CONSEJERÍA DE EDUCACIÓN

Comunidad de Madrid

IES ENRIQUE TIERNO GALVÁN

Parla

CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Curso 2024/2025

Proyecto DAM

TÍTULO: *Videojuego Arcane Hunter*

Alumno**:** *Daniel Expósito Seoane*

Tutor**:** *Jesús García Romero*

Junio de 2025

Índice

1. Conceptualización.

1.1 Definición del videojuego.

Arcane Hunter es un videojuego de **acción y aventura** en 2D desarrollado con el motor **Godot Engine**. El proyecto busca recrear una experiencia clásica de exploración de mundos, combate, recolección de objetos e interacción con NPCs (Non-player character o Personaje no jugable). El jugador encarna a un caballero que atraviesa diversos entornos, enfrentándose a enemigos adaptativos y jefes desafiantes.

Se narra la historia de un caballero que, durante **una invasión de criaturas oscuras**, pierde a su **amada**, llevada más allá de los confines del mundo conocido.

Motivado por el amor y la pérdida, **el caballero** de Arcane Hunter se embarca en una misión épica: atravesar **mundos salvajes y hostiles**, donde el aire, la gravedad y los peligros cambian en cada entorno.

En su viaje, enfrentará **enemigos dinámicos y agresivos** que evolucionan para detenerlo. Recolectará **runas antiguas y monedas mágicas**, que le darán fuerza para avanzar y desbloquear secretos ocultos.

Cada mundo superado lo acerca más a su destino, pero también aumenta el desafío. Las criaturas del vacío, comandadas por un ser desconocido, harán todo lo posible para impedir su avance. Su único propósito: Rescatar a su amada y restaurar la luz perdida en su hogar.

1.2 Objetivos.

El objetivo inicial del proyecto es ofrecer un **juego de calidad** que combine mecánicas modernas con una estética **retro en pixel art**. El sistema de progresión incluye recolección de monedas y runas, combate contra enemigos, desbloqueo de niveles y un sistema de logros basado en estrellas por niveles.

Basado en un **prototipo previo** en el que se busca mejorar y ampliar sus mecánicas, optimizar su rendimiento e incorporar nuevas funcionalidades que enriquezcan la experiencia del jugador.

1.3 Orientación y público destino.

La aplicación está orientada a un **público joven y adulto** aficionado a los **juegos indie** de plataformas *on-premise*. Se ha priorizado la jugabilidad fluida, la rejugabilidad y el diseño cuidado de niveles.

1.4 Story Board.

En el **Anexo II** se adjunta el **Story Board** del videojuego, donde se representan las escenas clave y el flujo de la historia visualmente.

2. Análisis y diseño de la base de datos.

2.1 Análisis.

Debido a la naturaleza del proyecto, **no se ha utilizado una base de datos tradicional** para la persistencia de la información. En su lugar, se optará por almacenar datos relevantes —como configuraciones, progreso del jugador y estadísticas— en **ficheros JSON internos**, adecuados para un videojuego local y ligero.

Esta elección se fundamenta en varios motivos:

- **Sencillez y agilidad**: La estructura de datos que maneja el videojuego es relativamente simple y no justifica el uso de un sistema de gestión de bases de datos relacional o documental complejo.

- **Persistencia local**: El juego está diseñado para plataformas donde los datos del usuario permanecen en su dispositivo, sin necesidad de conexión a servidores externos.

- **Escalabilidad suficiente**: Aunque la información puede crecer (por ejemplo, estadísticas de niveles completados), el volumen no alcanzará un tamaño que requiera soluciones de bases de datos más robustas.

- **Legibilidad y accesibilidad**: Son formatos de texto plano, fácilmente comprensibles y modificables incluso manualmente en caso necesario.

- **Portabilidad**: Los datos se pueden mover entre dispositivos, realizar copias de seguridad o restauraciones de forma sencilla.

- **Flexibilidad de actualización**: La estructura de los archivos permite añadir nuevos campos sin romper compatibilidad con versiones anteriores del juego.

- **Rendimiento**: Al estar diseñados para un acceso rápido y directo, reducen la latencia y mejoran la experiencia del jugador en dispositivos de recursos limitados.

2.2 Diseño.

El diseño conceptual consiste en la definición de esquemas JSON como:

2.2.1 Configuración / Ajustes del usuario

Se registran las preferencias personales del jugador, tales como:

- Volumen general del audio.

- Volumen de los efectos de sonido.

- Activación o desactivación de efectos visuales como partículas ambientales.

Estos ajustes permiten adaptar la experiencia de juego según las necesidades o gustos individuales del jugador y son persistentes entre sesiones.

2.2.2 Estadísticas generales del jugador

Este fichero almacena datos agregados que reflejan el progreso global del usuario a lo largo de la partida, incluyendo:

- Número total de monedas recogidas.
- Número total de runas obtenidas.
- Tiempo total jugado acumulado.
- Número de enemigos derrotados.
- Número de jefes derrotados.
- Número total de muertes del personaje jugador.

Estas estadísticas sirven tanto para ofrecer retroalimentación al jugador como para alimentar pantallas de resumen o logros.

2.2.3 Estadísticas individuales por nivel

Se conserva información específica para cada nivel superado, incluyendo:

- Número de monedas obtenidas en el nivel.
- Número de runas recogidas en el nivel.
- Tiempo invertido en completar el nivel.

- Cantidad de estrellas alcanzadas (de 0 a 3).

Estos datos permiten mostrar un progreso detallado y motivar al jugador a mejorar su rendimiento en cada nivel.

2.3 Diseño artístico y recursos visuales

El desarrollo del apartado artístico de *Arcane Hunter* ha sido un proceso fundamental dentro de la fase de diseño. Cada fondo, botón, barra de salud y cada fotograma de las animaciones de personajes ha requerido una pieza artística individual. Si bien parte del material ha sido extraído de bibliotecas de recursos bajo licencia libre pero comercialmente limitada (**Anexo I**), se ha seguido un criterio coherente de estilo visual, manteniendo una estética retro en pixel art en todo el proyecto.

Durante la fase de preproducción se recopilaron referencias, se definieron estilos y se documentaron necesidades gráficas. Esta etapa facilitó el trabajo posterior en la selección, adaptación y programación de los recursos gráficos.

Algunos aspectos destacados del diseño artístico incluyen:

- **Material conceptual**: A partir del storyboard (**Anexo II**) y de la definición del universo del juego, se establecieron los temas visuales predominantes (fantasía oscura, ambientación pixelada, atmósferas dinámicas).

- **Interfaz de usuario coherente**: Los menús, indicadores y botones comparten una estética clara y funcional, compatible con el resto del juego.

- **Animaciones personalizadas**: Se han editado y sincronizado manualmente múltiples spritesheets para animar personajes, enemigos, jefes y efectos ambientales (como lluvia, partículas o rayos).

- **Accesibilidad visual**: Se ha cuidado la legibilidad del texto, el contraste entre elementos y la claridad del HUD durante la partida.

La planificación previa y la organización del trabajo artístico han permitido mantener una producción eficiente, minimizando la improvisación en fases avanzadas. El uso de herramientas de edición como Aseprite, GIMP o editores online específicos para sprites ha complementado esta tarea.

3. Aspectos funcionales y de diseño de la aplicación.

3.1 Introducción.

La aplicación desarrollada se compone de varios **módulos principales**, cada uno enfocado en un conjunto de funcionalidades específicas que garantizan una experiencia de usuario fluida y estructurada. Estos módulos son:

- **Menú principal**: Punto de partida que permite navegar hacia las diferentes áreas del juego: inicio de partida, configuración, estadísticas o salida.

- **Mapa de selección de niveles**: Interfaz gráfica que proporciona acceso a los niveles desbloqueados y muestra el progreso en cada uno de ellos.

- **Sistema de combate**: Gestión de ataques, detección de impactos, daño a enemigos y control de estados de vida o muerte tanto de los enemigos como del jugador.

- **Sistema de recogida**: Control de recolección de monedas y runas, afectando directamente a las estadísticas acumuladas.

- **Pantallas de Game Over y Victoria**: Visualización de estadísticas detalladas obtenidas tras finalizar un nivel, con opciones de reinicio o retorno al mapa.

- **Pantalla de Estadísticas Totales**: Acumulación y visualización de los logros y progresos generales del jugador a lo largo del juego.

La programación se ha realizado principalmente en **GDScript**, respetando principios de **modularidad**, **reutilización** y **responsabilidad única** en el diseño de scripts, siguiendo buenas prácticas en el desarrollo de videojuegos.

3.2 Menú principal.

El **Menú principal** constituye la primera pantalla que el jugador visualiza al iniciar el videojuego. Se ha diseñado siguiendo una estética coherente con el estilo **retro pixel art** de *Arcane Hunter*, garantizando una experiencia de navegación clara, dinámica y atractiva.

El menú ofrece varias opciones fundamentales:

- **Botón de información/ayuda**: Proporciona acceso a un documento que explica el uso básico del videojuego, los controles y el objetivo del mismo, sirviendo como guía para jugadores principiantes.

- **Botón "like"**: Permite realizar una acción de reconocimiento hacia el creador del videojuego, incentivando la valoración positiva del proyecto.

- **Botón de ajustes/configuración**: Accede a una ventana donde el usuario puede modificar parámetros como el volumen general, volumen de efectos, idioma del juego (español/inglés) y activar o desactivar efectos visuales como partículas.

- **Botón de estadísticas**: Dirige a la visualización de las estadísticas acumuladas en la partida, mostrando tanto estadísticas globales como específicas por nivel.

- **Botón iniciar**: Permite comenzar la partida trasladando al usuario al mapa de selección de niveles.

- **Botón salir**: Cierra la aplicación de manera segura y ordenada.

La disposición de estos elementos respeta los principios de **usabilidad** y **accesibilidad**, facilitando el flujo de navegación para el usuario desde el primer momento.

3.3 Mapa de selección de niveles.

Una vez seleccionada la opción de iniciar el juego, el jugador accede al **Mapa de selección de niveles**, donde se presenta una representación gráfica interactiva en **estilo pixel art**.

En este mapa:

- Se muestran distintos entornos temáticos, cada uno representando al menos un nivel jugable.

- Los niveles desbloqueados están claramente diferenciados de los niveles aún bloqueados.

- Cada nivel muestra información previa como número de estrellas alcanzadas o progreso realizado.

- Se fomenta la **interactividad**: el jugador puede seleccionar los niveles pulsando sobre ellos para iniciar el combate.

- El mapa integra pequeños detalles animados como barcos navegando o elementos decorativos dinámicos que refuerzan la sensación de mundo vivo.

El diseño del mapa facilita una navegación intuitiva y promueve la exploración progresiva del contenido del videojuego.

3.4 Sistema de combate.

El **sistema de combate** es el núcleo jugable del videojuego *Arcane Hunter*. Está compuesto por las siguientes funcionalidades:

- **Ataques básicos**: El jugador puede atacar utilizando comandos específicos, generando daño a los enemigos dentro del área de ataque.

- **Detección de colisiones**: Se detectan impactos entre el área de ataque del jugador y los enemigos a través de sistemas de colisión eficientes.

- **Manejo de daños**: Cada ataque exitoso reduce la salud del enemigo. Al llegar a cero, se activa la animación de muerte correspondiente.

- **Estados del jugador**: El jugador puede recibir daño de enemigos, activándose estados como invulnerabilidad temporal tras ser herido para evitar muertes injustas por acumulación de impactos.

- **Animaciones sincronizadas**: Cada acción de ataque, daño o muerte está acompañada de una animación específica que refuerza la inmersión.

El sistema ha sido diseñado priorizando la **responsividad** y la **fluidez**, asegurando combates dinámicos y satisfactorios.

3.5 Sistema de recogida.

Durante las fases de combate, los jugadores tienen la oportunidad de **recolectar monedas y runas** dispersas en el escenario:

- **Monedas**: Sirven como indicador de progreso económico dentro del juego. Se suman automáticamente al contador de monedas tras ser recogidas.

- **Runas**: Representan objetos especiales de colección, fundamentales para completar los logros y estadísticas del juego.

Cada vez que el jugador recoge un objeto:

- Se reproduce un efecto sonoro específico.
- Se actualizan inmediatamente los contadores visuales en pantalla.
- Se almacena la información correspondiente en los ficheros JSON internos para su persistencia.

El sistema de recogida está integrado de forma que **no interrumpe** el ritmo de combate, premiando la exploración y el riesgo controlado.

3.6 Pantallas de Game Over y Victoria.

Al finalizar un nivel, se despliega una pantalla específica dependiendo del resultado:

- **Pantalla de Game Over**:
  - Se activa si el jugador pierde toda su vida.
  - Muestra estadísticas tales como: número de monedas obtenidas, runas recogidas, tiempo jugado, número de estrellas alcanzadas (0 en caso de derrota) y nombre del enemigo que causó la muerte.
  - Ofrece dos opciones: **Reintentar el nivel** o **volver al mapa de selección de niveles**.
- **Pantalla de Victoria**:
  - Se muestra tras completar satisfactoriamente un nivel.
  - Incluye las mismas estadísticas que en Game Over, añadiendo animaciones de celebración del personaje jugador.

- Permite al jugador elegir entre **reanudar el avance** hacia otros niveles o **repetir** para mejorar su puntuación.

Ambas pantallas garantizan que el jugador reciba **retroalimentación inmediata** sobre su rendimiento.

3.7 Pantalla de estadísticas totales.

La **Pantalla de estadísticas totales** recoge de manera global los datos acumulados durante toda la trayectoria del jugador:

- Monedas totales recogidas.
- Runas totales obtenidas.
- Tiempo total de juego acumulado.
- Enemigos y jefes derrotados.
- Número total de muertes sufridas.

Esta pantalla está diseñada con una presentación clara y motivadora, permitiendo al jugador seguir su progreso, establecer nuevas metas personales o buscar la mejora continua de sus resultados.

Además, esta información es persistente entre sesiones gracias al uso de los ficheros JSON internos.

4. Implementación de la aplicación

La implementación de *Arcane Hunter* se ha llevado a cabo utilizando tecnologías específicas y metodologías de desarrollo que garantizan un producto final sólido, optimizado y acorde a los estándares actuales en el desarrollo de videojuegos 2D independientes.

4.1 Motor de desarrollo

La aplicación ha sido desarrollada en **Godot Engine versión 4.3**, un motor de videojuegos de código abierto ampliamente utilizado en la industria independiente debido a sus múltiples ventajas:

- **Ligereza y eficiencia**: Godot permite tiempos de carga rápidos y un bajo consumo de recursos, ideal para un juego de estilo retro pixel art.

- **Sistema de escenas**: Facilita la creación modular de niveles, personajes, menús e interfaces mediante una estructura basada en nodos reutilizables.

- **Animaciones integradas**: Ofrece herramientas avanzadas para el manejo de animaciones, físicas 2D y sistemas de partículas.

- **Multiplataforma**: Permite exportar el juego a múltiples plataformas como Windows, Linux, Android, entre otras.

4.2 Lenguaje de programación

El proyecto ha sido programado íntegramente en **GDScript**, el lenguaje nativo de Godot:

- **Sencillo y expresivo**: GDScript está diseñado para ser fácil de aprender y extremadamente eficiente para el desarrollo de videojuegos.

- **Integración directa**: Permite acceder de forma inmediata a todos los nodos y sistemas de Godot, acelerando el tiempo de desarrollo.

- **Alto rendimiento**: Su naturaleza ligera y su compatibilidad total con el motor garantizan un desempeño excelente.

Se han seguido principios de **programación modular**, **buena práctica de nombres** y **responsabilidad única**, estructurando el código en scripts individuales asociados a nodos específicos.

4.3 Control de datos

En lugar de utilizar bases de datos tradicionales, se ha optado por almacenar la configuración del usuario y el progreso del juego en **ficheros JSON internos**, lo cual resulta más apropiado para:

- **Proyectos locales y offline**: Donde no se requiere persistencia remota ni sincronización de datos online.
- **Simplicidad de acceso**: Permite la lectura y escritura de configuraciones y estadísticas de forma rápida y segura.
- **Flexibilidad**: Facilita modificaciones futuras en la estructura de los datos sin requerir migraciones complejas.

El formato JSON se utiliza para registrar:

- Volúmenes de audio.
- Idioma seleccionado.
- Activación o desactivación de efectos visuales.
- Estadísticas acumulativas del jugador.
- Estadísticas individuales por nivel.

Toda la lectura y escritura de estos ficheros se realiza de manera controlada para prevenir corrupciones de datos.

4.4 Estilo visual

El videojuego adopta una estética **Pixel Art** cuidadosamente diseñada:

- **Inspiración retro**: Emula el estilo visual clásico de los videojuegos de las décadas de los 80 y 90.
- **Colores vivos y definidos**: Que garantizan una experiencia gráfica agradable y coherente.
- **Recursos optimizados**: Se han utilizado assets gráficos ligeros para mantener un rendimiento fluido incluso en dispositivos de gama baja.

Los escenarios, personajes y elementos de interfaz respetan las proporciones y resolución típicas del pixel art, garantizando una coherencia visual en todo momento.

4.5 Sistema de animaciones

El sistema de animaciones se basa en el uso de **SpriteFrames** de Godot:

- **Spritesheet organizados**: Cada personaje y objeto animado cuenta con su propio conjunto de sprites perfectamente ordenado.
- **Animaciones fluidas**: Cada acción (correr, atacar, saltar, recibir daño, morir, etc.) está asociada a su animación correspondiente.
- **Control centralizado**: Se utilizan nodos **AnimatedSprite2D** para gestionar de manera eficiente el cambio de animaciones en respuesta a eventos del juego.
- **Sincronización**: Las animaciones están sincronizadas con efectos sonoros, colisiones y cambios de estado.

4.6 Colisiones

La detección de colisiones se gestiona mediante la combinación de **áreas** (**Area2D**) y **cuerpos físicos** (**CharacterBody2D**, **StaticBody2D**):

- **Áreas**: Utilizadas para detectar eventos de interacción como recogida de objetos o zonas de daño.
- **Cuerpos físicos**: Empleados para representar entidades sólidas como el jugador, enemigos o estructuras del entorno.
- **Formas de colisión optimizadas**: Cada área o cuerpo físico cuenta con formas de colisión (**CollisionShape2D**) adaptadas a su sprite, mejorando la precisión sin afectar negativamente al rendimiento.
- **Capas y máscaras**: Se han configurado adecuadamente para asegurar que solo se detecten las colisiones necesarias en cada caso.

Esto proporciona un comportamiento físico consistente y esperado, crucial para la jugabilidad.

4.7 Sistema de audio

El apartado sonoro del juego se gestiona a través de:

- **Efectos de sonido**: Asociados a acciones como ataques, daños, recogida de objetos, saltos, muertes, entre otros.
- **Música de fondo**: Temas musicales que ambientan tanto el menú principal como las distintas fases de juego.
- **Control de volumen**: El volumen general y de los efectos puede ajustarse mediante las opciones del menú de configuración.

Se utilizan nodos **AudioStreamPlayer2D** para efectos de sonido localizados espacialmente y **AudioStreamPlayer** para música de fondo global.

El sistema de audio respeta las decisiones del usuario guardadas en los ficheros JSON, aplicando automáticamente la configuración de volumen establecida.

4.8 Interfaz de usuario (UI)

Se han diseñado y programado elementos de **interfaz de usuario intuitivos e interactivos**:

- **Botones con retroalimentación visual**: Cambian de estado al ser seleccionados o presionados.
- **Etiquetas y contadores**: Visualizan información relevante como monedas obtenidas, runas recolectadas o vida restante.
- **Pantallas de transición**: Informan sobre el estado del jugador (victoria o derrota) de manera clara y motivadora.
- **Menús modales**: Ajustes, ayuda y estadísticas se presentan mediante ventanas emergentes que no rompen la inmersión.
- **Accesibilidad**: Se ha cuidado la legibilidad del texto, el tamaño de los botones y la disposición de los elementos para garantizar su uso en distintos tipos de pantallas y resoluciones.

Toda la UI mantiene la coherencia estética con el resto del diseño gráfico basado en pixel art, contribuyendo a una experiencia de usuario sólida y agradable.

5. Interfaz de usuario

La interfaz de usuario (*UI*) de *Arcane Hunter* ha sido cuidadosamente diseñada para ofrecer una experiencia intuitiva, agradable y coherente con la estética pixel art del videojuego. Se prioriza la accesibilidad, la claridad visual y la facilidad de uso en todos los

menús y elementos visuales, asegurando que cualquier tipo de jugador pueda interactuar con el sistema de manera natural.

5.1 Descripción de la interfaz

La estructura de la interfaz principal está organizada en distintas secciones o momentos de interacción con el usuario, adaptadas al flujo del videojuego:

Menú de inicio

- **Botones grandes y accesibles**: Cada opción disponible (Iniciar, Opciones, Estadísticas, Salir) se presenta mediante botones de gran tamaño, claramente diferenciados por su iconografía y tipografía retro.

- **Diseño visual coherente**: El fondo, los marcos y los botones utilizan elementos visuales pixelados que mantienen la inmersión estética.

- **Accesibilidad inmediata**: Las opciones principales son visibles desde el primer momento, sin necesidad de desplazarse o navegar excesivamente.

Pantalla de selección de nivel

- **Uso de iconos representativos**: Cada nivel se representa mediante un icono distintivo sobre el mapa, acompañado de indicadores como estrellas obtenidas y nombre o número de nivel.

- **Caminos de navegación**: El diseño incluye caminos o rutas visuales que guían al jugador de manera natural por el orden de los niveles disponibles, facilitando la progresión.

- **Acceso condicional**: Los niveles bloqueados o aún no superados muestran un icono distinto o un color desaturado para indicar su estado.

HUD en niveles

Durante la partida, el jugador dispone de un **HUD (Heads-Up Display)** compacto y funcional que incluye:

- **Indicador de monedas**: Número de monedas recogidas en el nivel.
- **Indicador de runas**: Cantidad de runas recolectadas hasta el momento.
- **Barra de vida**: Representación visual de la salud actual del jugador mediante una barra de progreso de estilo pixel art.
- **Indicador de tiempo**: Cronómetro que mide el tiempo transcurrido en el nivel, fundamental para calcular puntuaciones o recompensas.

Este HUD se encuentra discretamente ubicado para no saturar la pantalla, pero ofrece la información necesaria de un vistazo rápido.

Menú de pausa

Al pulsar la tecla correspondiente, se accede a un menú de pausa que presenta las siguientes opciones:

- **Reanudar partida**: Permite continuar el nivel desde el mismo punto en el que se pausó.
- **Reiniciar nivel**: Reinicia el nivel actual desde su comienzo.
- **Salir al menú de selección**: Permite abandonar el nivel actual y regresar al mapa de selección de niveles.

El menú de pausa mantiene la coherencia gráfica con el resto de la interfaz y garantiza que las acciones sean intuitivas y claras, incluso en situaciones de estrés o presión durante el juego.

5.2 Manual de usuario

La interacción con el videojuego es simple, accesible y diseñada para garantizar una curva de aprendizaje rápida incluso para usuarios sin experiencia previa en videojuegos de plataformas o acción.

A continuación, se detallan los principales controles:

Movimiento

- **Teclas de dirección (A/W/D)**:
  Permiten mover al personaje hacia la izquierda, derecha o realizar saltos verticales según corresponda.

Ataque

- **Botón izquierdo del ratón**:
  Realiza el ataque principal del personaje. Dependiendo de las circunstancias del combate, se adapta para golpear enemigos o interactuar con elementos del entorno.

Recoger objeto

- **Automático al contacto**:
  No se requiere ninguna acción adicional para recoger monedas o runas. Al entrar en contacto físico con el objeto, este se suma automáticamente al contador.

Pausa

- **Tecla Escape**:
  Abre el menú de pausa en cualquier momento durante una partida activa.

Interacción

- **Botones en pantalla o atajos de teclado**:
  En menús y pantallas de interfaz, el usuario puede interactuar tanto con el

ratón (haciendo clic sobre botones) como utilizando atajos de teclado (por ejemplo, teclas de dirección y Enter para confirmar).

Este conjunto de controles garantiza una experiencia accesible para la mayoría de jugadores, manteniendo la simplicidad sin sacrificar el dinamismo del juego.

6. Aspectos destacables del desarrollo

6.1 Implementación de parallax avanzado

Para aumentar la sensación de inmersión en el entorno del juego, se implementó un **efecto de parallax avanzado**. Esto se consiguió mediante el nodo **ParallaxLayer** de Godot 4.3, configurando cada capa con un **factor de movimiento** distinto. Las capas más cercanas se desplazan más rápidamente respecto a la cámara, mientras que las capas más lejanas lo hacen más lentamente, simulando profundidad en escenarios bidimensionales.

Se distribuyeron múltiples fondos (cielo, montañas, árboles y suelos) en distintas capas de **ParallaxLayer**, respetando una coherencia visual que aporta dinamismo y realismo a los desplazamientos del jugador.

6.2 Sistema de trueno con efectos de destello sincronizados con sonido

Otro punto clave fue la creación de un **sistema de truenos** que combina animaciones, efectos visuales y sonido.

El trueno se dispara de manera **aleatoria** cada ciertos segundos (entre 1 y 10 segundos), mediante el siguiente flujo:

- Animación del rayo (**AnimatedSprite2D**) reproduciendo 6 frames.
- Destello blanco inicial (**ColorRect**) seguido de un degradado hacia azul claro usando interpolaciones (**Tween**).

- Sonido del trueno (**AudioStreamPlayer**) con un pequeño retardo respecto al destello para simular la física real.

Este comportamiento es coordinado de manera asíncrona para que el rayo no siempre caiga en momentos predecibles, otorgando realismo meteorológico.

6.3 Programación de enemigos y jefes con mecánicas especiales

El videojuego presenta enemigos básicos (gorgonas, minotauros) y un **jefe final** que tiene un comportamiento mucho más complejo:

- **IA básica**: Los enemigos detectan colisiones, patrullan el terreno, atacan al jugador mediante un **Area2D** y reaccionan al daño recibido.

- **IA avanzada del jefe**: El jefe realiza ataques variados, reacciona con burla al hacer daño y sincroniza animaciones especiales de ataque, daño y muerte.

Este sistema de enemigos se diseñó usando patrones de detección (**Area2D** y **RayCast2D**) y estados internos como **is_attacking**, **esta_herido** e **is_dead**.

6.4 Desarrollo de un mapa de niveles interactivo, integrando nodos y rutas desbloqueables

Se implementó un **mapa de selección de niveles** con:

- Iconos representativos de cada entorno o nivel.

- Rutas dibujadas que simulan caminos entre niveles.

- Sistema de desbloqueo de niveles basado en progreso del jugador.

Esto no solo facilita la navegación, sino que también mejora la presentación visual del progreso del jugador de manera muy intuitiva y clara.

6.5 Optimización del rendimiento en efectos visuales

Para asegurar que el videojuego pudiera ejecutarse en una amplia gama de dispositivos, se llevaron a cabo varias optimizaciones:

- **Uso eficiente de partículas**: Las partículas como la lluvia, sangre o polvo se limitaron en cantidad y área de emisión.
- **Animaciones ligeras**: Se recortaron spritesheets para optimizar el tamaño de memoria.
- **Control de audio**: Se instanciaron audios bajo demanda y se liberaron cuando ya no eran necesarios.

Además, se mantuvo siempre la prioridad de **no cargar escenas pesadas de fondo** innecesariamente.

6.6 Uso de JSON para configuraciones

Se eligió trabajar con **ficheros JSON** para guardar:

- Ajustes de usuario (volumen, música, efectos gráficos).
- Progreso en los niveles (niveles desbloqueados, monedas y runas conseguidas).
- Estadísticas totales acumuladas.

Este enfoque simplifica el guardado y la carga de información de forma rápida y directa sin tener que recurrir a bases de datos complejas.

Ejemplo de implementación de cofres con generación aleatoria

Una de las mecánicas implementadas que merece ser destacada es el comportamiento de los **cofres** que generan monedas y runas de manera aleatoria. El código funcional es el siguiente:

```
func soltar_monedas():
    var cantidad = randi_range(min_monedas, max_monedas)
    for i in range(cantidad):
        var moneda = moneda_escena.instantiate() as RigidBody2D
        get_tree().current_scene.add_child(moneda)
        moneda.global_position = global_position + Vector2(randf_range(-10, 10), -10)
        moneda.apply_impulse(Vector2(randf_range(-100, 100), -150))
```

Este fragmento refleja:

- **Randomización controlada** de número y posición de monedas.
- **Física realista** con impulsos aplicados tras la apertura del cofre.
- **Modularidad** para adaptarse a cualquier tipo de objeto recolectable.

Este diseño permite que cada cofre entregue una experiencia diferente al jugador, aumentando la rejugabilidad y la sensación de recompensa.

7. Líneas de trabajo futuro

A pesar del importante grado de avance en el desarrollo de **Arcane Hunter**, se plantean diversas líneas de trabajo a futuro para expandir y enriquecer la experiencia de juego, tanto en contenidos como en accesibilidad:

7.1 Implementación de logros internos

Se pretende incluir un **sistema de logros** que premie hitos conseguidos por el jugador, tales como:

- Derrotar a un número determinado de enemigos.
- Acumular grandes cantidades de monedas o runas.
- Completar niveles en un tiempo récord.
- Superar niveles sin recibir daño.

Estos logros, almacenados también en ficheros JSON, potenciarían la rejugabilidad y motivarían al usuario a explorar el 100% del contenido.

7.2 Expansión de niveles y nuevos tipos de enemigos

Actualmente el juego cuenta con un conjunto de niveles base, pero se planea:

- **Añadir nuevos entornos** temáticos (desiertos, castillos, cavernas heladas, etc.).
- **Introducir enemigos inéditos**, con patrones de ataque y mecánicas diferentes (enemigos voladores, enemigos inmunes a ciertos ataques, enemigos de emboscada, entre otros).
- **Crear nuevos jefes finales** que exijan habilidades tácticas más complejas.

Esto garantizaría que el jugador encuentre siempre desafíos renovados.

7.3 Traducción completa a múltiples idiomas

Actualmente el juego soporta español de forma básica. Se plantea:

- Extender la localización a otros idiomas como inglés.
- Usar sistemas de internacionalización que permitan cargar los textos dinámicamente desde archivos externos.
- Adaptar correctamente los formatos de fecha, hora o monedas dependiendo del idioma seleccionado.

Con ello se abriría el mercado a un público internacional mucho más amplio.

7.4 Implementación de secretos, mecánicas y mapeado

Para mejorar la experiencia del jugador con Arcane Hunter, se plantean varios puntos:

- Crear rutas y logros secretos desbloqueables mediante el progreso del jugador.
- Crear mecánicas de entorno como minería, balanceo o escalada.
- Crear un mapa de nivel donde el jugador sea capaz de guiarse.

7.5 Sistema de habilidades y mejoras

Se plantea agregar un **sistema de progresión de personaje**, que permita:

- Desbloquear habilidades especiales (doble salto, ataques cargados, escudos temporales, etc.).
- Mejorar estadísticas como velocidad, ataque o vida máxima.

Estas mejoras se podrían adquirir mediante monedas o runas, añadiendo una capa adicional de estrategia a la experiencia de juego.

7.6 Mejoras en la accesibilidad

Para hacer **Arcane Hunter** accesible a la mayor cantidad posible de jugadores, se consideran futuras mejoras como:

- Inclusión de un **modo daltonismo**.
- Opciones para **remapear controles** a elección del jugador.
- **Ajustes de dificultad** para personas con discapacidades motrices o cognitivas.
- Subtítulos y descripciones para efectos de sonido importantes.

El objetivo es garantizar que cualquier usuario pueda disfrutar del juego sin barreras.

8. Ciclo de vida y desarrollo del videojuego

8.1 Planificación del proyecto

Desde el inicio del desarrollo de *Arcane Hunter*, se estableció una **planificación estructurada** que permitiera organizar adecuadamente todas las fases del proyecto. Esta planificación no solo respondía a una necesidad académica, sino que resultó esencial para poder abordar el desarrollo de un videojuego funcional y completo en un periodo de tiempo limitado.

Para ello, se elaboró un **cronograma de trabajo semanal (Anexo III)**, dividiendo el proyecto en fases y tareas específicas: diseño del sistema de combate, creación de enemigos, diseño de niveles, implementación de estadísticas, pruebas, y creación de la interfaz, entre otras. Cada tarea fue asignada a una o varias semanas en función de su complejidad y su dependencia de otras funcionalidades ya implementadas.

Durante esta etapa también se establecieron los **objetivos generales y específicos** del videojuego, incluyendo tanto metas técnicas (por ejemplo, la integración de estadísticas mediante archivos JSON o el uso del motor Godot 4.3) como metas de diseño (una experiencia retro, un sistema de combate fluido, jefes con IA avanzada, etc.).

Esta planificación inicial permitió mantener una **ruta de desarrollo clara y ordenada**, facilitando la toma de decisiones y la priorización de tareas a medida que el proyecto avanzaba.

8.2 Análisis de requisitos

Tal y como se detalla en el **apartado 2.1 del presente documento**, el análisis del videojuego consistió en definir qué elementos debía contener para cumplir sus objetivos técnicos y jugables. Este análisis abarcó tanto aspectos funcionales como no funcionales, teniendo en cuenta el público destino y las condiciones de uso previstas.

Se definieron funcionalidades clave como el sistema de combate, el control de estadísticas, el guardado local en JSON o la posibilidad de configurar aspectos del juego desde un menú de ajustes.

Además, se establecieron restricciones técnicas y de diseño que permitieran mantener el rendimiento y la escalabilidad del proyecto, tal y como se justifica también en el **apartado 3.1**.

8.3 Diseño del sistema

El diseño general del sistema, desarrollado y descrito con detalle en el **capítulo 4**, se fundamenta en una arquitectura modular y escalable, basada en escenas independientes con nodos personalizados para cada tipo de objeto o enemigo.

Se diseñaron escenas para cada pantalla (inicio, ajustes, estadísticas, victoria, derrota...) y se desarrollaron scripts separados para el comportamiento de los enemigos, el jugador, los cofres, el jefe final, entre otros. Esta modularidad permitió realizar pruebas independientes por componente y facilitó la depuración.

El diseño también integró elementos visuales coherentes con el estilo retro del proyecto, y se centró en ofrecer una interfaz de usuario accesible y clara, como se puede consultar con más detalle en el **capítulo 5**.

8.4 Implementación progresiva (modelo incremental)

Una vez definidas las bases del diseño, se procedió a implementar el videojuego de forma **progresiva e incremental**. Aunque el proyecto había sido planificado en fases, durante

la programación se adoptó una metodología **iterativa**, en la que cada bloque funcional se desarrollaba, probaba y mejoraba antes de pasar al siguiente.

Este enfoque permitió detectar errores de forma temprana y aplicar ajustes sobre la marcha, sin comprometer la estabilidad del resto del sistema.

Por ejemplo:

- Se comenzó desarrollando el movimiento del jugador, que posteriormente se complementó con el sistema de ataque, la detección de enemigos y las animaciones de daño.

- El sistema de estadísticas se implementó primero con contadores básicos y luego se expandió para incluir valores más complejos como estrellas por nivel, tiempo acumulado o número de muertes.

- Los menús de configuración y estadísticas fueron evolucionando en paralelo con la programación del HUD y la integración de los JSON.

Gracias a este enfoque incremental, fue posible **probar cada componente de manera aislada**, y una vez validado, integrarlo con el resto del sistema sin introducir errores globales.

Además, se aprovecharon las funcionalidades del editor de Godot para agilizar la iteración, como la recarga rápida de escenas, el inspector de nodos y el sistema de señales para eventos.

8.5 Pruebas de validación

El proceso de desarrollo de *Arcane Hunter* no se limitó únicamente a implementar funcionalidades. También se aplicó un enfoque riguroso de **validación continua**, cuyo objetivo fue garantizar que el videojuego cumpliera con los requisitos definidos y que su funcionamiento fuera correcto en todas las situaciones posibles.

Durante todo el desarrollo se llevaron a cabo **pruebas de validación manuales** de dos tipos: **pruebas de caja negra** y **pruebas de caja blanca**. Estas permitieron analizar el comportamiento del sistema desde distintos enfoques, contribuyendo a la mejora constante del código y de la experiencia de usuario.

8.5.1 Introducción a las pruebas

Las pruebas se realizaron de forma **paralela al desarrollo**, no solo al final del proceso. Esta metodología permitió **detectar errores tempranamente** y aplicar soluciones inmediatas, evitando acumulación de fallos en fases avanzadas.

Además, se utilizaron distintas herramientas de depuración integradas en el editor de Godot, como:

- **Mensajes de traza (print, print_debug)** para seguir el flujo del código.
- **Breakpoints** para pausar la ejecución en puntos concretos y comprobar el estado interno del sistema.
- **Inspección visual** mediante el árbol de nodos en tiempo de ejecución.
- **Testeo manual** de los distintos escenarios, menús y situaciones del juego.

Este enfoque combinó tanto el análisis interno del código como la verificación externa desde el punto de vista del jugador.

8.5.2 Pruebas de caja negra

Las pruebas de **caja negra** se enfocan en verificar que el comportamiento del videojuego fuera el esperado, **sin tener en cuenta cómo está implementado internamente el código**.

Estas pruebas se centraron en la interacción del usuario con el sistema, los valores visibles en pantalla, las transiciones entre escenas y la respuesta del juego ante distintas acciones.

Algunos ejemplos de pruebas de caja negra realizadas:

- **Menú de ajustes**: Se comprobó que al modificar los sliders de volumen, los cambios se reflejaban correctamente en el audio del juego.
- **Pantalla de Game Over**: Se validó que esta apareciera únicamente al morir el jugador, mostrando correctamente las estadísticas del nivel.
- **Pantalla de victoria**: Se verificó que se mostraran correctamente los datos al finalizar con éxito un nivel, incluyendo la cantidad de estrellas alcanzadas.
- **Menú de selección de idioma**: Se evaluó que los textos cambiaran al seleccionar otro idioma, afectando tanto los botones como las etiquetas del HUD.
- **Sistema de recogida**: Se confirmó que al recoger monedas o runas, los contadores en pantalla se actualizan de forma inmediata.

Estas pruebas permitieron detectar problemas visuales, errores de navegación y valores que no se actualizaban correctamente en determinados estados del juego.

8.5.3 Pruebas de caja blanca

Las pruebas de **caja blanca** se centraron en analizar el funcionamiento interno del código fuente, comprobando las condiciones lógicas, estructuras de control, bucles y llamadas a funciones.

Estas pruebas fueron esenciales para validar partes del juego que no siempre se ven directamente en pantalla, pero que influyen en el resultado.

Ejemplos de pruebas de caja blanca realizadas:

- **Cálculo de puntuación**: Se revisó el método encargado de calcular las estrellas obtenidas en función de monedas, runas y tiempo. Se validaron las distintas condiciones y combinaciones para asegurar que el cálculo era justo y coherente.

- **Reanimación del Esqueleto**: Se verificó que, al morir un enemigo de tipo esqueleto, se revivía correctamente hasta ser rematado en el suelo por el jugador.

- **Sistema de daño y animación**: Se comprobó que al recibir daño, tanto el jugador como los enemigos entraran en estado de "herido", reproduciendo la animación correspondiente sin interrumpir otras funciones.

- **IA del jefe final**: Se analizaron los árboles de decisión internos que controlaban los ataques del jefe según la distancia al jugador, asegurando que las decisiones se ejecutaran correctamente.

- **Control de vida máxima**: Se probó que el jugador no pudiera recuperar más vida de la permitida tras eliminar enemigos o recoger objetos curativos.

Este tipo de pruebas resultó especialmente útil para detectar **errores lógicos o condiciones mal definidas** que podrían haber generado comportamientos erróneos o inconsistentes en determinadas situaciones.

8.5.4 Herramientas empleadas durante la validación

Para realizar todas estas pruebas, se emplearon diferentes herramientas y recursos disponibles en el entorno de desarrollo:

- **Mensajes de consola (print(), print_debug())**: Permitieron seguir paso a paso el comportamiento de funciones críticas, como la generación de enemigos o la actualización de estadísticas.

- **Breakpoints**: Utilizados para detener la ejecución en tiempo real y comprobar el estado de las variables y nodos involucrados.

- **Panel de depuración de Godot**: Facilitó el análisis visual de la escena, permitiendo ver nodos ocultos, áreas de colisión activas y nodos en ejecución.

- **Control de flujo manual**: En determinadas funciones, se implementaron contadores o banderas temporales para asegurarse de que una acción se ejecutara solo una vez (por ejemplo, la animación de muerte del jefe).

Estas herramientas fueron aplicadas de manera combinada en sesiones de validación periódicas, asegurando que tanto las funcionalidades nuevas como las ya implementadas se comportaran correctamente.

8.5.5 Conclusión de la fase de pruebas

La aplicación de pruebas de validación en *Arcane Hunter* fue un proceso constante y necesario, que acompañó todo el desarrollo del proyecto. La combinación de pruebas de caja negra (externas) y caja blanca (internas) permitió asegurar no solo que el videojuego funcionara correctamente, sino que también ofreciera una experiencia de usuario satisfactoria, fluida y libre de errores críticos.

Gracias a este proceso de control, se lograron corregir comportamientos inesperados, optimizar el rendimiento en ciertas escenas, y afinar detalles jugables que mejoraron notablemente el producto final.

9. Conclusiones

El desarrollo de *Arcane Hunter* hasta la fecha ha permitido validar la viabilidad técnica y artística del proyecto planteado inicialmente. Se ha logrado transformar una propuesta conceptual en una aplicación interactiva que, aunque aún se encuentra en proceso de expansión, presenta una estructura sólida y coherente.

A nivel técnico, la elección del motor **Godot Engine 4.3** y del lenguaje **GDScript** ha permitido implementar mecánicas complejas como sistemas de combate dinámicos, gestión de estadísticas, control de niveles mediante ficheros JSON, y un entorno visual enriquecido por animaciones detalladas y efectos climáticos dinámicos. El enfoque modular en la programación ha favorecido la escalabilidad y el mantenimiento del proyecto, facilitando futuras expansiones como la inclusión de logros, nuevas áreas y modalidades multijugador.

Desde el punto de vista artístico y de diseño, la integración de assets de alta calidad en estilo **pixel art** ha proporcionado una coherencia estética entre la jugabilidad y la ambientación narrativa del juego. Asimismo, el diseño de interfaz de usuario se ha centrado en la claridad, la simplicidad y la funcionalidad, respetando principios de accesibilidad para mejorar la experiencia del jugador.

La metodología de trabajo seguida —basada en una planificación iterativa, el aprovechamiento de recursos gratuitos/licenciados y una documentación continua— ha permitido optimizar tiempos y recursos, adaptándose a los cambios y retos surgidos durante el proceso de desarrollo.

10. Bibliografía/Enlaces de interés.

A continuación se recogen todas las referencias y fuentes de recursos empleadas en el desarrollo del proyecto **Arcane Hunter**:

- CraftPix.net. (2025). *Free and Premium Game Assets*. Recuperado de: https://craftpix.net/

- Itch.io. (2025). *Asset Packs for Indie Games*. Recuperado de: https://itch.io/

- OpenGameArt.org. (2025). *Free Game Assets*. Recuperado de: https://opengameart.org/

- Godot Engine Documentation. (2025). *Godot Engine Official Documentation*. Recuperado de: https://docs.godotengine.org/en/stable/

- Pixabay. (2025). *Free Resource Assets*. Recuperado de: https://pixabay.com/es/

- Enlaces adicionales recogidos en el documento "**ENLACES_TFG.txt**" (ver Anexos).

Estos enlaces han sido utilizados tanto para adquirir **assets** gráficos y sonoros de uso libre o bajo licencia, como para consultar documentación técnica y guías de buenas prácticas en el desarrollo de videojuegos con **Godot Engine**.

11. Anexos.

I. Recursos usados.

A. Personajes.

- CraftPix.net. (2025). *Free Street Animal Pixel Art Asset Pack*. https://craftpix.net/freebies/free-street-animal-pixel-art-asset-pack/

- CraftPix.net. (2025). *Free Skeleton Pixel Art Sprite Sheets*. https://craftpix.net/freebies/free-skeleton-pixel-art-sprite-sheets/

- CraftPix.net. (2025). *Free Desert Enemy Sprite Sheets Pixel Art*. https://craftpix.net/freebies/free-desert-enemy-sprite-sheets-pixel-art/

- CraftPix.net. (2025). *Snow Enemy Character Sprites Pixel Art*. https://craftpix.net/product/snow-enemy-character-sprites-pixel-art/

- CraftPix.net. (2025). *Octopus, Jellyfish, Shark and Turtle Free Sprite Pixel Art*. https://craftpix.net/freebies/octopus-jellyfish-shark-and-turtle-free-sprite-pixel-art/

- CraftPix.net. (2025). *Free Underwater Enemies Pixel Art Character Pack*.

https://craftpix.net/freebies/free-underwater-enemies-pixel-art-character-pack/

- CraftPix.net. (2025). *Swamp Enemies Sprite Sheets Pixel Art*. https://craftpix.net/product/swamp-enemies-sprite-sheets-pixel-art/

- CraftPix.net. (2025). *Free Fishing Game Assets Pixel Art Pack*. https://craftpix.net/freebies/free-fishing-game-assets-pixel-art-pack/

- CraftPix.net. (2025). *Free 3 Character Sprite Sheets Pixel Art*. https://craftpix.net/freebies/free-3-character-sprite-sheets-pixel-art/

- CraftPix.net. (2025). *Free Swamp Bosses Pixel Art Character Pack*. https://craftpix.net/freebies/free-swamp-bosses-pixel-art-character-pack/

- CraftPix.net. (2025). *Water Monsters Pixel Art Sprite Sheet Pack*. https://craftpix.net/product/water-monsters-pixel-art-sprite-sheet-pack/

- CraftPix.net. (2025). *Snow Bosses Game Character Pixel Art Pack*. https://craftpix.net/product/snow-bosses-game-character-pixel-art-pack/

- CraftPix.net. (2025). *Cave Bosses Pixel Art Game Sprites*. https://craftpix.net/product/cave-bosses-pixel-art-game-sprites/

- CraftPix.net. (2025). *Desert Bosses Pixel Art Character Pack*. https://craftpix.net/product/desert-bosses-pixel-art-character-pack/

B. Tilesets

- CraftPix.net. (2025). *Desert 2D Tileset Pixel Art*. https://craftpix.net/product/desert-2d-tileset-pixel-art/

- CraftPix.net. (2025). *Water Pack Tileset for Platformer Pixel Art*. https://craftpix.net/product/water-pack-tileset-for-platformer-pixel-art/

- CraftPix.net. (2025). *Snow 2D Game Tileset Pixel Art*. https://craftpix.net/product/snow-2d-game-tileset-pixel-art/

- CraftPix.net. (2025). *Platformer Pixel Art Cave 2D Tileset*. https://craftpix.net/product/platformer-pixel-art-cave-2d-tileset/

- CraftPix.net. (2025). *Free Swamp Game Tileset Pixel Art*. https://craftpix.net/freebies/free-swamp-game-tileset-pixel-art/

C. Retratos de Jefes (Portraits)

- CraftPix.net. (2025). *Bosses Portraits for Dialogues Pixel Art*. https://craftpix.net/product/bosses-portraits-for-dialogues-pixel-art/

- CraftPix.net. (2025). *Free Boss Portrait Pack 6*. https://craftpix.net/freebies/free-boss-portrait-pack-6/

- CraftPix.net. (2025). *Monster Portraits Pixel Art*. https://craftpix.net/product/monster-portraits-pixel-art/

- CraftPix.net. (2025). *Monster Portraits with Emotions*. https://craftpix.net/product/monster-portraits-with-emotions/

D. Efectos

- CraftPix.net. (2025). *11 Free Pixel Art Explosion Sprites*. https://craftpix.net/freebies/11-free-pixel-art-explosion-sprites/

- CraftPix.net. (2025). *Weather Effects Pixel Art Assets Pack*. https://craftpix.net/product/weather-effects-pixel-art-assets-pack/

E. Interfaz de Usuario

- CraftPix.net. (2025). *Game User Interface Pixel Art*. https://craftpix.net/product/game-user-interface-pixel-art/

- DanoPix. (2025). *Health Bar by Dano*. https://danopix.itch.io/health-bar-by-dano

- dafont.com. (2025). *Pixellari Font*. https://www.dafont.com/pixellari.font

F. Sonido

- Pixabay. (2025). *Free Sound Effects*. https://pixabay.com/es/sound-effects/search/

II. Imágenes

- Story Board.

1

- Un caballero medieval se prepara para su viaje en un mundo de fantasía
- Plano general
- Introducción del protagonista

2

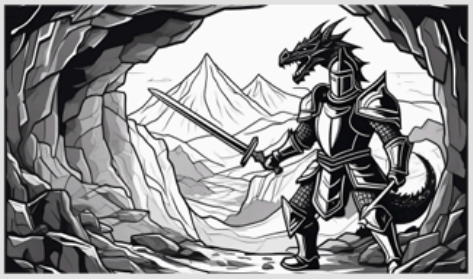
- El caballero se enfrenta a su primer enemigo en un bosque oscuro
- Plano medio
- Primer encuentro con el enemigo

3

- El caballero explora un castillo antiguo en busca de pistas
- Plano detalle
- Exploración y descubrimiento

4

- El caballero encuentra un mapa que revela la ubicación de su amada
- Plano detalle
- Momento de descubrimiento crucial



5

- El caballero lucha contra un dragón en una cueva llena de lava
- Plano general
- Batalla épica

6

- El caballero rescata a su amada y escapan juntos
- Plano medio
- Resolución y final feliz

III. Cronograma

Semana	Actividades desarrolladas
1/4/2025 - 6/4/2025	Desarrollo inicial del videojuego
7/4/2025 - 13/4/2025	Desarrollo de lógica restante del nivel 1.
14/4/2025 - 20/4/2025	Desarrollo de la interfaz de usuario.
21/4/2025 - 27/4/2025	Implementación de ficheros JSON para guardar datos.
28/4/2025 - 4/5/2025	Implementación de jefes y nueva lógica de desarrollo del nivel 1.
5/5/2025 - 11/5/2025	Ordenación de código.
12/5/2025 - 18/5/2025	Desarrollo del nivel 2 con nuevas mecánicas, entorno accesible e interactivo y enemigos letales.
19/5/2025 - 25/5/2025	Desarrollo de la lógica restante del nivel 2 e implementación de modo multijugador (2 jugadores).
26/5/2025 - 1/6/2025	Corrección de errores, optimización de recursos y mejora de aspectos del videojuego.
2/6/2025 - 8/6/2025	Planificación de la presentación del videojuego.

IV. Repositorio.

- Repositorio.
  https://github.com/erikdaniel2004/Arcane-Hunter.git