantes de cada juego e implementar una combinación de dichas características en este proyecto.

Las características mencionadas a continuación fueron elegidas según los puntos que se quieren alcanzar en este proyecto.

3.1.1. For Honor - Ubisoft

For Honor, desarrollado por Ubisoft, es la fuente de inspiración principal para la mecánica de combate en tercera persona: el sistema de combate direccional. Este sistema de combate es sencillo: el jugador dispone de un espacio abierto en el que combatir contra un enemigo. El jugador puede fijar su vista en dicho enemigo y elegir en qué dirección atacar. Las opciones son: izquierda, derecha y arriba. Lo interesante de este sistema de combate ocurre cuando un jugador ataca a otro por uno de los tres lados y el otro tiene el arma en esa misma posición. Cuando eso ocurre, el segundo jugador bloquea satisfactoriamente el ataque del primero. En caso contrario, el primer jugador consigue evadir la defensa del segundo llegando a hacerle daño.

Figura 3.1: Sistema de combate direccional implementado por Ubisoft

[1] las mecánicas son las dinámicas que media entre las reglas de juego y la experiencia del jugador

3.1.2. Saga Souls - FromSoftware

Esta saga[2] sirve de inspiración por su gran ambientación y el magnífico trabajo que hace con la atmósfera del juego, llegando a crear mucho misterio, ansiedad y peligro en algunos tramos del juego. El detalle más importante de su sistema de combate es el compromiso del jugador con sus ataques. En títulos como «Bayonetta» o «Genshin Impact» las animaciones pueden ser canceladas para aumentar el daño de un combo o incluso resguardarse de una embestida enemiga. En la saga Souls, en cambio, las animaciones duran su ciclo completo, eso requiere que el jugador piense antes de actuar. Cuando se realiza un ataque, no se puede realizar ninguna otra acción hasta que su animación termine. Eso deja expuesto al jugador y es ahí donde reside la mayor parte de la emoción del juego.

Figura 3.2: Ambientación fantástica de Dark Souls

Figura 3.3: Modo multijugador online de Dark Souls

[2] Página web FromSoftware: www.fromsoftware.jp/ww

3.1.3. Hollow Knight - Team Cherry

Hollow Knight[3] es un videojuego «indie»[4] publicado en 2017 por Team Cherry. Se trata de un «metroidvania»[5] con un estilo artístico muy particular, una dinámica bien equilibrada entre la exploración y los enfrentamientos, y con una ambientación fantástica debido a su apartado musical. Cada zona de Hollow Knight tiene su propia personalidad, su propia paleta de colores y tema musical haciendo así que cada demarcación se sienta única y especial.

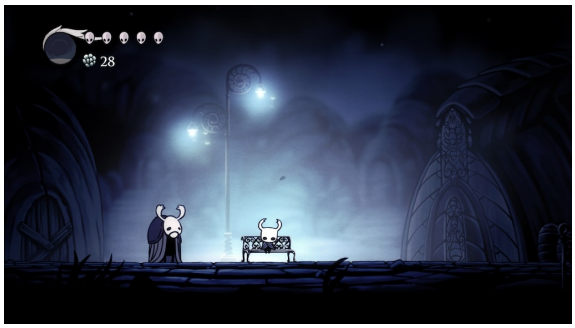


Figura 3.4: Protagonista de Hollow Knight en Bocasucia

3.1.4. Comparación de características

Características/Juego	For Honor	Dark Souls	Hollow Knight
Ambientación consistente	✗	✓	✓
Animaciones no cancelables	✓	✓	✓
Banda sonora inmersiva	✗	✓	✓
3D	✓	✓	✗
Personaje customizable	✗	✓	✗
Sistema de subida de niveles	✗	✓	✗
Árbol de habilidades	✓	✗	✓
Modo historia	✗	✓	✓
Modo multijugador online	✓	✓	✗
Diseño de nivel interconectado	✗	✓	✓
Mecánicas de juegos plataforma	✗	✗	✓
Pocas pantallas de carga	✗	✓	✓
Mundo explorable e interesante	✗	✓	✓
Historia adaptativa	✗	✗	✗
Diálogo activo	✗	✗	✗

Tabla 3.1: Tabla comparativa de los diferentes videojuegos estudiados

Como conclusión, los puntos más importantes para este proyecto son: un **mundo explorable e interesante**, **historia adaptativa con diálogo activo** para el jugador, una **ambientación consistente** junto a un **buen sistema de combates con animaciones que no se pueden cancelar**, y por último **una banda sonora inmersiva en un entorno 3D**. La idea principal de este proyecto es mezclar cada uno de estas características recogidas de diferentes obras y crear una experiencia nueva, visualmente bonita, inmersiva, intensa, interesante y entretenida para sus jugadores.

[3] Página web de Team Cherry: www.teamcherry.com.au
[4] Independiente
[5] Género de videojuego de acción y aventura basado en un concepto de plataformas no lineal

3.2 Motor de videojuego

Un motor de videojuego es un programa software que dispone de diferentes herramientas que facilitan y agilizan el desarrollo de un juego. Estos motores se suelen encargar de muchas tareas complejas y necesarias en juegos. Por ejemplo: sistema de detección de colisiones, motor de físicas, motor de renderizado gráfico (2D y/o 3D), sistema de sonidos, sistema de redes, animaciones y muchas otras cosas.

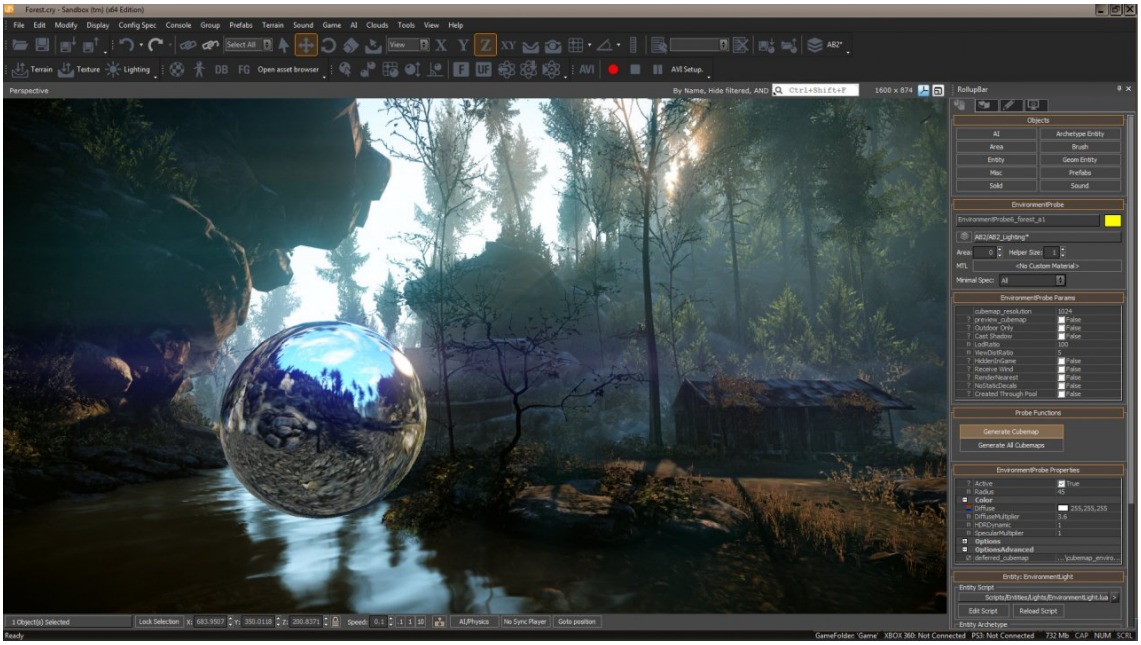


Figura 3.5: Motor de videojuegos CryEngine

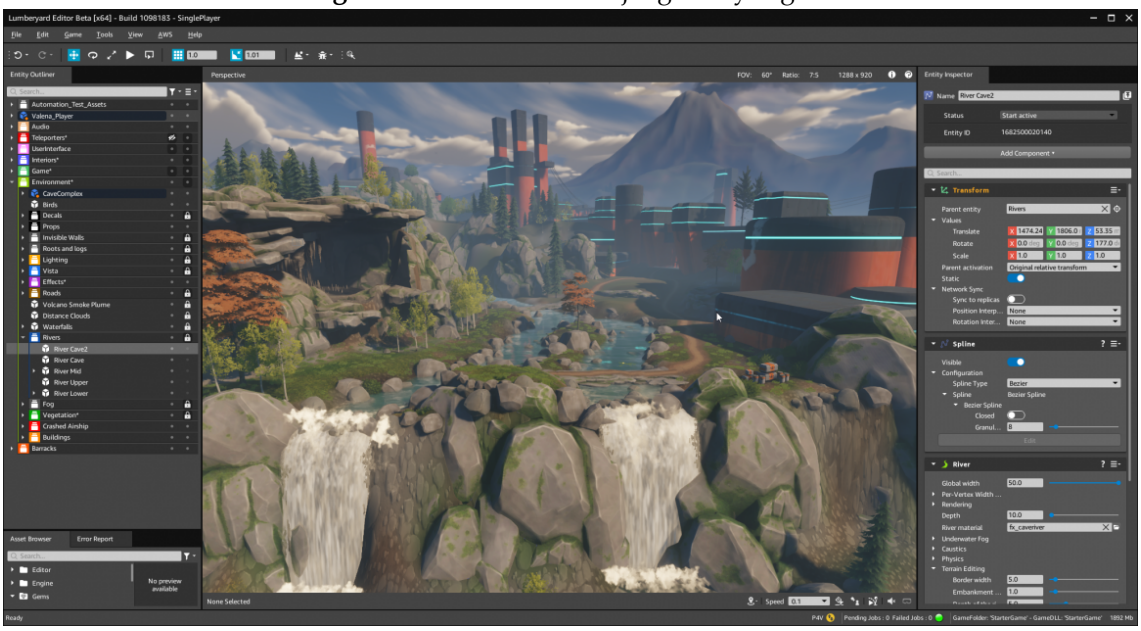


Figura 3.6: Motor de videojuegos Lumberyard de Amazon

Utilizar soluciones ya programadas, diseñadas y probadas por un equipo dedicado, evitando así «reinventar la rueda» y aumentar los costes temporales, es la mayor ventaja que proporciona utilizar un motor de videojuegos existente. Por ese motivo, año tras año, muchas personas deciden empezar en el desarrollo de videojuegos.

3.2.1. Opciones

En este apartado del documento se hablará de las diferentes opciones y particularidades de los motores de videojuegos disponibles en el mercado y según eso se hará una elección para la herramienta que se utilizará en este proyecto.

- **Unreal Engine 4:** se trata de uno de los motores de videojuegos multiplataforma 2D y 3D más conocidos y utilizados, estando en el mercado desde 1998. Muchos títulos «Triple A» fueron hechos usando Unreal Engine. En Unreal se trabaja en «C++» lo cual proporciona a los programadores muchísima libertad y capacidad de optimización al ser un lenguaje de programación muy eficiente. Otro punto a favor de Unreal Engine[6] está en la alta calidad gráfica de los productos finales. Todos los recursos de Megascans[7] son totalmente gratuitos para sus usuarios.

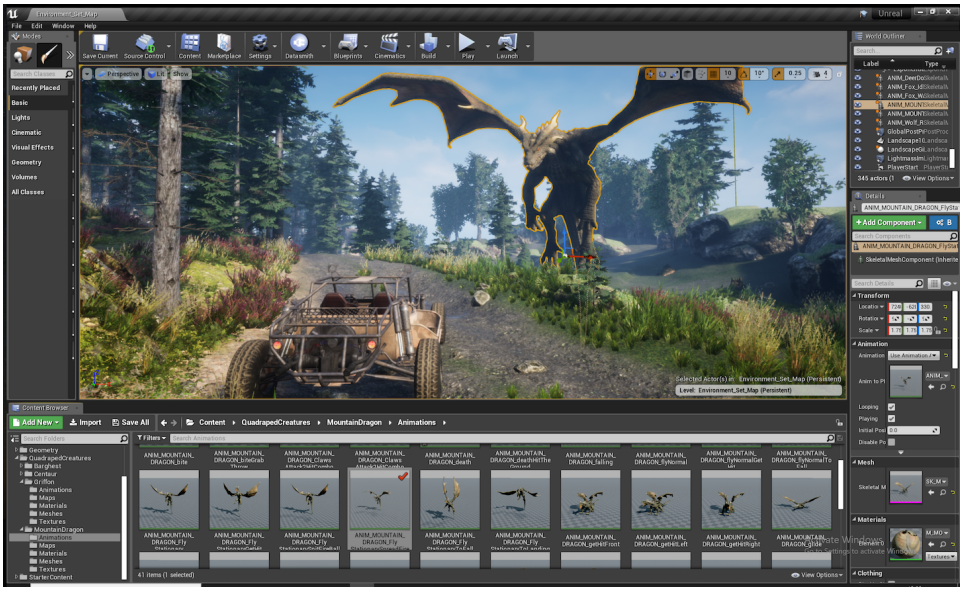


Figura 3.7: Editor de Unreal Engine 4

Figura 3.8: Bosque hecho con recursos escaneados de Megascans

[6] Página web de Unreal Engine: www.unrealengine.com
[7] Megascans es una aplicación de escritorio y una librería online de escaneados 3D de muy alta resolución.

- **Godot:** es la opción de código abierto más popular del mercado, se trata de un motor para juegos 2D y 3D, desarrollado por la comunidad de Godot[8] y publicado con la Licencia MIT. Los proyectos creados se pueden exportar a Windows, OS X, Linux, Android, iOS y HTML5.

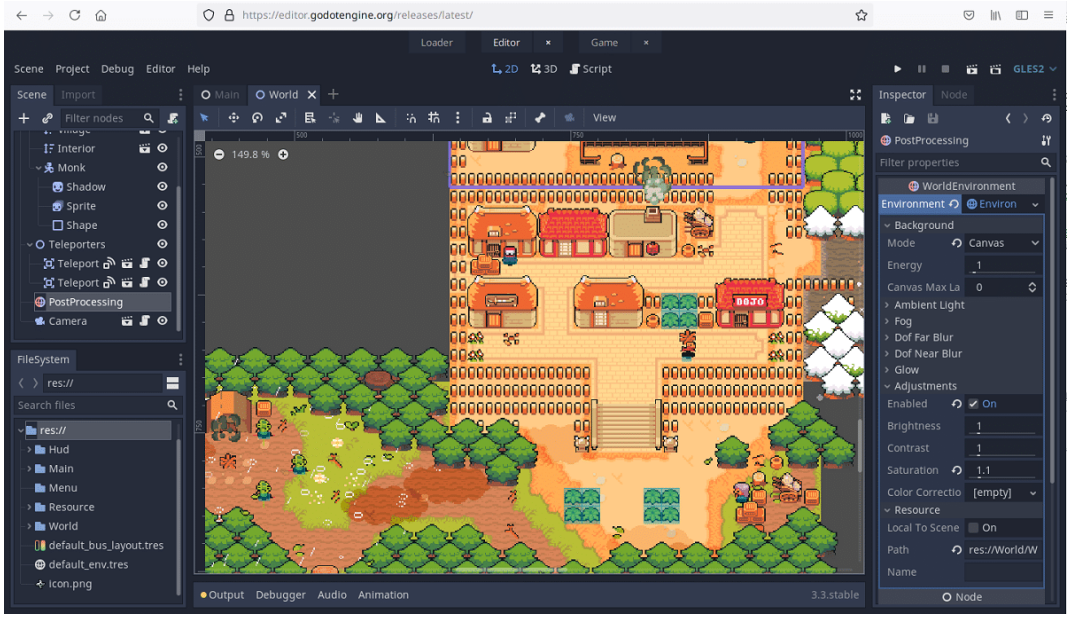


Figura 3.9: Editor de Godot

Figura 3.10: «Garden Path» es un juego hecho con Godot

[8] Página web de Godot: www.godotengine.org

- **Unity:** sin duda se trata de la opción más popular entre los desarrolladores independientes. Es un motor de videojuego multiplataforma creado en 2005 por Unity Technologies[9]. Cuenta con un amplio catálogo de assets[10] gratis y de pago en su tienda online (la Asset Store), mucha documentación y vídeos tutoriales por todo internet, lo cual lo convierte en una buena herramienta para empezar en el desarrollo de videojuegos.

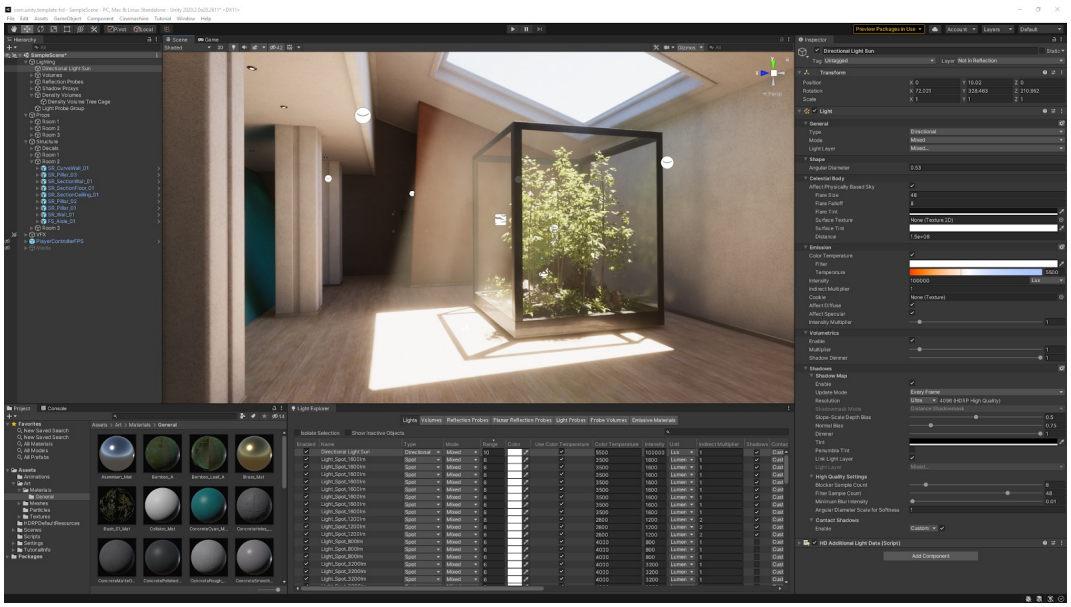


Figura 3.11: Editor de Unity

Figura 3.12: «Ghost of a Tale» es un juego hecho con Unity

[9] Página web de Unity: www.unity.com/es

[10] Son recursos utilizados para hacer videojuegos que pueden ser modelos 3D, música, animaciones y etc.

- **GameMaker Studio 2:** es un motor de videojuegos centrado en el renderizado de gráficos en 2D. A diferencia de los motores previamente mencionados, GameMaker Studio 2 no es del todo gratuita ya que es necesario pagar mensualidades con tal de exportar los proyectos a cualquier tipo de plataforma. La ventaja que tiene sobre su competencia reside en lo fácil que es crear un proyecto teniendo muy pocos o nulo conocimiento de programación.

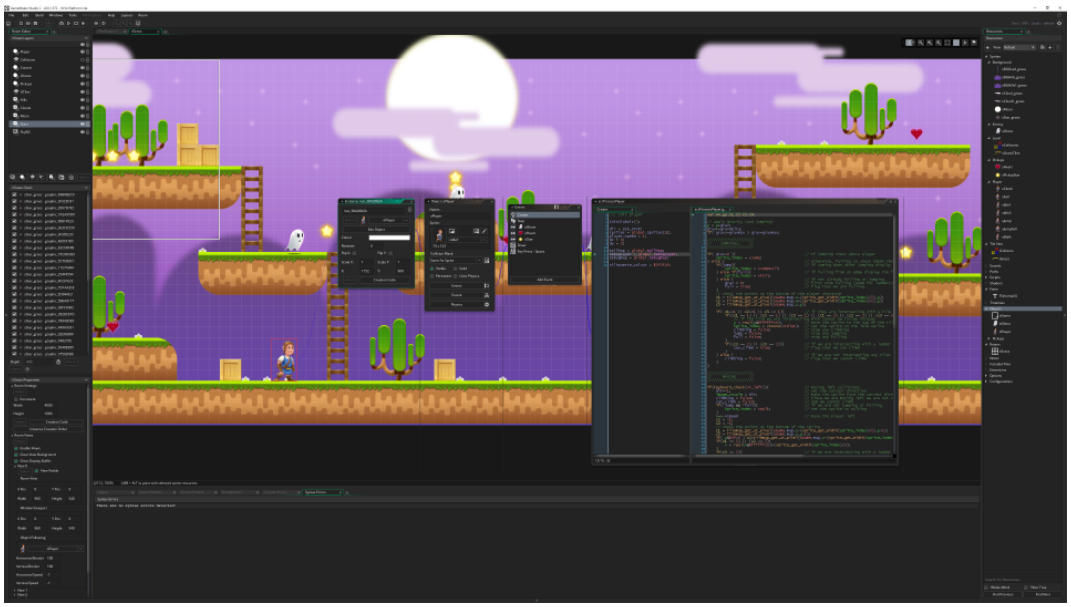


Figura 3.13: Editor GameMaker Studio 2

Figura 3.14: «Hyper Light Drifter» es un juego hecho con GameMaker Studio 2

3.2.2. Comparaciones y decisión final

Comparación de motores de videojuegos				
	Unreal Engine 4	Godot	Unity	GameMaker Studios 2
Open-source	✗	✓	✗	✗
Gratis (o de coste bajo)	✓	✓	✓	✗
Exportación gratuita	✓	✓	✓	✗
2D	✓	✓	✓	✓
3D	✓	✓	✓	✗
Herramienta de animaciones	✓	✓	✓	✓
Interfaz amigable e intuitiva	✗	✗	✗	✓
Beginner Friendly	✗	✓	✗	✓
Multiplataforma gratuito	✓	✓	✓	✗
Ligero	✗	✓	✗	✓
Comunidad extendida	✓	✓	✓	✓
Experiencia previa del equipo	✓	✗	✓	✗

Tabla 3.2: Tabla comparativa de los diferentes motores de videojuegos estudiados

Se ha decidido abordar este proyecto utilizando Unity Engine como motor de videojuegos. Los motivos detrás de esta decisión han sido los siguientes:

- Unity cuenta con la comunidad más extendida de desarrolladores
- El equipo cuenta con experiencia previa utilizando este motor
- Es de muy bajo coste (gratis en caso de haber ganado menos de cien mil dólares en los últimos doce meses)

3.3 Juego en línea

Para construir un multijugador en línea es necesario tener un servidor donde los jugadores se pueden conectar y compartir partida. Este proyecto pretende abarcar este problema de forma de forma sencilla y barata debido al cercano plazo de entrega. Para ello es necesario mirar las posibles opciones.

3.3.1. Opciones

Las posibles herramientas para crear un juego con multijugador en Unity son:

- Photon Unity Networking (PUN) de PhotonEngine: PUN es una solución bien integrada con el motor de Unity y sus diferentes componentes como las animaciones, posición y rotación en el espacio tridimensional.
  - **Ventajas**: PUN está diseñado para ser trabajado con Unity desde el editor añadiendo unos pocos componentes y creando el código oportuno es posible crear y sincronizar las partidas con mucha facilidad.
  - **Desventajas**: No es de código abierto y la opción de «hosting»[11] gratuita no es muy potente. Se puede montar un servidor propio utilizando PUN pero la única opción disponible es hacerlo en Windows Server 2012.

Figura 3.15: Logo de Photon Unity Networking

[11] Se trata de alojamiento web para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía web

- Nakama de Heroic Labs: Nakama se trata de un servidor de videojuegos escalable y de código abierto.

  - **Ventajas**: Es totalmente de código abierto, lo cual implica que los únicos gastos del multijugador en línea vendrá del hosting del servidor. Se puede utilizar en proyectos que no formen parte de Unity.
  - **Desventajas**: Los precios del hosting por parte del equipo de Heroic Labs es muy elevado.

Figura 3.16: Logo de Nakama

- Mirror Networking de Vis2k: consiste en un proyecto de código abierto para crear servidores multijugadores online para el motor de videojuegos Unity.

  - **Ventajas**: Está pensado y diseñado para ser compatible con Unity. Fue hecho para soportar un gran número de jugadores en línea, típico de un MMORPG[12]
  - **Desventajas**: No cuenta con ninguna solución de hosting, ni siquiera de pago. En caso de utilizar Mirror, es necesario alojar y mantener el servidor obligatoriamente.

Figura 3.17: Logo de Mirror Networking

[12]Videojuegos de rol que permite interactuar con otras personas que juegan de forma simultánea.

3.3.2. Solución para el multijugador online

Comparación de frameworks multijugador online			
	PUN	Nakama	Mirror Networking
Open-source	✗	✓	✓
Versión gratuita	✓	✓	✓
Dispone de servicio de alojamiento web	✓	✓	✗
Se puede alojar en servidores propios	✓	✓	✓
Buena documentación	✓	✓	✗
Beginner friendly	✓	✓	✓
RPC	✓	✓	✓
Diseñado para Unity	✓	✓	✓
Ligero	✗	✓	✗
UDP	✓	✓	✓
TCP	✓	✓	✓
Multiplataforma	✓	✓	✓

Tabla 3.3: Tabla comparativa de los diferentes frameworks para hacer el multijugador en línea

Según las características comparadas con la información obtenida en la página web oficial de cada proveedor, se ha elegido Nakama como solución para este proyecto.

3.3.3. Servicio de alojamiento

A día de hoy montar complejas infraestructuras con tal de disponer de servidores web no es la única opción disponible. El alojamiento en la nube[13] permite un rápido despliegue de servicios omitiendo cuestiones complejas y agotadoras como el mantenimiento, escalabilidad, disponibilidad e incluso ubicación física de las máquinas. Existen servicios que se encargan de todas estas dificultades y complicaciones a un precio muy asequible. A continuación se hablará de las posibles opciones que se han tenido en cuenta en la producción del backend[14] del multijugador online de Melilac.

- Microsoft Azure: la solución de máquinas virtuales de Azure es un servicio de computación en la nube creado por Microsoft cuyo objetivo es el de montar, desplegar y administrar aplicaciones y servicios web.

Figura 3.18: Microsoft Azure

[13]Los servicios en la nube son infraestructuras que alojan los proveedores externos y que se ponen a disposición de los usuarios a través de Internet.

[14]Es la parte de un servicio típicamente web que se encarga de que toda la lógica de funcionamiento

- Amazon Web Services (AWS): es una colección de servicios web (como EC2[15], Lambda[16], Route53[17], entre otras) que en conjunto forman una plataforma de computación en la nube.

Figura 3.19: AWS EC2

Comparación de servicios en nube		
	AWS	MS Azure
Open-source	✗	✗
Gratis por tiempo limitado	✗	✓
Gratis según tipo de instancia	✓	✗
Buena documentación	✓	✓
Más eficiente con el uso de CPU	✗	✓
Mejor control de la memoria	✓	✗

Tabla 3.4: Tabla comparativa de los diferentes servicios en cloud computing

Y por último es necesario elegir la relación entre cliente y servidor.

- Server-Authoritative Multiplayer: Es un término que se utiliza cuando la autoridad y control sobre la partida y todo lo que ocurre lo dictamina el servidor.
  - **Ventajas:** es muy complicado hacer trampas. Todos las personas conectadas juegan a la partida tal y como lo dice el servidor.
  - **Inconvenientes:** la implementación es complicada y es necesario implementar un algoritmo de predicción de estados para reducir la latencia[18] de las partidas.
- Client-Authoritative Multiplayer: Término aplicado al funcionamiento de las partidas multijugador online en que la autoridad y control sobre la partida está repartida entre todos los clientes conectados a la sesión.
  - **Ventajas:** rápida implementación, no es necesario implementar ningún algoritmo de predicción de estados. A la larga acaba siendo más barato ya que el servidor los costes computacionales.
  - **Inconvenientes:** es muy fácil hacer trampas ya que cada cliente es el encargado de modificar, almacenar y distribuir los datos de su instancia.

Para un juego en tiempo real donde cada fotograma de animación importa, es muy recomendable usar la estructura donde la autoridad de la partida la tiene el servidor. Eso

[15] EC2 permite a los usuarios alquilar computadores virtuales en los cuales pueden ejecutar sus propias aplicaciones.
[16] AWS Lambda es una plataforma informática sin servidor impulsada por eventos
[17] Amazon Route 53 es un servicio de sistema de nombres de dominio escalable y de alta disponibilidad.
[18] la latencia de red es la suma de retardos temporales dentro de una red.

evita muchos problemas como las trampas y la sincronización errónea de las partidas. El problema de esta solución al multijugador reside en que es mucho más compleja la implementación de dicho servidor, es necesario implementar un algoritmo de predicción de estados y eventualmente acaba teniendo costes más elevados.

La solución ideal para un juego como Melilac sería el despliegue de un servidor autoritario que controle los valores oportunos de cada jugador como la posición, rotación, animaciones y etc. Por cuestión de eficiencia se ha elegido el modelo más sencillo y barato «client-authoritative» donde cada cliente se encarga de actualizar los valores de su partida y distribuirlo por la red para que los demás clientes puedan sincronizarse.

En resumidas cuentas se trata de una red P2P centralizado[19] que se encarga de distribuir la información entre todos los clientes. La mayor desventaja del funcionamiento de este servidor está en que las partidas en línea no funcionan correctamente en caso de que alguno de los dos jugadores tenga una mala conexión o un ordenador poco potente.

Figura 3.20: Estructura P2P centralizado

[19] Arquitectura monolítica en la que todas la comunicación se hacen a través de un único servidor que sirve de punto de enlace entre dos o más nodos.

3.3.4. Decisión final

Para este proyecto se utilizará el motor de videojuegos Unity debido a la experiencia, familiaridad y la comodidad que se cuenta con esta herramienta. Respecto al multijugador online se ha optado por Nakama de Heroic Labs ya que es de código abierto y totalmente gratis. Para alojar dicho servidor en cuestión se ha elegido el servicio en nube de Amazon Web Services.

3.4 Plataforma destino

La plataforma destino para este proyecto es Windows 10 ya que es el PC[20] es la plataforma más fácil de exportar y probar, ya que el equipo de desarrollo no necesitaría invertir en diferentes consolas.

3.5 Análisis DAFO

Se trata de una herramienta de gestión que proporciona información necesaria para la implementación de acciones y medidas correctivas, y para el desarrollo de proyectos de mejora facilitando así las decisiones estratégicas de mercado. El nombre DAFO, corresponde a los cuatro elementos que se evalúan en el desarrollo del análisis: las debilidades, amenazas, fortalezas y oportunidades.

Debilidades	**Fortalezas**
El equipo de desarrollo es muy pequeño y, sin presupuesto, a largo plazo con una dedicación completa el proyecto es prácticamente insostenible	Al tratarse de un producto independiente, el precio de venta es bastante inferior al estándar de sesenta euros, lo cual nos proporciona una ventaja en el mercado.
Al empezar de cero y sin ningún tipo de fama o reconocimiento, es complicado dar a conocer el producto y hacer que alcance un gran número de personas	El juego aporta ideas nuevas con mucha experimentación, cosa que no suelen hacer los grandes títulos. Eso haría que destacáramos en el mercado.

Tabla 3.5: Tabla del análisis interno DAFO

Amenazas	**Oportunidad**
El mundo de los videojuegos está compuesto por grandes titanes con un gran equipo de desarrollo detrás y mucho presupuesto. Ser reconocido cuando existen compañías que sacan mucho contenido en muy poco tiempo es muy difícil ya que el producto podría acabar siendo olvidado o ignorado junto a muchos otros títulos	Debido a la situación vivida a lo largo de este curso con el confinamiento y la pandemia, muchas personas se han visto con más tiempo libre y han decidido invertirlo jugando juegos nuevos. De hecho, Steam ha reportado un gran aumento en el número de usuarios activos este dos mil veinte a causa del Covid-19.

Tabla 3.6: Tabla del análisis externo DAFO

[20] «Personal computer», ordenador personal. Habitualmente es un equipo de sobremesa.

CAPÍTULO 4
Desarrollo

4.1 Relación del trabajo desarrollado con los estudios cursados

La relación principal de este proyecto con los estudios cursados está en el hecho de que, desarrollar un videojuego es desarrollar un producto software, y, como todo desarrollo software de calidad, es necesario hacer una planificación del trabajo eficiente, realista, ordenada, siguiendo pautas y haciendo uso de buenas prácticas.

A continuación se expondrá la planificación de las tareas, los objetivos iniciales del proyecto y cómo ha evolucionado al largo del tiempo.

Una de las primeras necesidades del proyecto es establecer la metodología del proceso de desarrollo. Se ha decidido utilizar la metodología ágil **Scrum** para el desarrollo de este proyecto por varios motivos:

- Sprints: separar la cadencia de trabajo en sprints asegura que al final de cada uno haya un incremento significativo en el desarrollo del producto.
- Tiempo de desarrollo: cuando el equipo requiere de un MVP y el tiempo es un factor muy presente, utilizar Scrum en lugar de Kanban es una buena idea ya que se separan las unidades de trabajo más importantes para el MVP en los primeros sprints. Dichas unidades de trabajo no se pueden cambiar a otro sprint, asegurando así la integridad del MVP.
- Kanban es de flujo continuo: este flujo de trabajo tiene sentido para desarrollos continuos sin restricciones de tiempos y mayor libertad con las tareas o unidades de trabajo, esta metodología es muy útil para proyectos de mantenimiento pero no tanto en proyectos de desarrollo a corto plazo como es el caso de Melilac.

4.2 Organización del trabajo. Scrum.

Se disponía de aproximadamente sesenta y tres días para realizar el desarrollo de este proyecto. Teniendo eso en cuenta se ha decidido dividir el trabajo en tres sprints de veintiún días. Se ha utilizado la herramienta Trello[1] y eventualmente se hizo el cambio a ClickUp[2] como herramienta de organización y el control de tiempos, sprints y unidades de trabajo.

Otro factor a tener en cuenta es el tamaño del proyecto. La idea original era hacer un juego con una campaña principal que incluyera más de diez horas de contenido obligatorio, muchas misiones secundarias para aumentar la duración de la campaña principal con contenido diferente y entretenido y, por último, un modo multijugador online donde los jugadores pudieran competir el uno contra el otro.

La parte más compleja del este proyecto sin duda es la campaña principal, llegando a una estimación de unas mil doscientas horas de trabajo aproximadamente. Por ese motivo se ha decidido omitir esta parte del videojuego en la realización de este trabajo de fin de grado.

4.2.1. Estableciendo roles

Ya que el equipo está formado por un solo desarrollador y ese mismo desarrollador es el Product Owner, la asignación de roles no es nada compleja.

4.2.2. Sprint 1

El primer sprint del proyecto empieza el **01/06** y termina en el **22/06**. Ambos días incluidos. El objetivo principal de este sprint era crear un «PlayableCharacter»[3]. Eso incluye:

- Personaje 3D con animaciones y un sistema de movimiento que le permite moverse por el entorno.

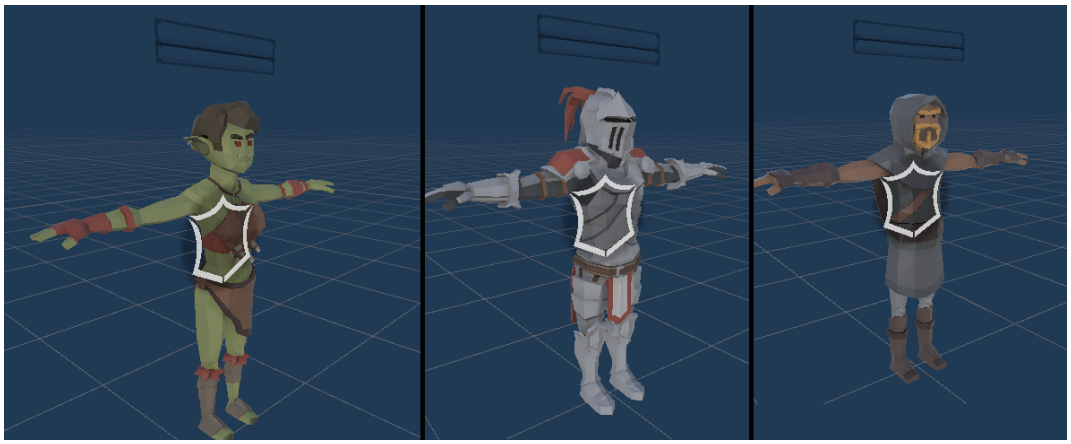


Figura 4.1: Modelo de los personajes

[1] Página web de Trello: www.trello.com
[2] Página web de ClickUp: www.clickup.com
[3] Personaje que se puede mover y que interactúa con el entorno

- Los tres estados principales de los jugadores: explorando, combatiendo, muerto y sus animaciones correspondientes

  El estado «Explorando» es el primer estado en el que se puede encontrar el PlayableCharacter. Es el estado que más libertad proporciona. El jugador puede controlar libremente la cámara y moverse por todo el terreno. Es bueno para observar zonas y conocer el entorno.

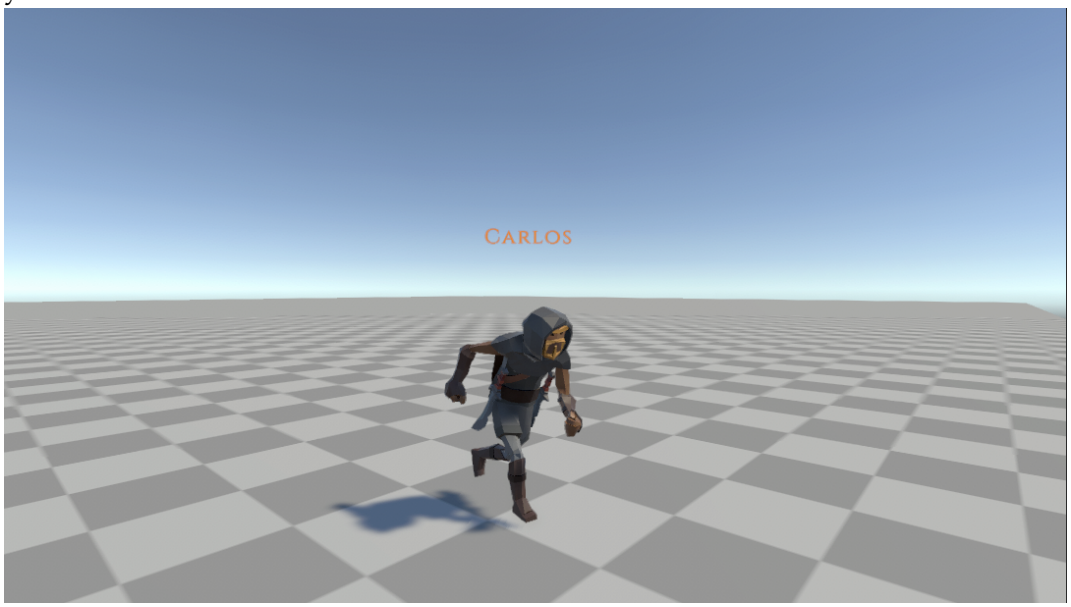


Figura 4.2: Estado A: Explorando

El estado «Combatiendo» es el segundo estado en el que se puede encontrar el PlayableCharacter. Es el estado que permite el combate entre PlayableCharacters. Desde este estado se tiene acceso al combate direccional, esquivas, ataques y bloqueos.

Figura 4.3: Estado B: Combatiendo

El estado «Combatiendo» es el segundo estado en el que se puede encontrar el PlayableCharacter. Es el estado que permite el combate entre PlayableCharacters. Desde este estado se tiene acceso al combate direccional, esquivas, ataques y blo-

queos.

Figura 4.4: Estado C: Muerto

- Animaciones para cada clase de los PlayableCharacters (caballero, ladrón y bárbaro)

  Todas las clases comparten la misma estructura de animaciones:

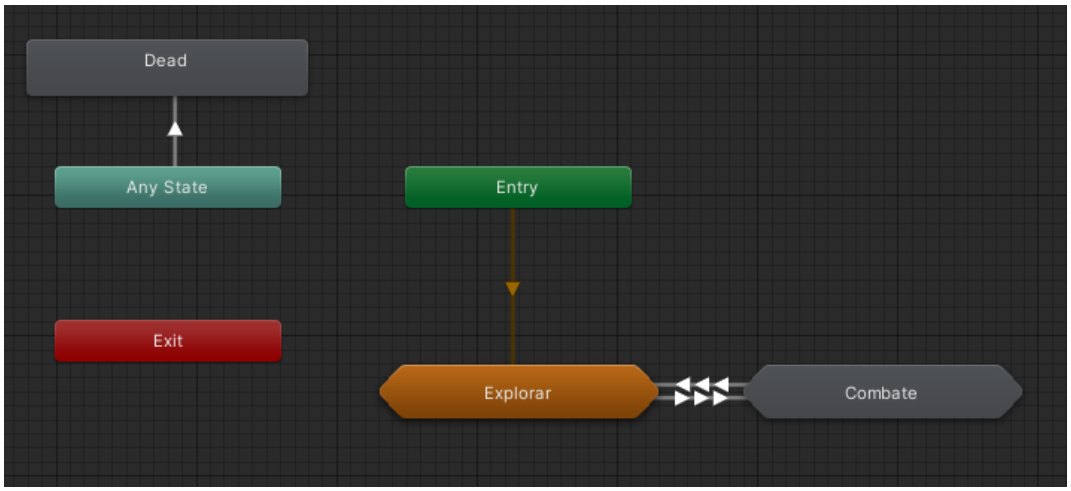


Figura 4.5: Animator de todas las clases

Internamente están compuestas por máquinas de estado que, en función de los valores de entrada aportados por el jugador y su salud total, ejecuta unas animaciones u otras.

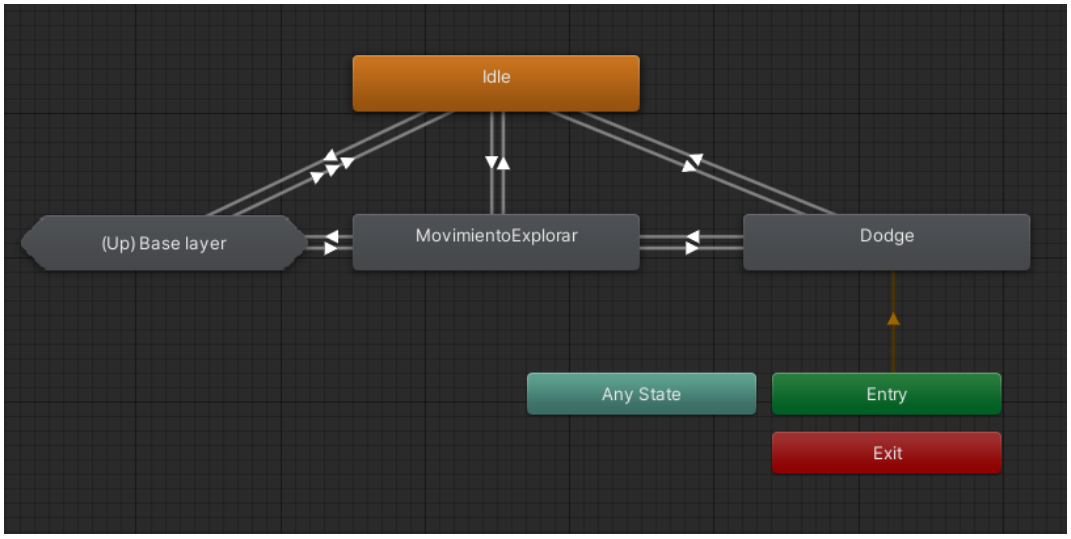


Figura 4.6: Animaciones para el estado de explorar

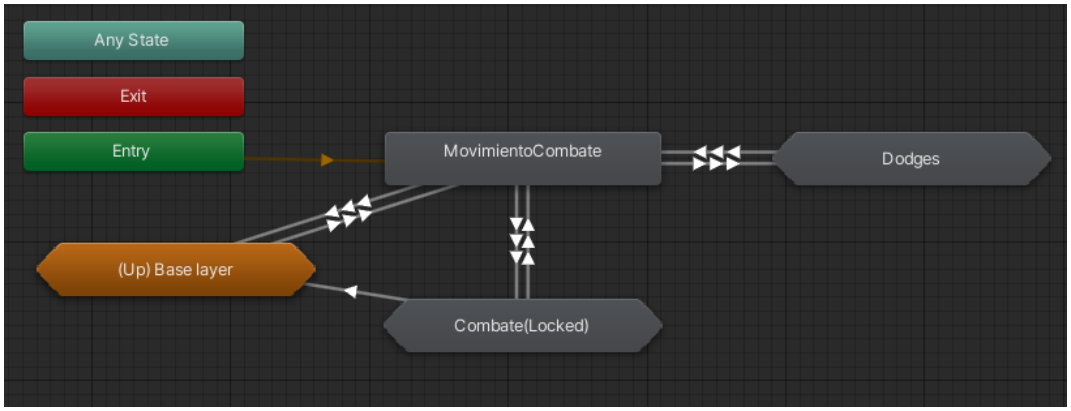


Figura 4.7: Animaciones para el estado Combatiendo A

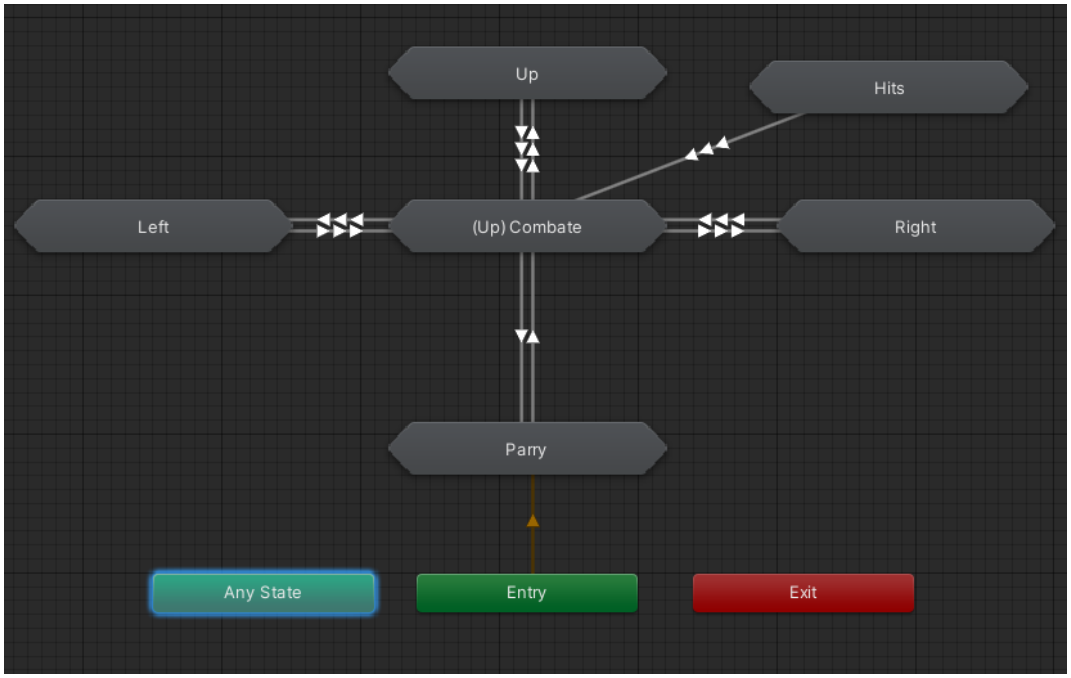


Figura 4.8: Animaciones para el estado Combatiendo B

- Sistema de control de cámaras en tercera persona que facilita al jugador el movimiento de la vista por el entorno. Para eso se ha utilizado el componente «Cinemachine» de Unity. Se trata de una solución fácil e inteligente para el control de cámaras.

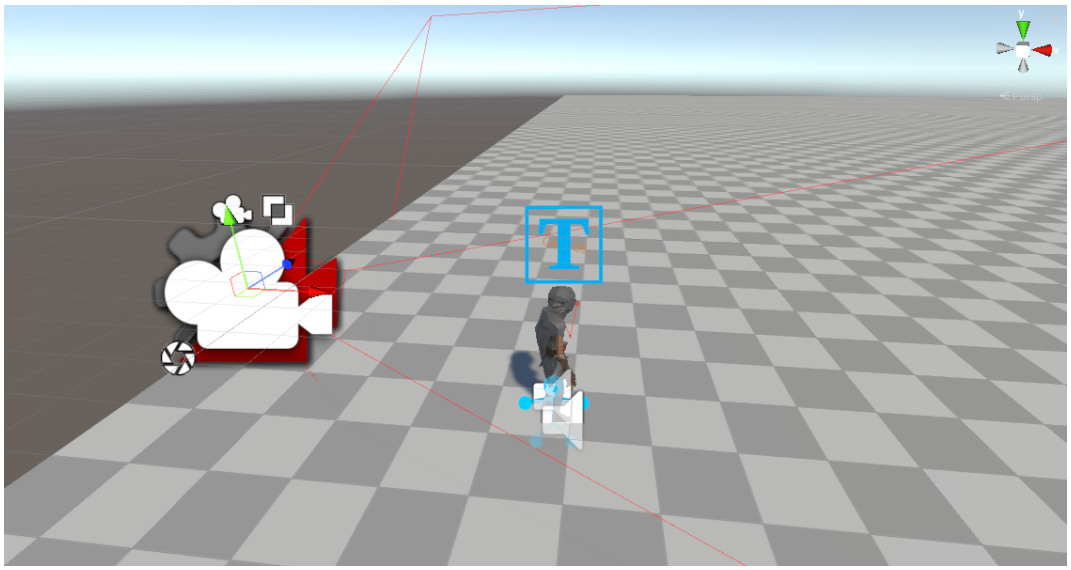


Figura 4.9: Cámara virtual que sigue al jugador

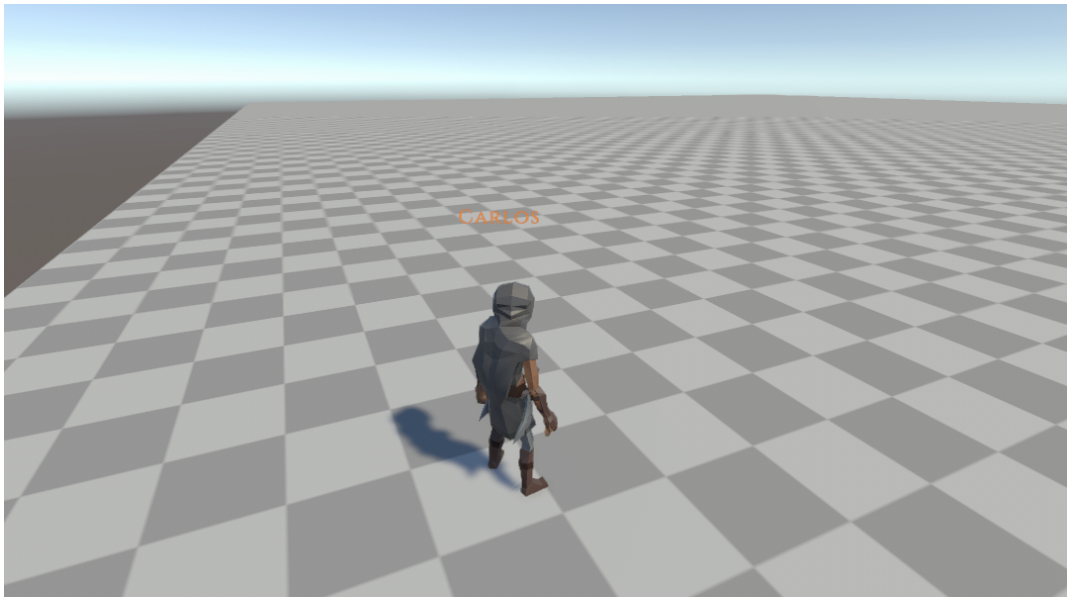


Figura 4.10: Lo que renderiza la cámara desde esa posición

Esta es una sección del diagrama de Gantt generado por la plataforma ClickUp, en él podemos ver las diferentes unidades de trabajo que se han ido haciendo al largo del tiempo.

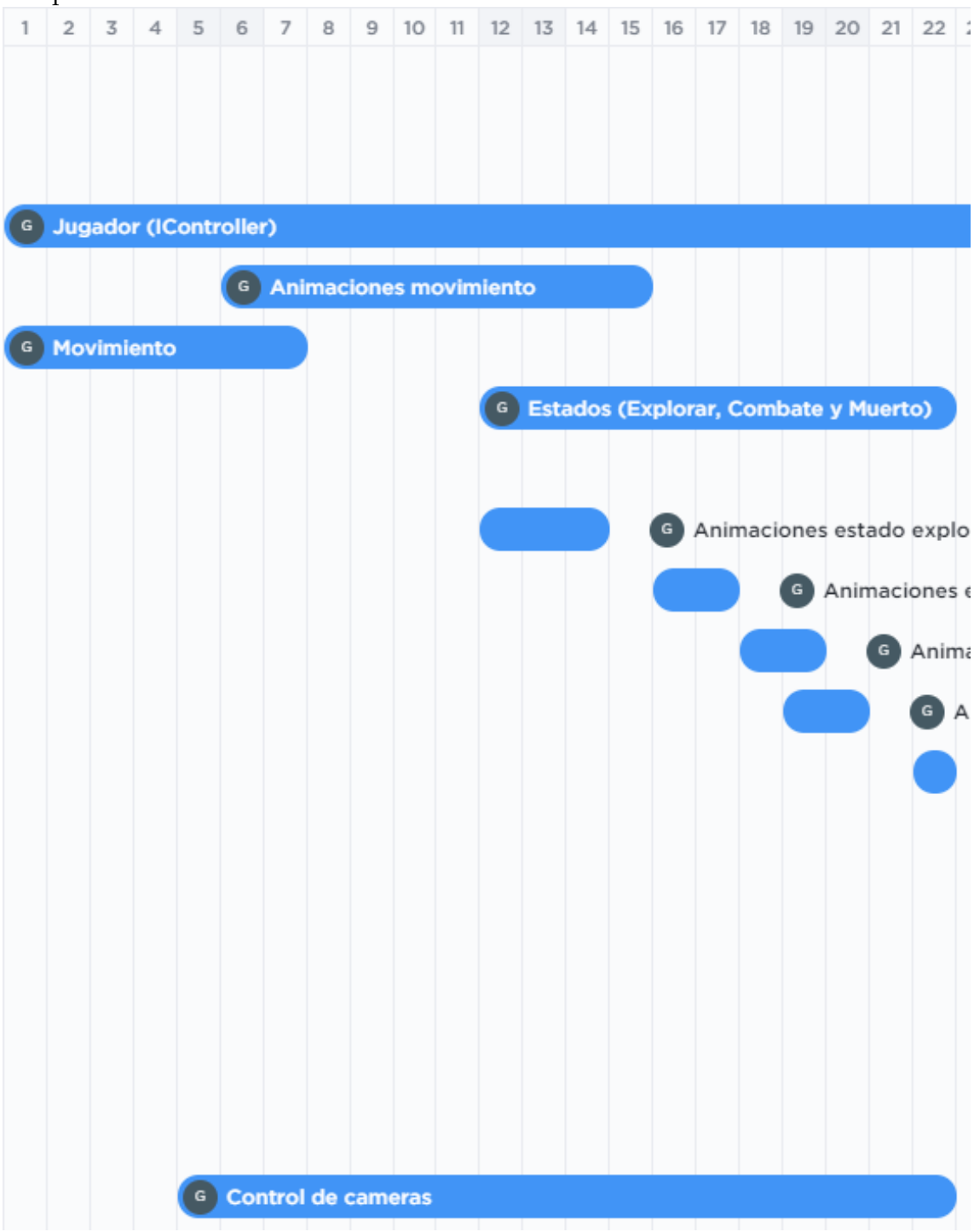


Figura 4.11: Diagrama Gantt para el Sprint 1

4.2.3. Sprint 2

El segundo sprint del proyecto empieza el **23/06** y termina en el **14/07**. El objetivo de este sprint es implementar la parte más importante de este MVP: hacer que el juego sea agradable de jugar y que las partidas se puedan crear y alojar en la nube.

- Sistema de combate direccional que afecta a otros PlayableCharacters (armas que hacen daño y PlayableCharacters que reaccionan al daño)
- Sistema de feedbacks que mejoran la jugabilidad del juego y da a entender al jugador de que ha ocurrido algo (se ha recibido daño, ha conseguido derrotar a su oponente...)
- Instancia de servidores en EC2 con backend de Nakama para el multijugador online. Lógica para restaurar y cancelar el matchmaking.
- Envío de datos al servidor de Nakama que se propaga a todos los clientes conectados a la partida.

Desarrollo del servidor con Nakama en AWS

El primer paso para brindar el proyecto de un multijugador online es desplegar un servidor en la nube utilizando las tecnologías comentadas previamente.

- **Instancia en AWS:** Tras iniciar sesión en la consola de AWS[4] tenemos que buscar la solución que mejor convenga para las características de este proyecto. Cada cliente planea enviar aproximadamente sesenta peticiones de posición por segundo más un número variable de peticiones de control (inputs de teclado y ratón). Una vez recibidas dichas peticiones, el servidor se encarga de redistribuirlas a todos los demás clientes.

  Teniendo en cuenta dichas necesidades del proyecto, se ha decidido utilizar AWS EC2 como servicio en la nube. El tipo de instancia utilizada ha sido un «t3a.micro» con dos vCPUs[5] y un GiB de memoria principal.

  Seleccionamos EC2, elegimos la imagen de Ubuntu Server 20.04 LTS para la máquina virtual, configuramos los grupos de seguridad (abrimos los puertos de entrada)

[4] aws.amazon.com

[5] Núcleo virtual que se asigna a una máquina virtual o a un núcleo de procesador físico si el servidor no está particionado para máquinas virtuales.

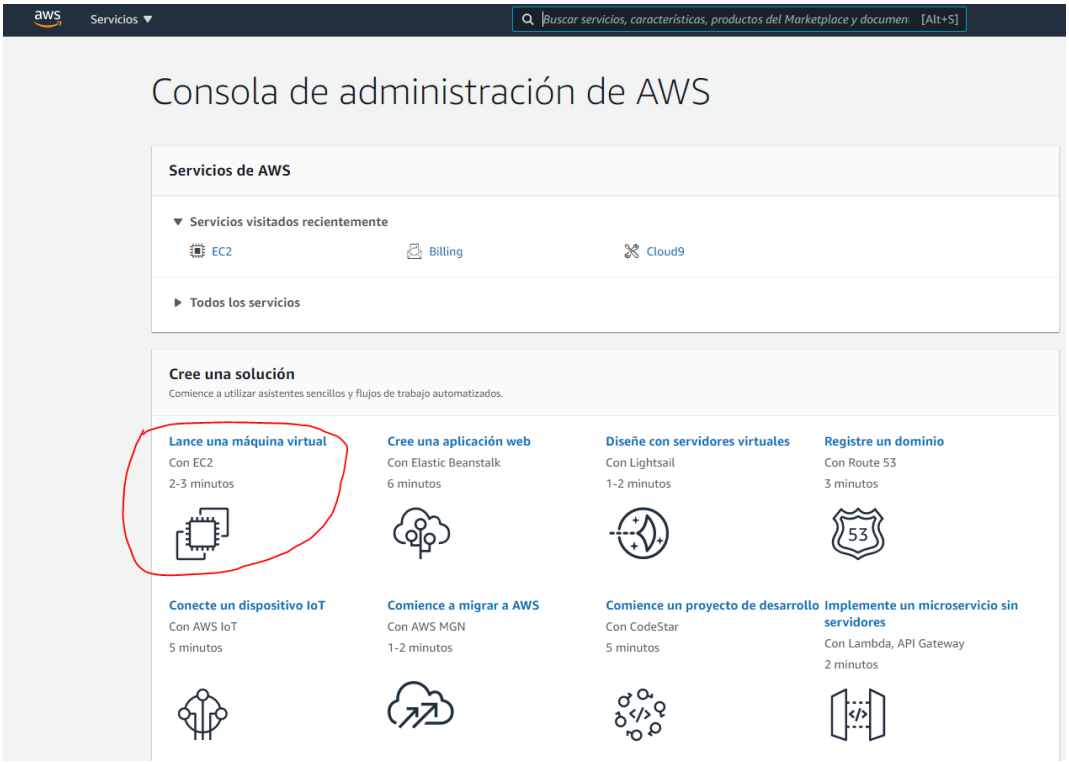


Figura 4.12: Máquinas virtuales de la solución EC2

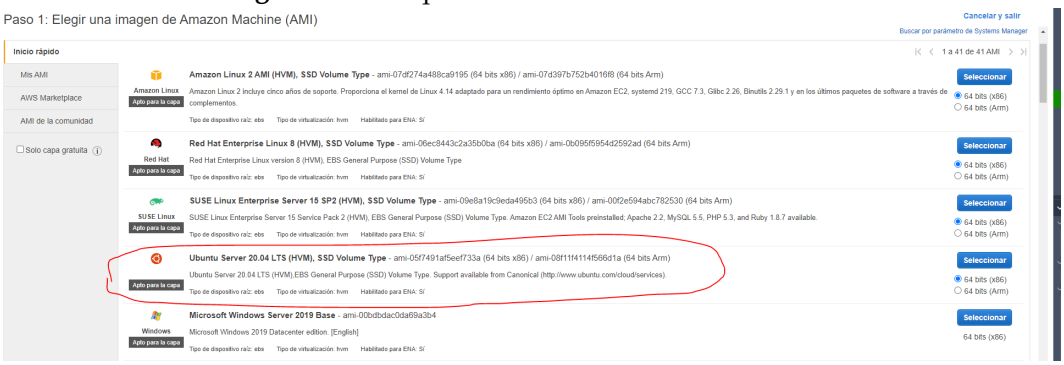


Figura 4.13: AMI de Ubuntu Server 20.04

Intervalo de p...	Protocolo	Origen	Grupos de seguridad
22	TCP	0.0.0.0/0	chemchu-security-group
22	TCP	::/0	chemchu-security-group
7349 - 7351	TCP	0.0.0.0/0	chemchu-security-group
7349 - 7351	TCP	::/0	chemchu-security-group
27017	TCP	0.0.0.0/0	chemchu-security-group
27017	TCP	::/0	chemchu-security-group

Figura 4.14: Puertos abiertos para el funcionamiento del servidor

Una vez realizados estos pasos, la instancia de EC2 ya estaría en ejecución y lista para ser configurada.

- **Nakama Server:** A continuación es necesario desplegar el servidor de Nakama en la instancia de EC2 para que pueda atender las peticiones entrantes. Por suerte Heroic

Labs ha facilitado dicho trabajo para todos aquellos que lo necesiten mediante la herramienta Docker[6].

Se necesita conectar a la instancia de EC2, mediante ssh o con la propia consola proporcionada por Amazon, e instalar Docker y docker-compose, y desplegar los diferentes contenedores. Para ello, se ha necesitado escribir las siguientes instrucciones en la consola.

Instalación de Docker y docker-compose

```
$ apt install docker docker-compose -y
```

Se crea la carpeta donde el servidor estará ubicado

```
$ mkdir NakamaServer
```

Por último, es necesario crear un documento de texto, dentro de NakamaServer, llamado «docker-compose.yml» en el que están todas las dependencias necesarias para levantar los diferentes contenedores.

Contenido del docker-compose.yml

```
    version: '3'
services:
  cockroachdb:
    image: cockroachdb/cockroach:latest-v20.2
    command: start-single-node --insecure --store=attrs=ssd,path=/var/lib
        /cockroach/
    restart: "no"
    volumes:
      - data:/var/lib/cockroach
    expose:
      - "8080"
      - "26257"
    ports:
      - "26257:26257"
      - "8080:8080"
  nakama:
    image: heroiclabs/nakama:3.2.1
    entrypoint:
      - "/bin/sh"
      - "-ecx"
      - >
        /nakama/nakama migrate up --database.address root@cockroachdb
            :26257 &&
        exec /nakama/nakama --name nakama1 --database.address
            root@cockroachdb:26257 --logger.level DEBUG --session.
            token_expiry_sec 7200 --metrics.prometheus_port 9100
    restart: "no"
    links:
      - "cockroachdb:db"
    depends_on:
      - cockroachdb
      - prometheus
    volumes:
      - ./:/nakama/data
    expose:
      - "7349"
      - "7350"
      - "7351"
      - "9100"
```

[6]Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software

```
    ports:
      - "7349:7349"
      - "7350:7350"
      - "7351:7351"
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:7350/"]
      interval: 10s
      timeout: 5s
      retries: 5
  prometheus:
    image: prom/prometheus
    entrypoint: /bin/sh -c
    command: |
      'sh -s <<EOF
        cat > ./prometheus.yml <<EON
      global:
        scrape_interval:     15s
        evaluation_interval: 15s
      scrape_configs:
        - job_name: prometheus
          static_configs:
          - targets: ['localhost:9090']
        - job_name: nakama
          metrics_path: /
          static_configs:
          - targets: ['nakama:9100']
      EON
      prometheus --config.file=./prometheus.yml
      EOF'
    ports:
      - '9090:9090'
volumes:
  data:
```

Y por último solo falta iniciar los contenedores.

Inicialización de NakamaServer

```
$ docker-compose up -d
```

Para comprobar que todo ha funcionado correctamente, se escribe en la consola Comando docker ps para ver los contenedores en funcionamiento.

```
$ docker ps
```

Y así queda comprobado el despliegue de Nakama Server mediante Docker, con sus tres contenedores en funcionamiento:

```
[ec2-user@ip-                ~]$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
9d4e39a2f16c   heroiclabs/nakama:3.2.1              "/bin/sh -ecx '/naka…"   8 weeks ago    Up 8 days (healthy)
b489d8cf1e05   prom/prometheus                      "/bin/sh -c 'sh -s <…"   8 weeks ago    Up 8 days
b8db36b51679   cockroachdb/cockroach:latest-v20.2    "/cockroach/cockroac…"   8 weeks ago    Up 3 days
```

Figura 4.15: Contenedores inicializados

Y el diagrama de Gantt correspondiente.

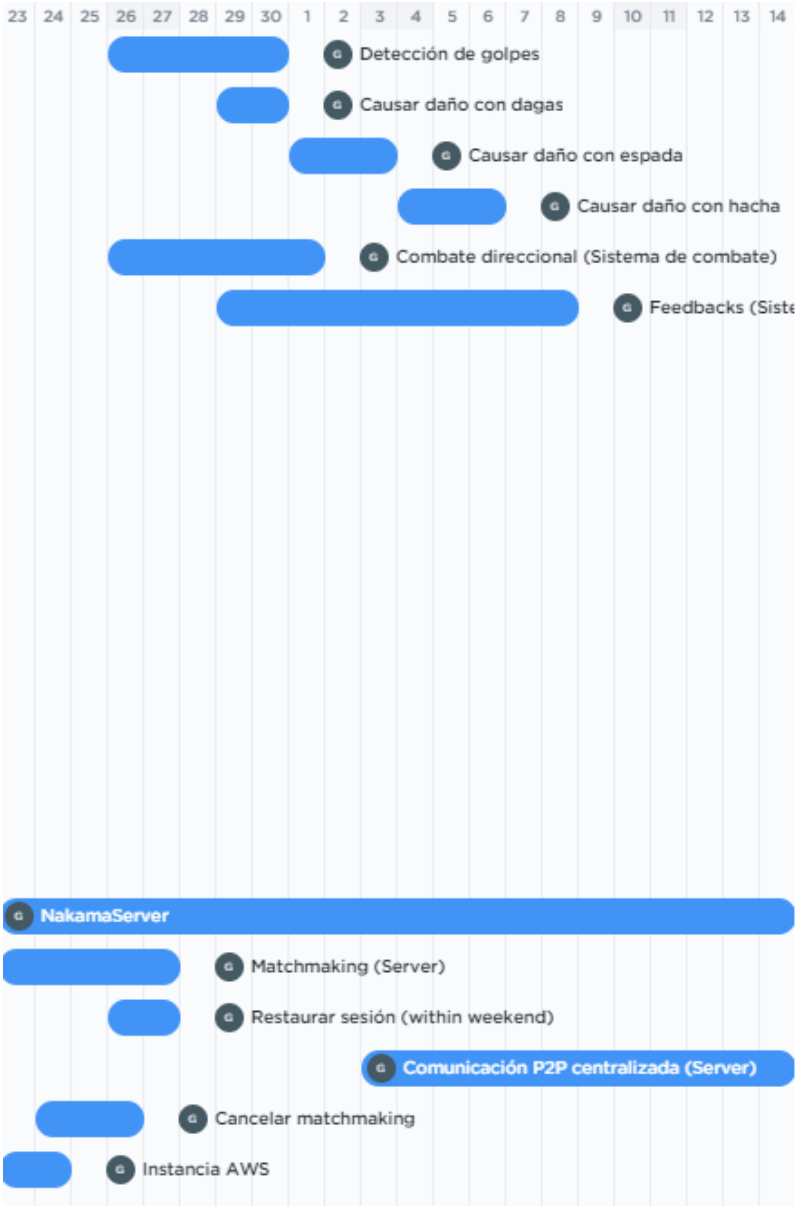


Figura 4.16: Diagrama Gantt para el Sprint 2

4.2.4. Sprint 3

El tercer y último sprint del proyecto empieza el **15/07** y termina en el **25/08**. Con una pausa desde los días 25/07 hasta el 14/08 por vacaciones.

Para este sprint se quiso realizar la parte menos técnica del desarrollo y centrarse especialmente en la presentación y el acabado del producto. Los puntos más importantes de este sprint han sido:

- La interfaz de usuario junto a los menús principales, menús de pausa e incluso las pantallas de carga.

Figura 4.17: Menú principal del juego

- El diseño de niveles para la Arena de los Ya-muertos.

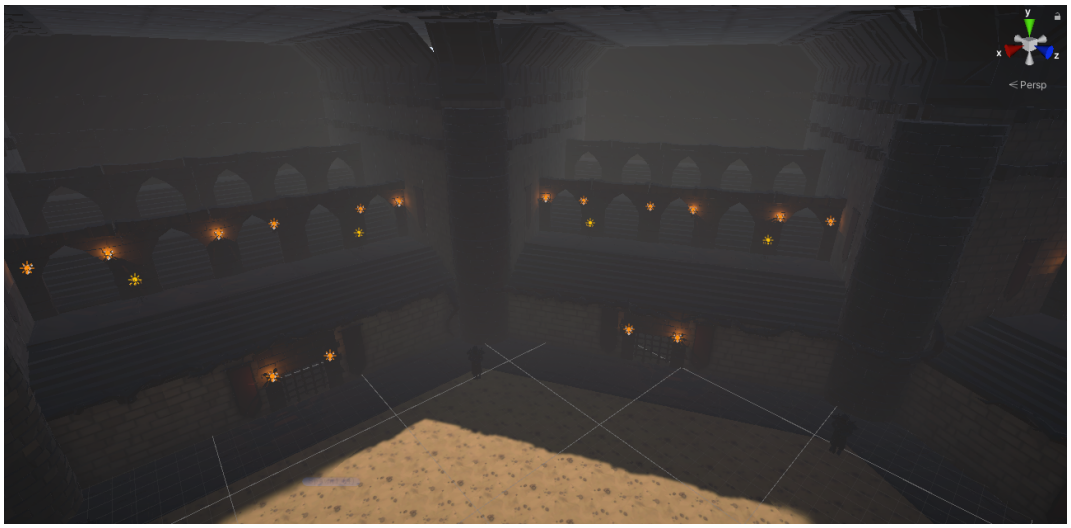


Figura 4.18: la Arena de los Ya-muertos visto desde arriba

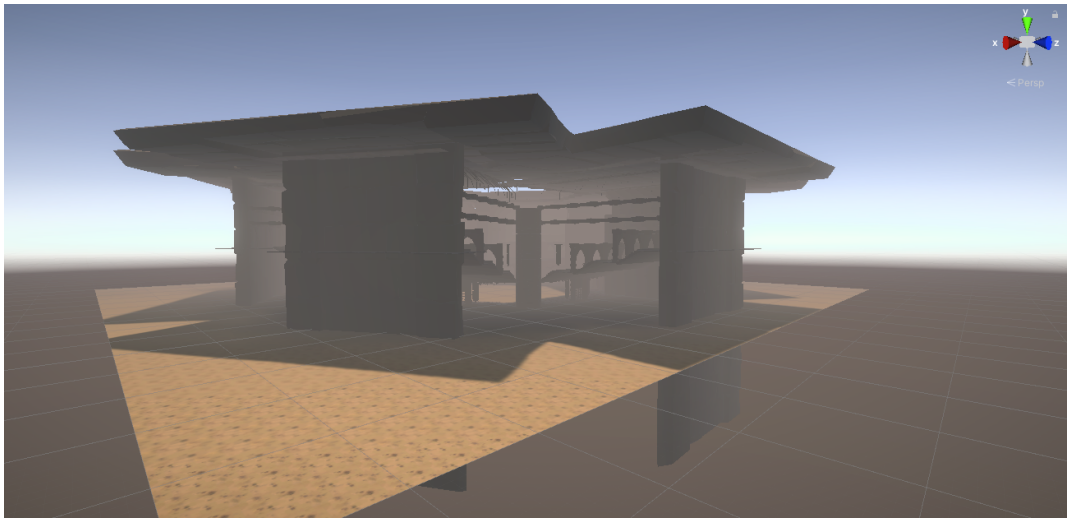


Figura 4.19: la Arena de los Ya-muertos visto desde fuera

- El diseño del bosque del menú principal.

Figura 4.20: El bosque del menú principal

- Los efectos de sonidos y la música en todo el proyecto.

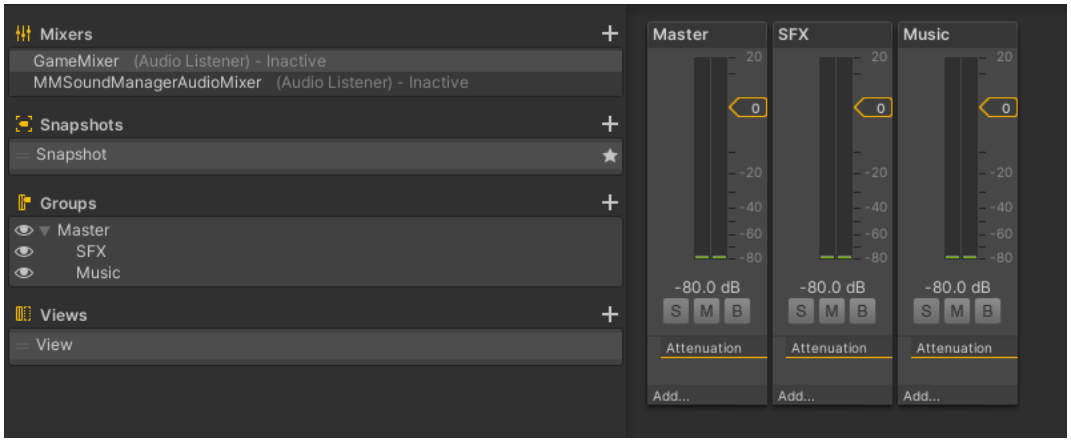


Figura 4.21: AudioMixer encargado de todos los clips de sonido del juego

- «Post-processing»[7] para mejorar la apariencia, la ambientación y la calidad gráfica del juego.

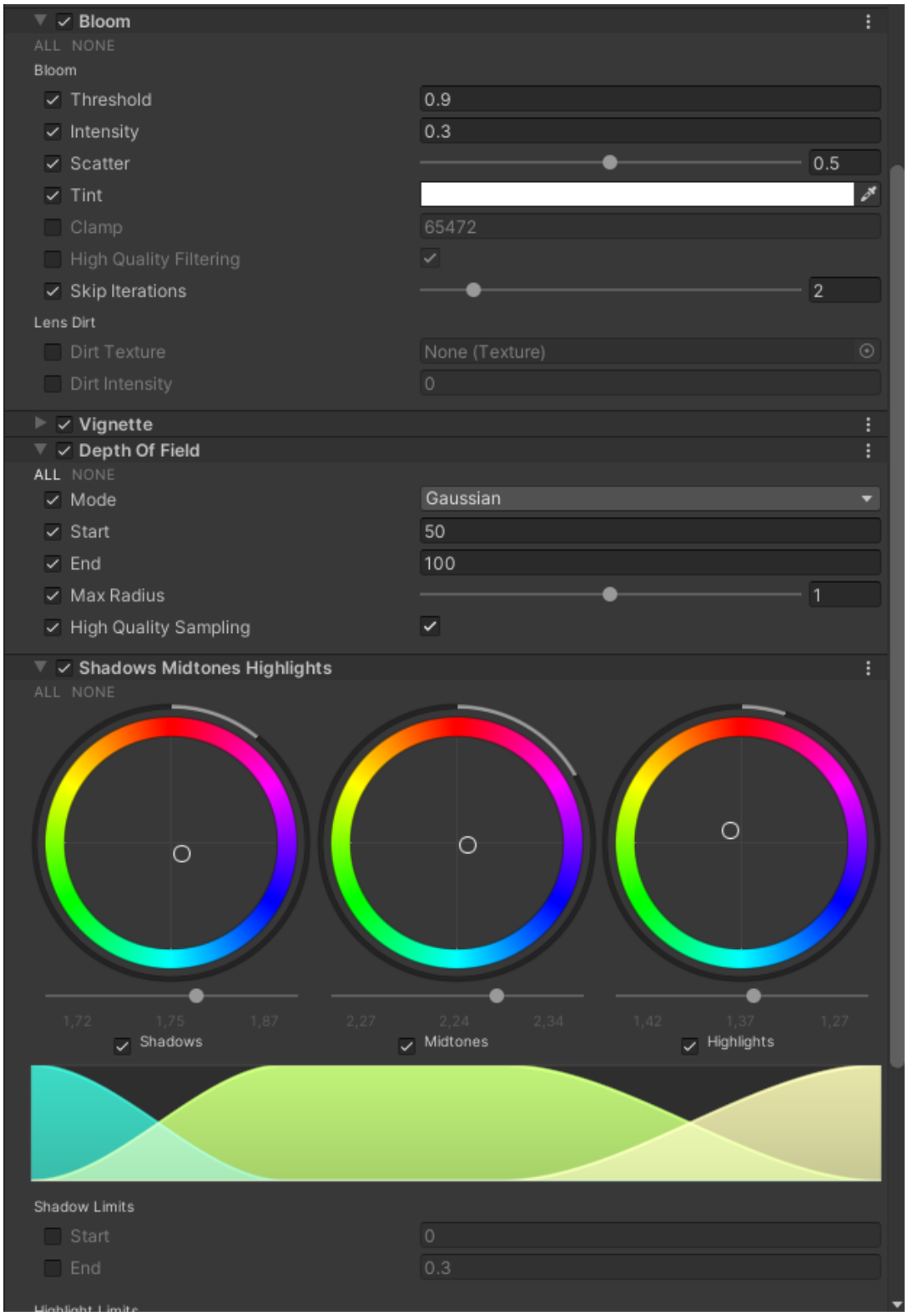


Figura 4.22: Efectos de procesamiento de imágenes aplicado en el proyecto

[7] Retoque de imagen que se realiza después del primer render que se graba en un buffer de la memoria.

Y el diagrama de Gantt correspondiente. Se puede apreciar un parón en la mitad del sprint. Esto se debe a motivos ajenos al desarrollo del producto.

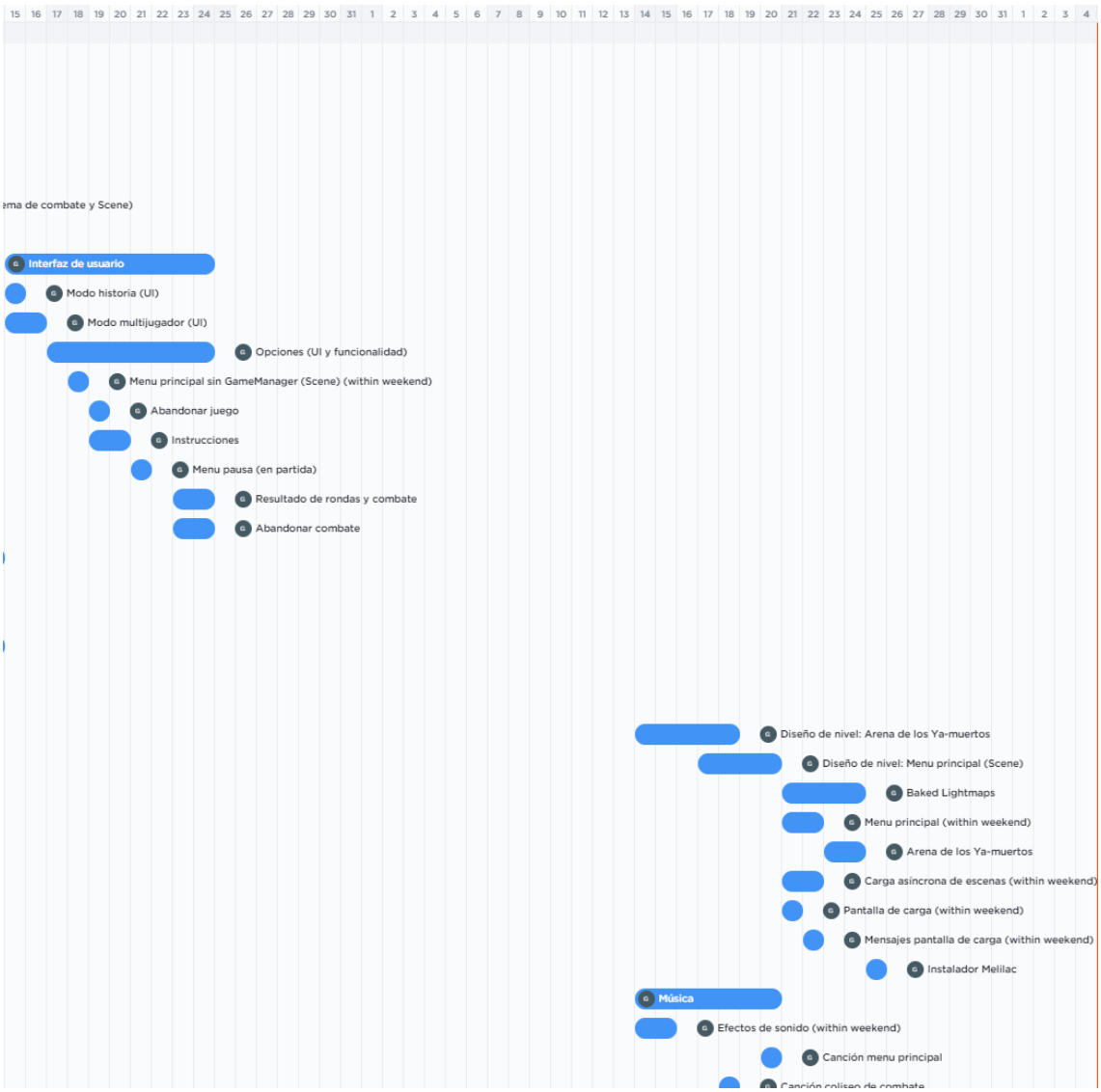


Figura 4.23: Diagrama Gantt para el Sprint 3

4.3 Estado final

A continuación se hablará del estado final del proyecto, los objetivos alcanzados y los flecos que han surgido en el desarrollo.

El tablero Kanban de abajo muestra el estado final del proyecto. Como se puede observar, muchas unidades de trabajo se han quedado en el backlog. Dichas unidades de trabajo hacen referencia a funcionalidades que se deben de implementar en la campaña principal del juego. En el backlog también se encuentran características que se pueden implementar en el modo multijugador para aumentar la complejidad del sistema de combate y mejorar la experiencia de los jugadores.

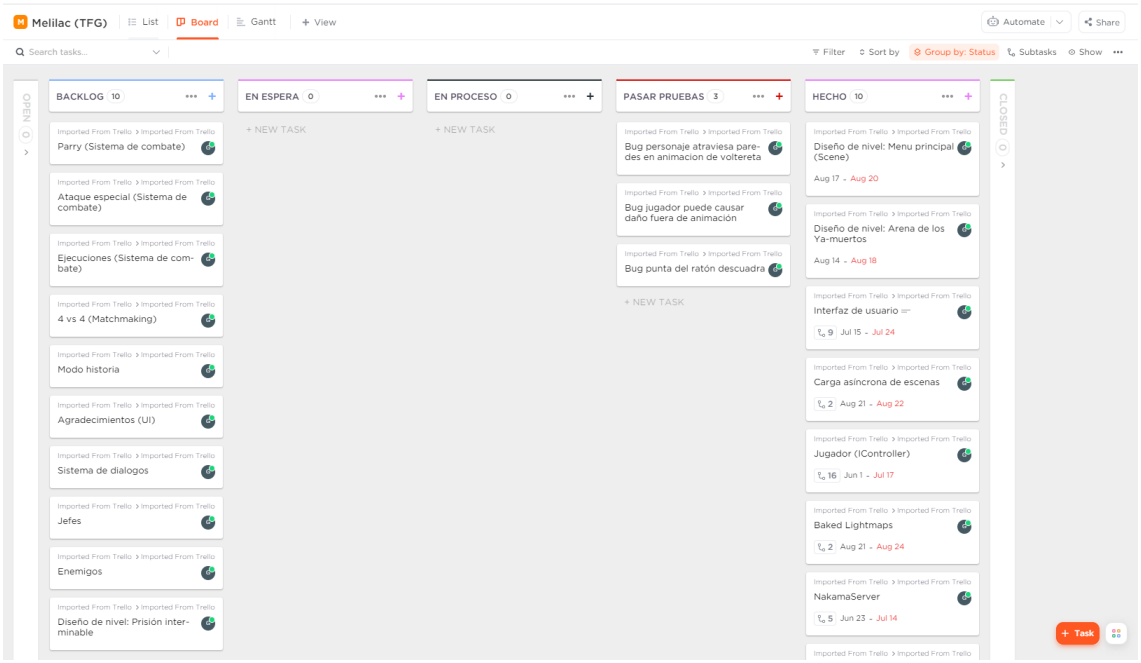


Figura 4.24: Tablero Kanban Melilac

4.3.1. Objetivos alcanzados

- Sprint 1:

Objetivos alcanzados	
	Resultado
Personaje jugable y movible	✓
La cámara rota adecuadamente alrededor del personaje	✓
Las animaciones funcionan correctamente	✓
El PlayableCharacter puede cambiar a cualquiera de los tres estados	✓
El PlayableCharacter interactúa correctamente con el entorno	✗

Tabla 4.1: Tabla de resultados Sprint 1

El PlayableCharacter no interactúa correctamente con el entorno debido a un problema con las colisiones del personaje. Aunque ocurran pocas veces, el PlayableCharacter atraviesa el suelo cuando muere y también puede atravesar paredes cuando realiza una voltereta.

- Sprint 2

Objetivos alcanzados	
	Resultado
El jugador puede dañar a un oponente	✓
El jugador puede recibir daño de un oponente	✓
El jugador bloquea los ataques correctamenete	✓
Los feedbacks funcionan correctamente	✓
El jugador puede buscar una partida	✓
El jugador puede cancelar la búsqueda de una partida	✓
El jugador puede cambiar la dirección de sus ataques	✓

Tabla 4.2: Tabla de resultados Sprint 2

Debido a la buena documentación y una gran cantidad de tiempo invertido en resolver problemas, la conexión con el servidor de Nakama funciona correctamente, las partidas se crean y se descartan adecuadamente y los jugadores puedes disfrutar de un combate de tres rondas.

- Sprint 3

Objetivos alcanzados	
	Resultado
La interfaz de usuario es agradable y combina con la estética del juego	✓
La interfaz de usuario funciona correctamente	✓
El botón de modo historia clarifica que este modo no está disponible	✓
Se pueden ajustar la resolución, calidad y volumen del juego	✓
Se puede poner en pantalla completa o en modo ventana	✓
Se puede salir del videojuego	✓
Se puede combatir en la Arena de los Ya-muertos	✓
Las partidas tienen un menú de pausa	✓
En el menú de pausa se puede ajustar la configuración del juego	✗
En el menú de pausa se puede reanudar la partida	✓
En el menú de pausa se puede abandonar la partida	✓
Las partidas se cargan asíncronamente	✓
El Menú principal y la Arena ahorran recursos al tener el mapa de luces precalculado	✓
El juego dispone de música y efectos de sonido	✓
El jugador puede ver su barra de salud y resistencia, además de su nombre	✓

Tabla 4.3: Tabla de resultados Sprint 3

En el último sprint se ha podido cumplir con casi todos las tareas. El juego dispone ya de un sistema de interfaz de usuario simple pero efectiva que en ningún momento confunde al jugador. Lo único que no se ha implementado es el menú de opciones (calidad de gráficos, resolución...) en el menú de pausa de la partida online.

4.3.2. Estructura de clases

Las diferentes clases e interfaces del proyecto se han separado en diferentes grupos. Dichos grupos son:

- PlayerCore:
  Las clases que se encuentran en este grupo son clases encargadas de todo el funcionamiento de los PlayableCharacters. Eso implica la detección de todos los valores de entrada por teclado y ratón, actualización fotograma a fotograma de los valores de posición, la salud total del personaje, su resistencia, hacia donde mira y etc.

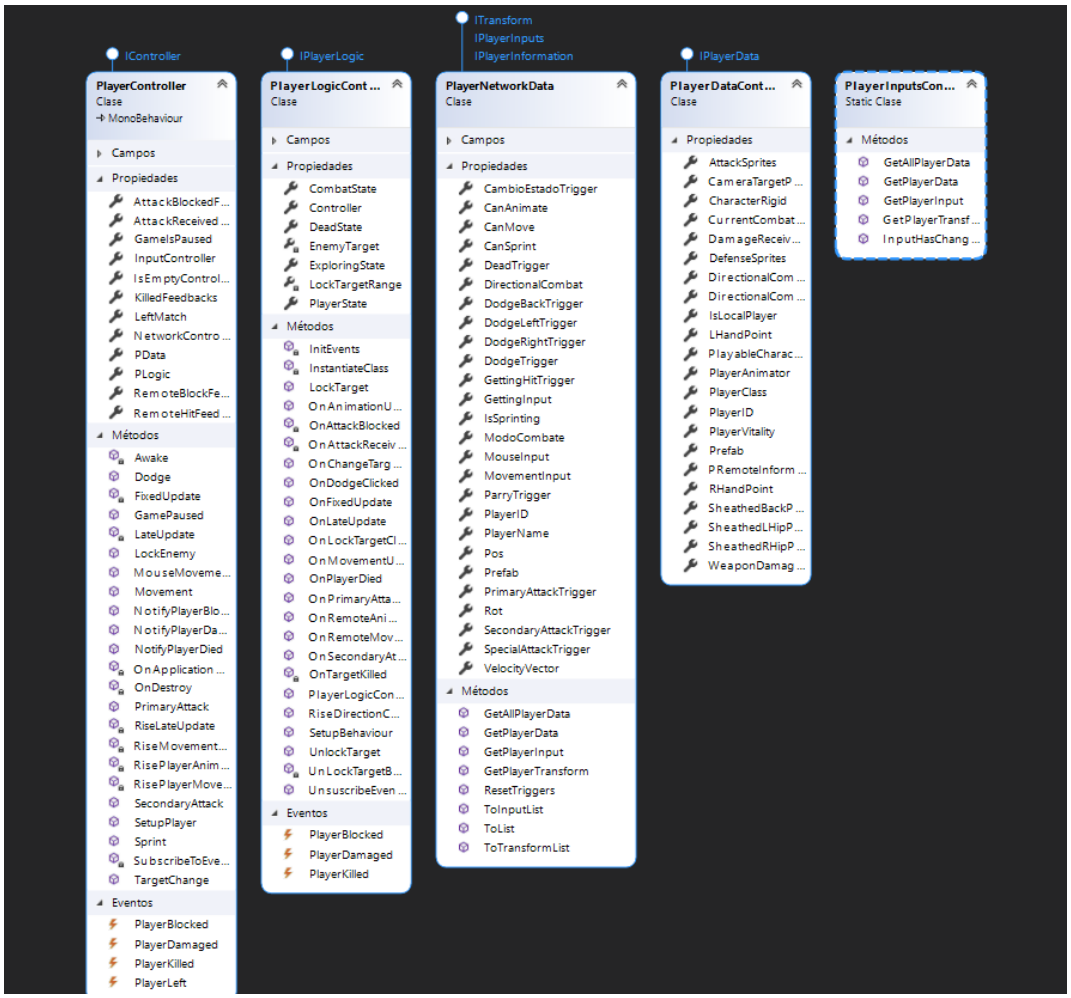


Figura 4.25: Clases encargadas del funcionamiento del PlayableCharacter

- States:
  El grupo de State solo contiene una clase abstracta y otras tres clases que heredan de ella (CombatState, DeadState y ExploringState).

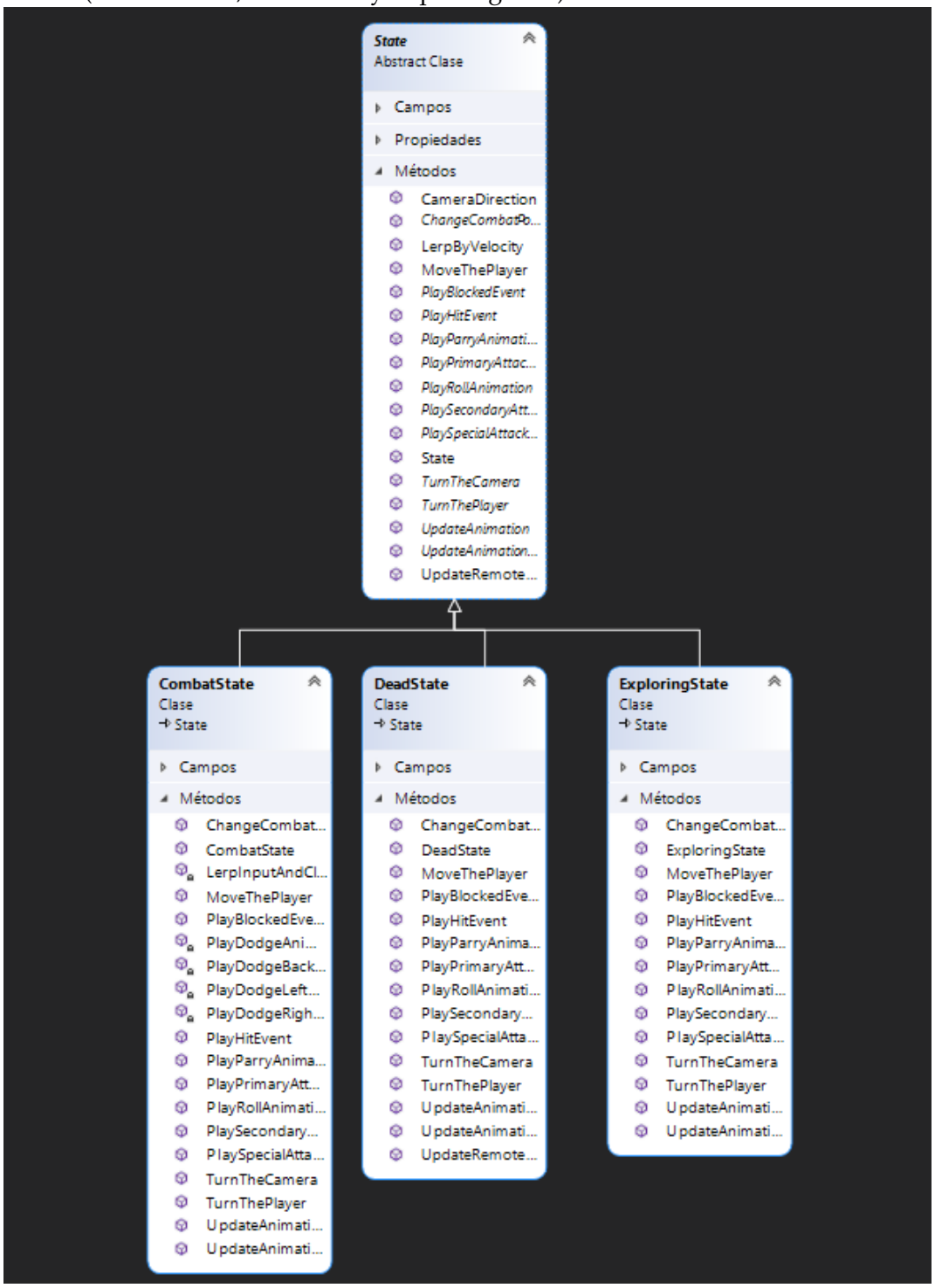


Figura 4.26: Clases que heredan de la clase abstracta State

- SingletonManagers:
  El grupo de clases más importante para el correcto funcionamiento del videojuego. Todas las clases en este grupo heredan de la clase genérica «Singleton». Eso asegurá que instancias de dichas clases siempre existan en el juego, independientemente de la escena actual.

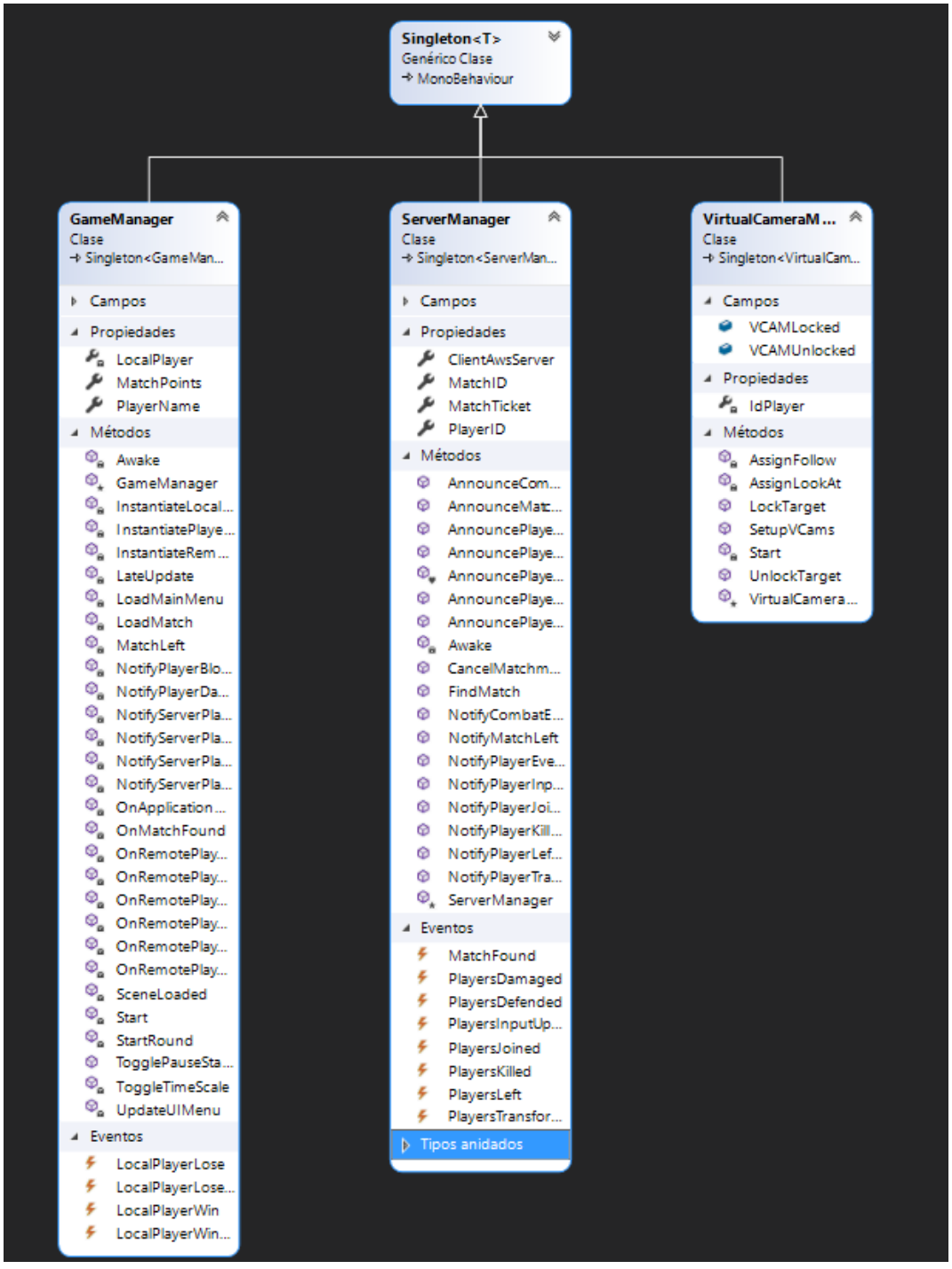


Figura 4.27: Managers que siguen el patrón de diseño «singleton»

- NonSingletonManagers:
  Clases controladoras que se encargan del correcto funcionamiento del juego. A diferencia de los otros controladores, la instancia de estas clases se crean y se destruyen sin problema alguno por el GameManager.

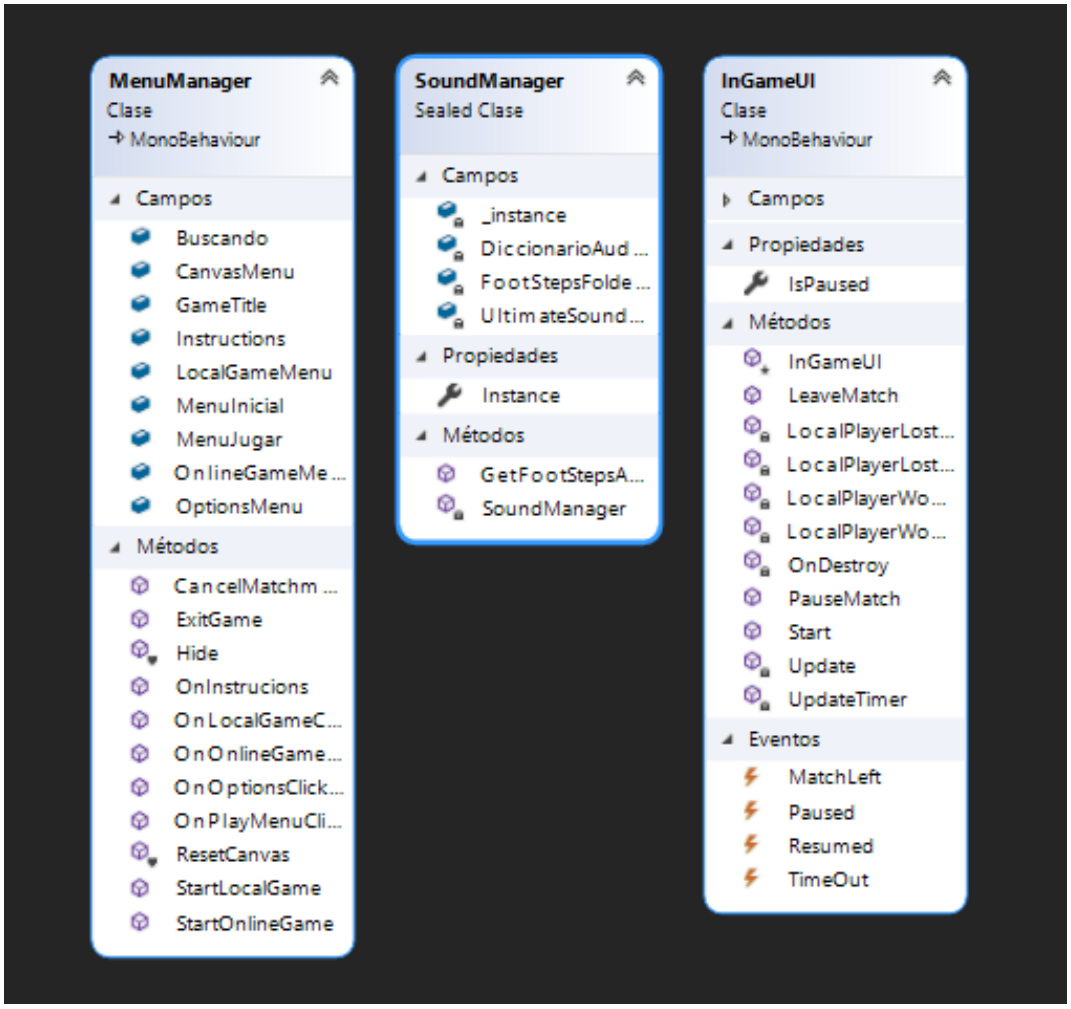


Figura 4.28: Managers de diferentes componentes del juego.

- NakamaAPI:
  Clases encargadas de enviar y recibir peticiones por la red. El ServerManager dispone de una instancia de cada clase. Con dichas instancias envía peticiones al servidor y propaga la información que recibe al GameManager.

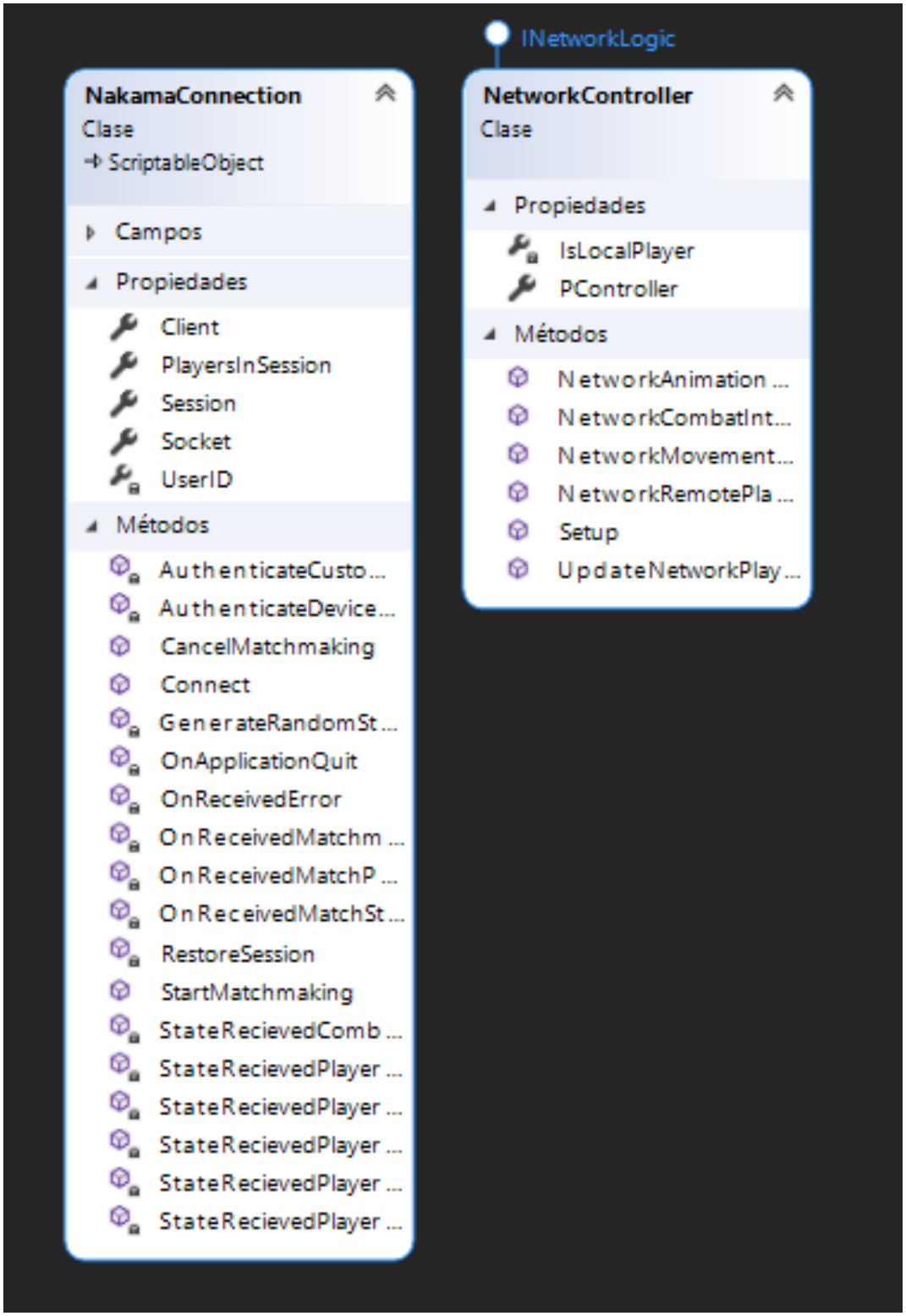


Figura 4.29: Clases encargadas de conectar con la instancia de NakamaServer en AWS

- SceneLoaders:
  Clases encargadas de cambiar de escenas. Cargan el menú principal o el Coliseo.

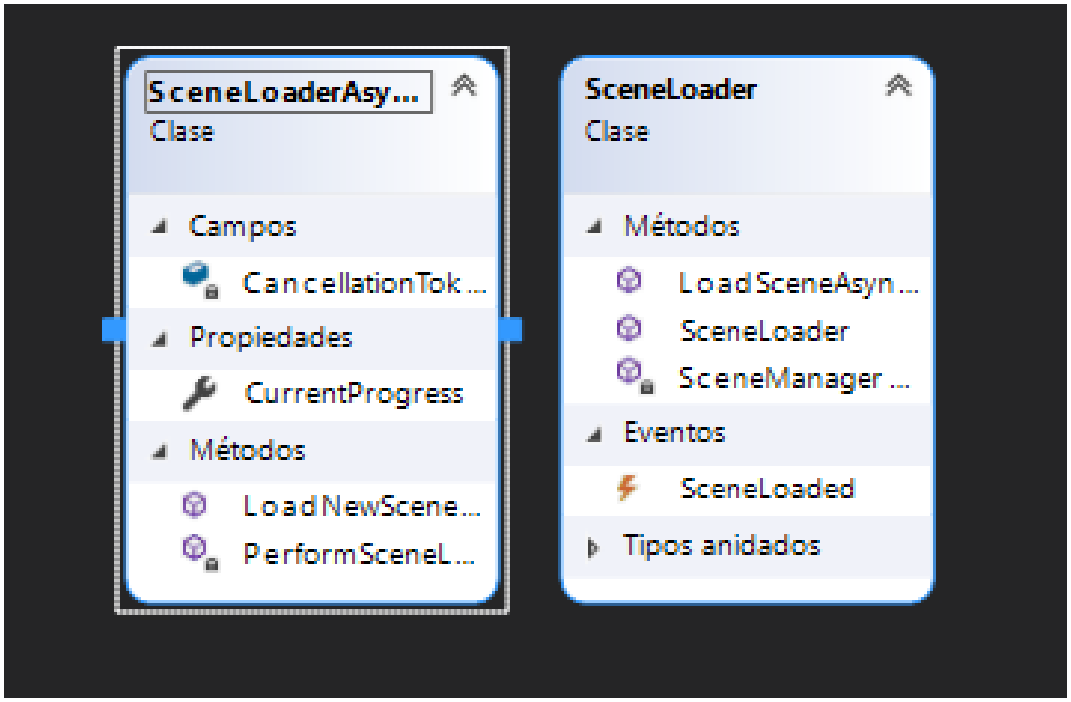


Figura 4.30: Clases encargadas de cambiar de escenas

- CombatLogic:
  Clases encargadas de toda la lógica del combate. Se comunican con diferentes componentes como la barra de salud, el GameManager, ServerManager y etc. Causan y reciben daño actualizando así las animaciones, la salud, los efectos de partículas, sonidos entre otras cosas.

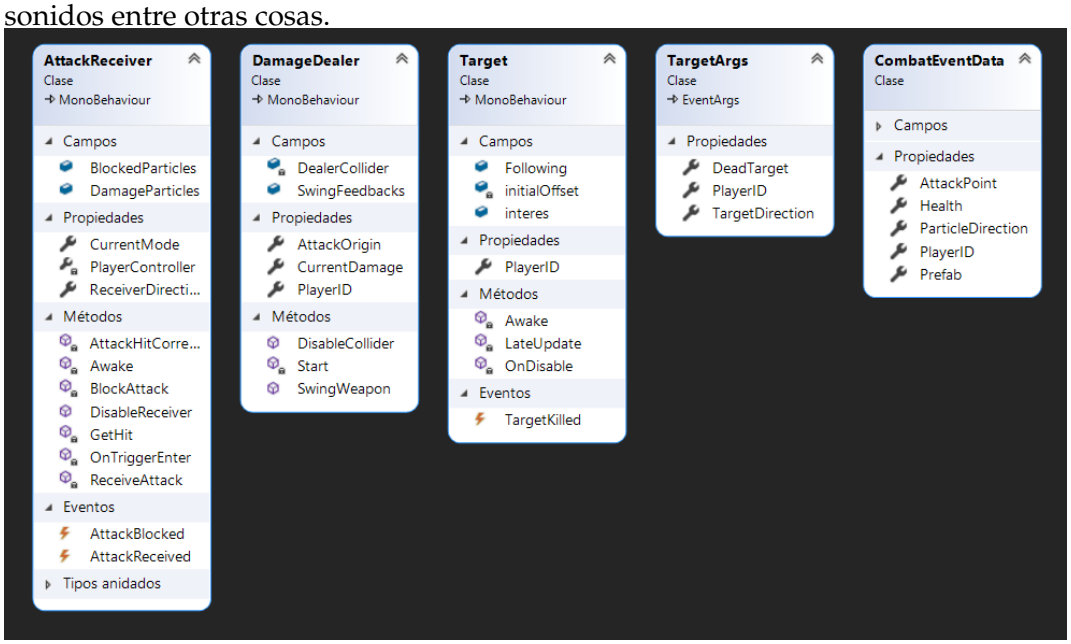


Figura 4.31: Clases encargadas de producir y detectar eventos de combate

- Utilities:
  Son clases sencillas de utilidad. Debido a la naturaleza de Unity, es muy fácil crear un componente pequeño y añadirle funcionalidad con unas pocas líneas de código. En esta categoría entran ese tipo de clases. Por ejemplo, «FootSounds» se encarga de detectar colisiones del terreno con el pie del jugador y así ejecutar un clip de audio de pisada.

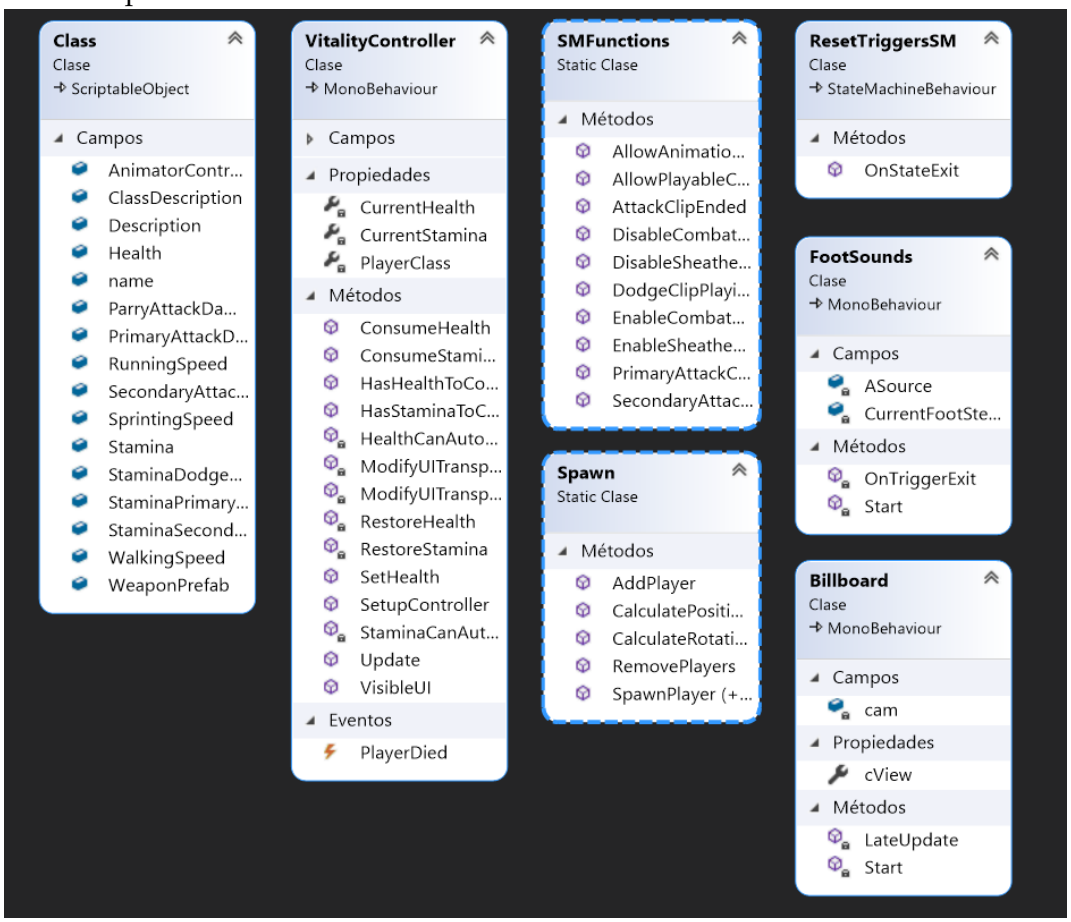


Figura 4.32: Clases de utilidad

CAPÍTULO 5
Trabajos futuros

A continuación se hablará de los posibles futuros trabajos respecto a este videojuego.

5.1 Modo historia

Actualmente Melilac sólo dispone de modo multijugador en línea, eso significa que el juego carece de mucho contenido original que enganche al jugador y le cuente una buena historia. En caso de querer continuar con el desarrollo de este videojuego, el siguiente paso sería implementar las unidades de trabajo que aporten funcionalidad que gire alrededor del modo historia y la campaña principal. Otro punto fuerte sería añadir contenido secundario como misiones opcionales que aporten más experiencia al jugador, habilidades escondidas que solo se pueden desbloquear si el jugador explora ciertas zonas ocultas. Personajes secundarios que aporten variedad al entorno.

5.1.1. Diálogos activos, historias adaptables

Algo que enriquecería mucho el contenido del juego son los diálogos activos[1] en momentos importantes de la trama y que en función de esas piezas de conversación la historia del juego evolucione de una forma u otra.

5.2 Combate

Una vez terminada la historia, sería interesante mejorar el sistema de combate del juego. Algo que estaría muy bien añadir sería un sistema de «parrys»[2] para añadir más comlejidad al sistema de combates. Otra idea interesante sería la implementación de un gancho o «grappling hook»[3]

[1] Diálogos donde el jugador puede elegir diferentes respuestas

[2] Un parry es movimiento que consiste en bloquear o rechazar el ataque de un enemigo, dejándolo en un estado vulnerable que permite contraatacarlo con muchísima fuerza.

[3] Mecánica cada vez más popular en los juegos de acción y aventuras. Consiste en una cuerda en cuya extremidad hay un tipo de ancla que sirve para engancharse a paredes o sitios elevados

Figura 5.1: Grappling hook clásico de la saga de videojuegos Just Cause

Otro idea interesante y prácticamente obligatoria, teniendo en cuenta la ambientación del juego, sería añadir un sistema de magias. Bolas de fuego, curaciones divinas, flechas arcanas, control mental y etc.

Figura 5.2: Vivi de Final Fantasy IX

CAPÍTULO 6
Conclusiones

Partiendo de lo que se ha explicado a lo largo de este documento, el trabajo de desarrollo, organización y resultado final del proyecto, puedo afirmar que el desarrollo del videojuego de combate en tercera persona con multijugador en línea ha sido un éxito.
¿Qué significa eso? Significa que los objetivos marcados para la fecha actual se han cumplido y se ha podido crear el MVP satisfactoriamente. Un equipo de una sola persona con sesenta y tres días de desarrollo a seis horas diarias de trabajo y poco presupuesto ha podido crear un producto software funcional de una calidad admisible. Sin duda se podría haber sacado un videojuego comercialmente aceptable si se hubiera dispuesto de más dinero y desarrolladores.

6.1 Conocimientos adquiridos

Desarrollar un videojuego supone tocar diversos temas diferentes como: apartado artístico, música y efectos de sonido, trama, conectividad y servidores, diseño de niveles, diseño de interfaz, programación y muchos otros componentes que se deben de tener en cuenta. Salvo que el equipo encargado del desarrollo cuente con muchas personas especializadas en diferentes temas, lo más probable es que, al final del día, uno tenga que tocar y probar muchas cosas diferentes hasta dar con lo que necesita. Eso implica una gran inversión de tiempo en el mero hecho de aprender en lugar de invertirlo en trabajar. En situaciones así es bueno utilizar herramientas usadas y probadas. Eso asegura la disponibilidad de una buena documentación y foros con muchas dudas resueltas. A la larga ese tipo de cosas hará que uno pueda avanzar y mejorar en el desarrollo de forma significativa.

Uno de los conocimientos adquiridos que resulta de más importancia es el funcionamiento de los videojuegos en línea. Sincronizar partidas, crear el matchmaking, envía y recibir información y actualizar diferentes entidades según los datos que se recibe por la red. Son cuestiones que yo no solía tener en cuenta al jugar en línea y que están ahí, presentes constantemente. Conocer más del tema hace que respete y admire más los títulos de éxito online en el mercado.

6.2 ¿Qué haría diferente si tuviera que empezar otra vez?

Si tuviera que empezar el proyecto una vez más desde cero, lo primero que haría sería excluir la clase de «caballero» en el producto final ya que las animaciones dejan mucho que desear.

Otra cuestión que habría hecho diferente sería el utilizar Photon en lugar de Nakama. Esto se debe a que Photon está específicamente diseñado para funcionar con Unity y a la larga eso me habría hecho más eficiente al largo del desarrollo. Todo sea dicho, Nakama es una herramienta muy buena y debido a su detallada documentación he podido desplegar un servidor en AWS con muchísima facilidad.

Bibliografía

[1] Simone Belli y Cristian López Raventós. "Breve historia de los videojuegos". En: *Athenea Digital. Revista de pensamiento e investigación social* 14 (2008), págs. 159-179.

[2] Juan Carlos Cantos Agudo. "Red P2P centralizada para el streaming de vídeo almacenado". Tesis doct. 2020.

[3] Borislav S. Đorđević, Slobodan P. Jovanović y Valentina V. Timčenko. "Cloud Computing in Amazon and Microsoft Azure platforms: Performance and service comparison". En: *2014 22nd Telecommunications Forum Telfor (TELFOR)*. 2014, págs. 931-934. DOI: 10.1109/TELFOR.2014.7034558.

[4] Jeanne B. Funk. "Reevaluating the Impact of Video Games". En: *Clinical Pediatrics* 32.2 (1993). PMID: 8432085, págs. 86-90. DOI: 10.1177/000992289303200205. eprint: https://doi.org/10.1177/000992289303200205. URL: https://doi.org/10.1177/000992289303200205.

[5] Eva Gil Romero y col. "Un estudio acerca del desarrollo de videojuegos mediante el motor gráfico Unity 3D". En: (2014).

[6] Javier Gómez y col. "Herramienta de diseño de juegos tipo mazmorra para Gamemaker Studio 2". B.S. thesis. 2019.

[7] Josh Jarrett. "Gaming the gift: The affective economy of League of Legends 'fair'free-to-play model". En: *Journal of Consumer Culture* 21.1 (2021), págs. 102-119.

[8] T Madhuri y P Sowjanya. "Microsoft Azure v/s Amazon AWS cloud services: A comparative study". En: *International Journal of Innovative Research in Science, Engineering and Technology* 5.3 (2016), págs. 3904-3907.

[9] Ville Majander. "Revenue models for video games". English. Master's thesis. Aalto University. School of Business, 2019, págs. 61+6. URL: http://urn.fi/URN:NBN:fi:aalto-201906234183.

[10] Maria Nicola y col. "The socio-economic implications of the coronavirus pandemic (COVID-19): A review". En: *International journal of surgery* 78 (2020), págs. 185-193.

[11] Díaz Olivera, Matamoros Hernández y col. "El análisis DAFO y los objetivos estratégicos". En: *Contribuciones a la Economía, marzo* (2011).

[12] D. Polančec e I. Mekterović. "Developing MOBA games using the Unity game engine". En: *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2017, págs. 1510-1515. DOI: 10.23919/MIPRO.2017.7973661.

[13] Andrew Sanders. *An introduction to Unreal engine 4*. CRC Press, 2016.

[14] Deniz Şener, Türkan Yalçın y Osman Gulseven. "The Impact of Covid-19 on the Video Game Industry". En: *Available at SSRN 3766147* (2021).

[15] Alan R Stagner. *Unity multiplayer games*. Packt Publishing Ltd, 2013.

[16] Jaakko Stenros, Janne Paavilainen y Frans Mäyrä. "Social interaction in games". En: *International Journal of arts and technology* 4.3 (2011), págs. 342-358.

[17] Jonathan Whetzel. *SIGNAL Post-Mortem: Lessons Learned Building an Online Experimental Wargaming Platform.* Inf. téc. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.

APÉNDICE A
GDD

El «Game Design Document» o «Documento de Diseño» (GDD para abreviar) es un documento cuyo contenido describe todo acerca del videojuego que trata. En dicho documento se explica todo lo que ha de tener el juego: desde sus posibles fuentes de inspiración, los componentes que lo forman, la idea que tiene detrás hasta lo que lo convierte en único.

Un GDD está hecho por el equipo de desarrollo y supone una colaboración entre las diferentes entidades encargadas de realizar el proyecto: programadores, artistas, diseñadores, músicos y etc. Dicho documento es utilizado como guía durante el proceso de desarrollo del juego. Tanto es así que las diferentes entidades que forman el equipo de desarrollo debe respetar el GDD durante el proceso, salvo en situaciones excepcionales.

Melilac

"Sobrevive un día más en la Arena de los Ya-muertos"

Gustavo Lee

Índice general

1 Introducción

En este documento se expondrá detalladamente cómo funciona Melilac, qué herramientas se han utilizado en su desarrollo, qué objetivos se pretenden alcanzar, las ideas básicas de este videojuego, sus mecánicas, plataformas de destino, elección artística y muchos otros detalles importantes para llevar a cabo este proyecto.

1.1. Concepto general

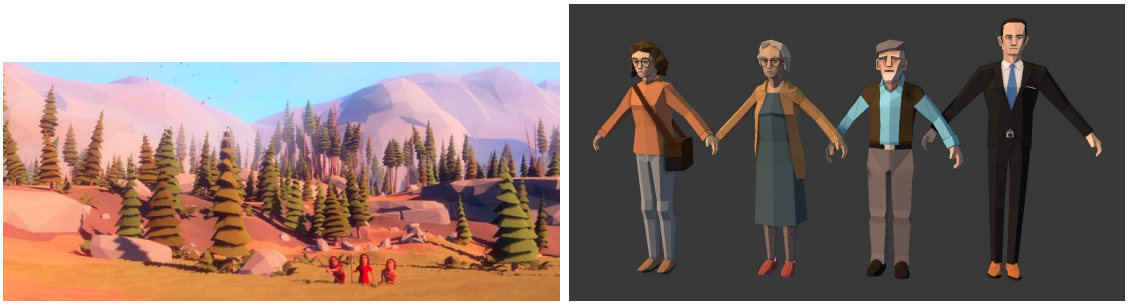
Melilac es un videojuego de combate en tercera persona de género fantástico. La idea más importante es ofrecer al jugador una buena historia mediante una campaña principal, y al mismo tiempo brindar la posibilidad de jugar en línea contra otras personas. Melilac dispone de un combate entretenido, una bonita banda sonora envuelta en una ambientación fantástica y medieval que atrapará al jugador desde el primer momento.

La idea más importante de Melilac es la de crear tensión en sus jugadores. La tensión que se busca no es la que puede provocar un título de terror. En cambio, Melilac busca crear momentos tensos mediante una competición igualada en la que cada decisión importa, y el medio por el cual se logra esto es mediante su sistema de combate. Este sistema de combate acaba creando un sentimiento de competitividad, especialmente en su modo multijugador online donde cada jugador debe derrotar al otro.

2 Especificaciones

2.1. Apartado artístico

El apartado artístico de Melilac es uno que se ve cada vez con más frecuencia en el mercado, especialmente entre los desarrolladores independientes. Lo que más llama la atención de dichos gráficos es lo muy cuadriculado y minimalista que pueden llegar a ser los modelos. Este arte tiene un nombre y es conocido en el mundo de los videojuegos como «Low Poly» debido al bajo número de polígonos que se usan para crear un objeto, personaje o entorno. El «Low Poly» además de tener un estilo único, proporciona una mejora en el rendimiento del sistema ya que no es necesario renderizar tantos vértices en cada fotograma.



(a) Bosque al estilo Low Poly (b) Ciudadanos al estilo Low Poly

Figura 2.1: Ejemplos arte Low Poly

El equipo encargado del apartado artístico de Melilac ha decidido hacer uso de los lotes Low Poly de Synty Studios porque este estilo minimalista supone un ahorro de espacio en los discos duros (al tener menos polígonos, ocupan menos), mejora la velocidad de renderizado y son baratos.

(a) Personajes Synty Studios (b) Mazmorra Synty Studios

Figura 2.2: Ejemplos arte de Synty Studios

2.2. Objetivos del desarrollo

Teniendo en cuenta el «MDA: A Formal Approach to Game Design and Game Research» de Robin Hunicke, Marc LeBlanc, Robert Zubek; existen, de una forma u otra, ocho tipos de diversión en un videojuego (sensación, compañerismo, desafío, fantasía, narrativa, descubrimiento, expresión, dedicación).
El objetivo de Melilac es abordar esos diferentes temas en función del modo de juego elegido por el usuario. Por ejemplo:

- Campaña principal: La campaña principal aborda la **fantasía** sumergiendo al jugador en un mundo nuevo y diferente, en una sociedad repleta de personas, humanas y no humanas. Además de la amplia variedad de razas, el jugador se encuentra en un universo donde existe la magia, injusticias, críticas sociales y muchos giros de guión poco esperados. La campaña principal también hace uso de la **narrativa** pues cuenta la historia de un esclavo obligado a luchar en un coliseo con tal de sobrevivir. Melilac también apela al sentimiento de **descubrimiento** ya que en cada acto de la historia principal presenta un nuevo desafío para el protagonista. De esa forma el jugador descubre, conforme vive los acontecimientos del juego, nuevas técnicas de combate, nuevos escenarios, nuevos objetos y habilidades especiales.

- Modo multijugador en línea: El modo multijugador tiene como objetivo crear combates memorables y divertidos, haciendo hincapié en el **desafío** que supone enfrentarse a otra persona.

The List

Because you asked, here is a brief list of the "Eight Kinds of Fun."

Sensation	**Fellowship**
Game as sense-pleasure	Game as social framework
Fantasy	**Discovery**
Game as make-believe	Game as uncharted territory
Narrative	**Expression**
Game as unfolding story	Game as soap box
Challenge	**Submission**
Game as obstacle course	Game as mindless pastime

Figura 2.3: Los ocho tipos de diversión por Marc LeBlanc

3 Jugabilidad y opciones

En esta sección hablaremos un poco sobre los distintos componentes que forman Melilac, desde la ambientación y la historia hasta los personajes y el objetivo principal del videojuego.

3.1. Historia

3.1.1. Acto I - Esperanza

Debajo del gran coliseo conocido como «La Arena de los Ya-muertos», o simplemente «La Arena» para abreviar, se halla «La Prisión Interminable». La Prisión Interminable se trata de una cueva subterránea integrado por un sinfín de pasillos de formación natural, húmedas rampas descendientes, paredes estrechas y agujeros resbaladizos cuyo extremo se desconoce. En La Prisión Interminable no existe una sola fuente de luz, en la Prisión Interminable no existen plantas ni animales, en la Prisión Interminable no existen el tiempo ni la cordura. Y aún así, en la Prisión Interminable, habitan centenas de personas. Personas cuyos destinos fueron sellados en el momento en que fueron encerrados en allí.

El juego empieza con el protagonista siendo encerrado en la Prisión Interminable. El protagonista es empujado cueva abajo sin saber siquiera el motivo de su encierro. Cada quincena del mes el Emperador Nómada celebra la «Conmemoración de los Ya-muertos» rescatando cincuenta de los prisioneros atrapados en la Prisión Interminable. A cambio de un deseo, el Emperador Nómada les pide que entretengan a él y al público de la Arena combatiendo por su vida.

El Emperador Nómada nunca ha concedido un solo deseo hasta ese día. Impresionado por la hazaña del protagonista, el Emperador le pregunta por su capricho. El protagonista le pide la libertad, el Emperador se lo concede con la condición de que sobreviva a cinco Conmemoraciones más.

3.1.2. Acto II - Sudor de sangre

El protagonista vuelve a la Prisión Interminable y hace lo que sea con tal de sobrevivir allí y a los futuros enfrentamientos.

Cada Conmemoración es diferente. Algunas tratan de luchas contra animales salvajes, otras Conmemoraciones tratan de sobrevivir a torturas en público, otras de luchar contra soldados armados y entrenados. Nadie nunca sobrevive.

Después de sobrevivir a cinco Conmemoraciones, el Emperador Nómada traiciona al protagonista y le da a elegir entre morir allí mismo o volver a la Prisión.

3.1.3. Acto III - Letargo

Derrotado, el protagonista vuelve a la Prisión Interminable. Sin saber qué hacer, toma la decisión de anda sin rumbo por la inmensidad de la cueva y sus estrechos pasillos, con la intención de nunca más volver.

Tras días vagando, sobreviviendo a base de gotas de agua e insectos, el protagonista encuentra una amplia camera iluminada por rocas que brillan con un azul intenso e intermitente. Decide pasar el resto de sus días allí. Al cabo de semanas de soledad, tristeza, impotencia y rabia acumulada, el protagonista decide llevar a los demás prisioneros a la recién bautizada «Camera Lunar» con tal de entrenarlos y preparar una revolución.

3.1.4. Acto IV - Insurrección

Con el paso del tiempo el protagonista logra convencer a todos los prisioneros de que se unan a su motín. No es tarea complicada, por algo se les conoce como «Ya-muertos». Una vez alguien es encerrado en esa la cueva, deja de ser considerado una persona a convertirse en un cadáver. La única diferencia con los cadáveres de verdad está en que los Ya-muertos sí sufren. Un año después, después de mucho sudor y sin haber vuelto a pisar el exterior, el protagonista da por terminado todos los preparativos para la revolución.

En el día de la Conmemoración, todos los prisioneros embisten la salida de la cueva. Rápidamente dejan fuera de combate e incluso matan a los arqueros del Emperador Nómada con piedras luminosas, mientras los prisioneros más fuertes ejecutan a los soldados de campo con sus propias manos. La revolución se lleva a cabo tal y como se había planeado. Son demasiados pocos soldados y arqueros en la Arena de los Ya-muertos. El Emperador Nómada nunca había temido un alzamiento pues todas las personas le eran leal o le temían lo suficiente para llegar a serlo. La Conmemoración era un día de júbilo, una fiesta para su imperio, era impensable que algo así pudiera ocurrir. Por eso no había soldados suficientes ese día, por eso nunca había soldados suficientes. No eran necesarios cuando se trataba de los Ya-muertos.

El Emperador libera a los leones, los Ya-muertos los abaten utilizando puñales de piedra y pesadas lanzas de estalactitas. El Emperador envía a los cinco únicos magos de la corte que mueren rápidamente a manos de los rebeldes, pero no sin antes haber calcinado a cuatro decenas de Ya-muertos. Desesperado y sin otra opción, el Emperador desencadena al gigante orco. El momento que llevaba esperando el protagonista durante todo un año. Haciendo uso de una magia que poco conoce y que apenas controla, nuestro protagonista fusiona su consciencia con la del verde gigante. Viendo todo de otra perspectiva y sin pensarlo dos veces, el orco avanza con una velocidad impresionante hacia el Emperador y de un solo golpe hace estallar todo su cuerpo.

Ante la muerte del Emperador Nómada, los pocos soldados aún vivos escapan lo mejor que pueden, los espectadores de la Arena intentan huir junto a ellos. El verde orco arrasa con las puertas de metal y paredes de piedra. Destruye todo cuanto puede con tal de liberar a los Ya-muertos que todavía siguen vivos. Escapan de la ciudad lo más rápido posible, huyen hacia las montañas donde se dice que existen infinitas fuentes de agua. Pocos sobreviven, pero sobreviven lo mejor que pueden.

3.2. Personajes

- Protagonista: Se trata de un personaje modular de género masculino o femenino. La idea está en que el jugador puede personalizar su apariencia antes de empezar la campaña principal.

- Emperador Nómada: Es el antagonista del juego. Se trata nada más y nada menos del hombre encargado de todo el sufrimiento de los prisioneros de la Prisión Interminable. Su apariencia engaña al principio pues parece un rey justo y bueno, pero las apariencias discrepan con la realidad de su persona. Nómada es un sádico hombre lleno de riquezas, con una corona de oro y una sonrisa llena de malicia. Gobierna sobre todos los demás habitantes del continente en el que se halla.

Figura 3.1: El Emperador Nómada

- Los Ya-muertos: Los Ya-muertos son todos los prisioneros del Emperador Nómada. Son retenidos dentro de la Prisión Interminable y sirven de distracción para el Emperador en los día de la «Conmemoración de los Ya-muertos» donde luchan hasta perecer con tal de entretenerle.

Figura 3.2: Los Ya-muertos

- Orco sin nombre: El Orco sin nombre se trata del primer prisionero Ya-muerto que tuvo el Emperador Nómada. Antes de contruir la Arena de los Ya-muertos, el Emperador Nómada necesitaba hacer algo con sus prisioneros con tal de pasar el rato. Usando magia antigua, le arrebató la conciencia y la encerró bajo llave en un baúl cuyo paradero es desconocido. A partir de ese momento, el Orco sin nombre pasó a convertirse en un cascarón vacío incapaz de hacer nada por sí solo. Eso es así hasta que el Emperador decide controlarlo con su magia antigua otra vez.

Figura 3.3: Orco a punto de golpear al Emperador Nómada

3.3. Multijugador online

La historia detrás del multijugador online es sencilla: dos Ya-muertos deben enfrentarse a muerte con tal de entretener al Emperador Nómada.

El modo en línea se puede acceder desde el menú principal, en la opción de «multijugador online». Allí se puede escribir el nombre que tendrá el jugador y su clase (ladrón, bárbaro, o caballero). Una vez seleccionada las opciones, empieza el matchmaking. Al cabo de unos segundos la partida es encontrada y aparece una pantalla de carga que, una vez finalizada, lleva al usuario a la partida para que luche contra otro jugador.

Figura 3.4: Luchadora bárbara en la Arena contra un ladrón

Cada partida comprende de tres rondas. El primer jugador que llegue a las tres rondas, gana. Una vez el combate haya sido ganado o perdido, ambos jugadores vuelven al menú principal donde pueden elegir jugar otra partida en línea si así desea.

3.4. Ambientación

Melilac está ambientado en un mundo fantástico cuyo dominio está en manos de un tirano llamado Emperador Nómada. La idea de esta ambientación llena de razas está en mostrar lo opresivo que es la situación bajo el control del Emperador. El juego pretende crear un ambiente represivo y angustiante para los ciudadanos que viven en esta situación. El entorno es prácticamente desértico ya que el Emperador Nómada es quien controla la única fuente de agua en todo el mundo.

3.5. Objetivo principal

No existe un solo objetivo principal en Melilac, cada modo de juego tiene su finalidad. La campaña principal pretende contar una historia intrigante y tensa donde el protagonista se ve obligado a sobrevivir y escapar de una prisión en la que diariamente se celebran combates a muerte. Con el transcurso de las contiendas, el protagonista desbloquea nuevas habilidades y aumenta su destreza con las armas que use más frecuentemente, llegando así a ser más letal con tal de sobrevivir a los desafíos más duros que le aguardan.
El objetivo principal del multijugador en línea es brindarle al usuario la oportunidad de tener uno o varios combates cortos contra otra persona.
La campaña principal se juega por la intriga, la historia y las nuevas habilidades. El multijugador en línea se juega por la competición, por romper la monotonía y sobre todo por pasar un rato divertido.

3.6. Mecánica central

El núcleo de Melilac es el sistema de combate direccional. Este sistema toma los movimientos de ratón y en función de ellos el jugador puede decidir si realizar un ataque desde arriba, izquierda o derecha. Una decisión importante en un combate es saber cuando embestir al enemigo y cuando no. Mientras un personaje mantenga la guardia y no decida realizar una acometida, bloqueará todos los ataque que lleguen por el lado en el que tenga el ratón.
Lo interesante de este sistema es el juego entre atacar y defender y la tensión que puede llegar a generar es divertida y con eso intenta jugar Melilac. En caso de que el jugador asalte por la derecha, su enemigo se puede defender poniendo su ratón a la izquierda (la derecha del atacante) y viceversa.

Estos son los iconos encargados de mostrarle al jugador su posición de combate actual:

1. **Indefenso:**
   Cuando el combate direccional del jugador muestra este icono, significa que todos los ataques, independientemente de la dirección origen, causará daño al jugador.

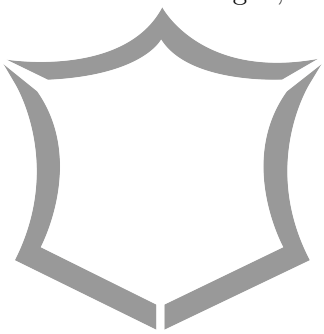


Figura 3.5: No se bloquea ataques por ningún lado

2. **Ataque/defensa por la izquierda:**
   Cuando el combate direccional del jugador muestra el icono de abajo, significa que el jugador bloqueará todos los ataque que vengan de la izquierda, también significa que el ataque que realice se hará desde la izquierda.

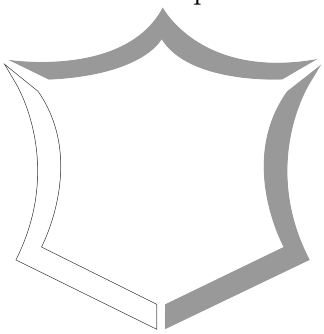


Figura 3.6: Ataque y bloqueo desde la izquierda

3. **Ataque/defensa por arriba:**
   Cuando el combate direccional del jugador muestra el icono de abajo, significa que el jugador bloqueará todos los ataque que vengan de arriba, también significa que el ataque que realice se hará desde ese mismo lado

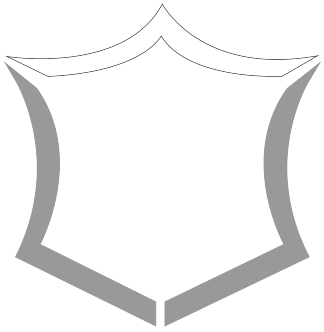


Figura 3.7: Ataque y bloqueo desde arriba

4. **Ataque/defensa por la derecha:**
   Cuando el combate direccional del jugador muestra el icono de abajo, significa que el jugador bloqueará todos los ataque que vengan de la derecha, también indica que el ataque que realice se hará desde la derecha

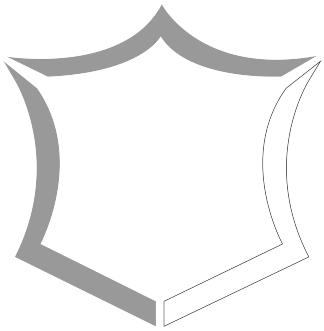


Figura 3.8: Ataque y bloqueo desde la derecha

3.7. Controles

Melilac está pensado y diseñado para ser jugado en un PC así que los controles son los siguientes:

- **Teclado -** los controles del teclado en Melilac están compuesto por siete botones siendo estos:

  1. WASD: estos cuatro botones son los que utiliza el jugador para controlar el movimiento del personaje.
     «W» hace que el personaje del jugador se mueva hacia delante.
     «A» hace que el personaje del jugador se mueva a la izquierda.
     «S» hace que el personaje del jugador se mueva hacia atrás.
     «D» hace que el personaje del jugador se mueva a la derecha.

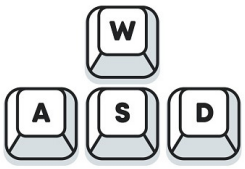


Figura 3.9: Teclas de movimiento

  2. Barra espaciadora: este botón es el que utiliza el jugador para las realizar las volteretas. Cuando el jugador no tiene la vista fijada en un enemigo, la voltereta siempre se realiza hacia donde esté mirando su personaje. En cambio, cuando la vista está fijada, la voltereta siempre se realiza en la dirección del movimiento del personaje.

Figura 3.10: Tecla de encargada de las volteretas

3. Shift: botón encargado del esprint del jugador. Cada clase tiene una velocidad de esprint diferente.

Figura 3.11: Tecla de encargada del esprint del jugador

4. Escape (ESC): botón encargado de abrir y cerrar el menú de pausa dentro de una partida. En dicho menú puedes elegir reanudar la partida o volver al menú principal.

Figura 3.12: Tecla Pausa

- **Ratón**

  el ratón es otro componente fundamental en la jugabildad de Melilac. Se encarga de controlar la cámera del jugador y de cambiar las posiciones para el combate direccional.

  1. Click izquierdo: botón del ratón encargado de realizar un ataque débil. El ataque se realizará por arriba, izquierda o derecha en función de la posición de ataque direccional.
  2. Click central: botón encargado de fijar la cámara en un objetivo, o en caso de que ya esté fijada, desfija la cámara. Cuando la cámara está fijada, el persojane del jugador desenfunda sus armas y se prepara para la batalla. En caso de desfijar la cámara, el personaje del jugador enfunda sus armas.
  3. Click derecho: botón del ratón encargado de realizar un ataque fuerte. El ataque se realizará por arriba, izquierda o derecha en función de la posición de ataque direccional.

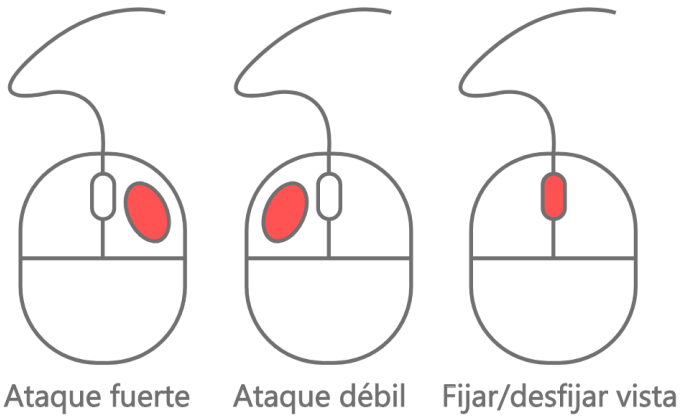


Figura 3.13: Botones encargados del combate

4. Movimiento a la izquierda: cambia la posición del combate direccional. El personaje del jugador bloqueará todos los ataques que vengan de la izquierda, y en caso de que pulse el click izquierdo o derecho, realizará un ataque por este mismo lado.

Figura 3.14: Movimiento combate direccional a la izquierda

5. Movimiento hacia arriba: cambia la posición del combate direccional. El personaje del jugador bloqueará todos los ataques que vengan de **arriba**, y en caso de que pulse el click izquierdo o derecho, realizará un ataque por este mismo lado.

Figura 3.15: Movimiento combate direccional hacia arriba

6. Movimiento a la derecha: cambia la posición del combate direccional. El personaje del jugador bloqueará todos los ataques que vengan de la **derecha**, y en caso de que pulse el click izquierdo o derecho, realizará un ataque por este mismo lado.

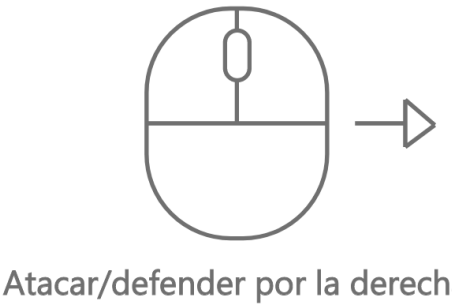


Figura 3.16: Movimiento combate direccional a la derecha

4 Front End

Los componentes de interfaz de usuario son muy sencillos. Melilac se compone de un menú principal donde se puede acceder al modo historia, al modo multijugador en línea, a las instrucciones, opciones y por último u botón para salir al escritorio.

La tipografía utilizada es la de Cinzel Regular para los títulos y Bentham para los textos. Ambas tipografías son de estilo egipcio que pegan bastante con la temática y lo que pretende transmitir Melilac.

4.1. Flujo del videojuego

Este es el diagrama de flujo de Melilac. La idea es hacer un flujo de aplicación fácil y sencillo en el que el jugador no se pierda navegando.

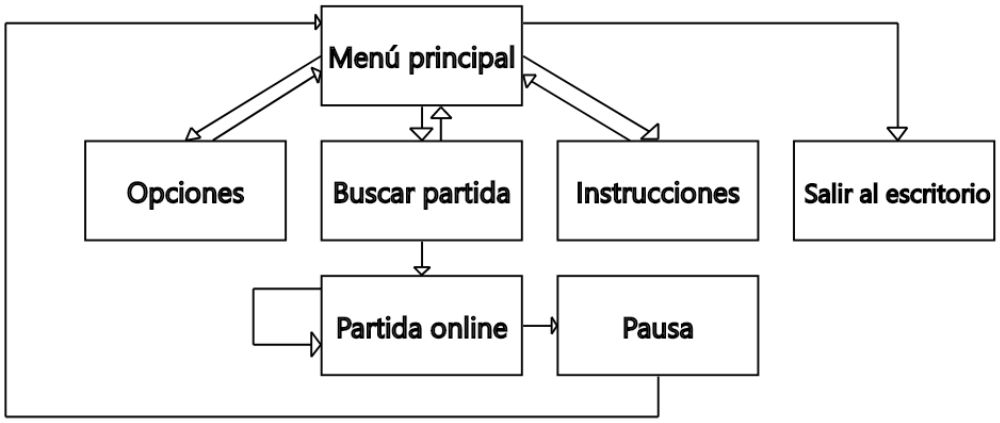


Figura 4.1: Menú principal de Melilac

4.2. Menús

A continuación se nombrarán los menús principales de Melilac

- Menú principal: Se trata del primer menú que ve el jugador. Desde este punto puede ir a cualquier otro lugar de Melilac

Figura 4.2: Diagrama de flujo del videojuego

- Menú jugar: En el menú jugar el usuario puede acceder al modo campaña o el multijugador en línea.

Figura 4.3: Menú jugar

- Menú partida en línea: Aquí el jugador puede elegir un nombre y una clase antes de iniciar el matchmaking

Figura 4.4: Menú partida en línea

- Pantalla búsqueda de partida: En esta pantalla se puede cancelar el matchmaking y volver a los menús anteriores. En caso de no hacerlo y encontrar partida, el jugador es llevado a otra escena donde puede luchar.

Figura 4.5: Buscando partida

- Instrucciones de combate: Aquí el jugador puede leer los comandos de combate. Son las intrucciones básicas del juego.

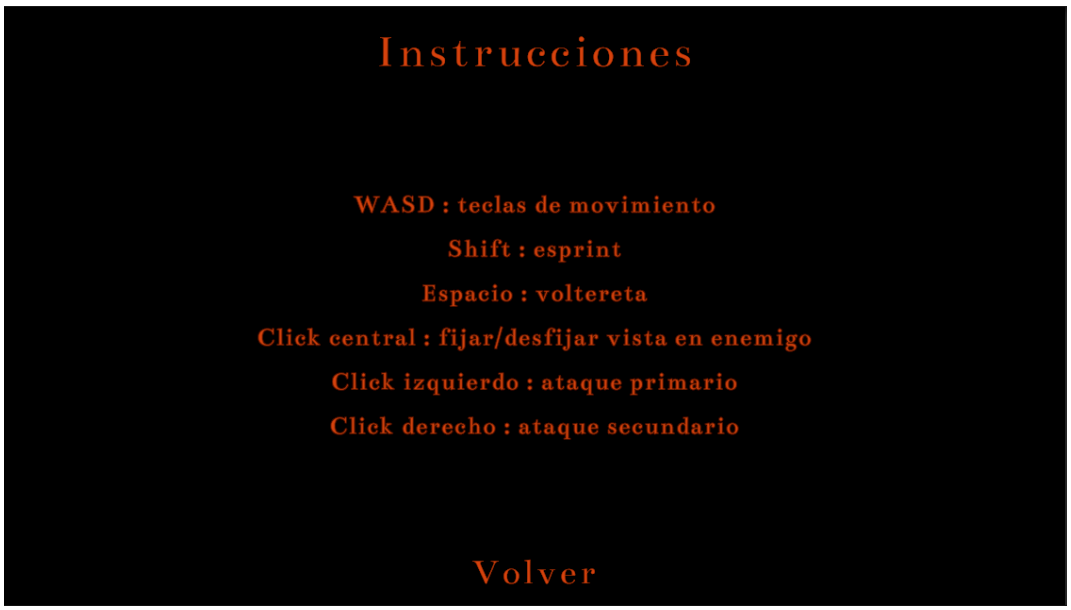


Figura 4.6: Instrucciones del combate

- Opciones del juego: En esta sección el jugador puede ajustar el videojuego a su gusto. Eso incluye ajustes de resolución, calidad de imagen, sonidos y pantalla completa o no.

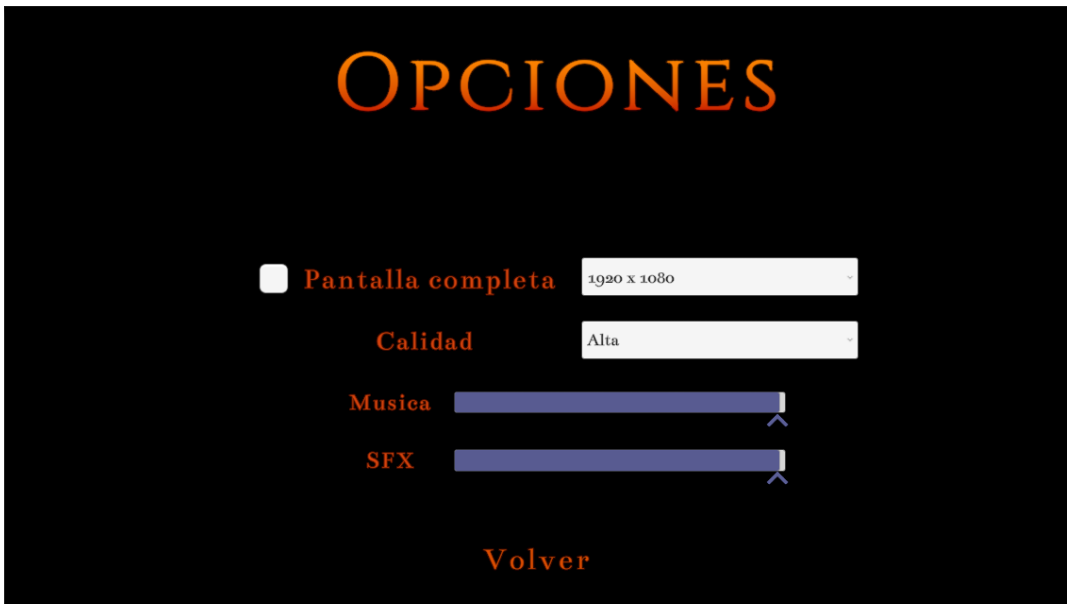


Figura 4.7: Opciones del juego

5 Tecnologías

En esta sección del GDD se expondrá la parte más técnina del desarrollo. Se mencionarán las tecnologías utilizadas, el motor gráfico cuestión, las herramientas que han sido utilizadas por desarrolladores, el hardware necesario y etc.

5.1. Plataformas finales

Melilac solo está disponible para ordenadores con Windows 7, Windows 8.1 o Windows 10, versiones de 64 bits.

5.2. Hardware

Para jugar a Melilac se necesita un ordenador con Windows, un teclado y un ratón. En caso de jugar al modo multijugador online, Melilac también necesitará conexión a internet.

Los requisitos mínimos del sistema son:

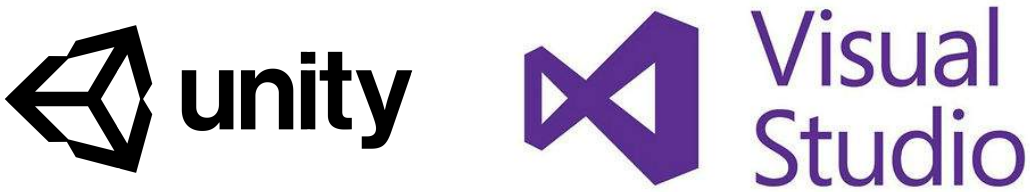
- CPU: Intel Core i3-3210 3.2 GHz / AMD A8-7600 APU 3.1 GHz
- RAM: 4 GB
- GPU: Nvidia GeForce 400 Series o AMD Radeon HD 7000 Series

Los requisitos recomendados del sistema son:

- CPU: Intel Core i5-4690 3.5 GHz / AMD A10-7800 APU 3.5 GHz
- RAM: 8 GB
- GPU: GeForce 700 Series o AMD Radeon Rx 200 Series

5.3. Herramientas y sistemas de desarrollo

Melilac se ha desarrollado utilizando Unity 2021.1.17f1 como motor gráfico. Para trabajar con esta herramienta, el equipo de desarrollo utiliza el lenguaje de programación C# en Visual Studio 2019 de Microsoft.



(a) Unity Engine (b) Visual Studio IDE

Figura 5.1: Herramientas principales en el desarrollo

Se ha utilizado Blender para el modelado de ciertos componentes tridimencionales sencillos (piedras, rocas, columnas entre otras cosas) y Ableton Live 10 como secuenciador de audio y MIDI para la composición de las canciones.

(a) Logo Blender (b) Logo Ableton Live 10

Figura 5.2: Herramientas secundarias en el desarrollo

6 Equipo y créditos

El equipo encargado de este proyecto está compuesta por una sola persona, yo, Gustavo Lee. Soy un alumno de ingeniería informática de la Universitat Politècnica de València. Tengo veintitrés años y he decidido aceptar el reto de hacer mi primer videojuego "serioçomo trabajo de fin de grado. Me he encargado de casi todos los aspectos de este juego, salvo únicamente por los diseños tipográficos. Desde la programación, diseño de niveles, composición de luces y sonidos. Especial mención a María Molas Ruíz por sus consejos acerca del diseño tipográfico de Melilac.

Figura 6.1: Combate en la Arena de los Ya-muertos