Institut Puig Castellar
Santa Coloma de Gramenet

FAETERNA

(Projecte de desenvolupament)
CFGM Desenvolupament de Videojocs
i Realitat Virtual

Arnau Baena, Sergi Flores
Haolin Zheng y Wenkai Zhou

DEV

Resumen del proyecto:

Este proyecto tiene como temática principal el desarrollo de un videojuego que combina el género Metroidvania con un sistema de gestión de recursos, creando una experiencia que equilibra la exploración y el combate con la construcción y el crecimiento estratégico. El jugador puede cambiar entre diferentes personajes, cada uno con habilidades únicas que permiten superar obstáculos y descubrir zonas secretas, aportando dinamismo y profundidad a la exploración.

El objetivo principal del proyecto es diseñar e implementar un juego que conecte de manera coherente la parte de acción con la gestión de recursos, integrándolos a través de la narrativa centrada en el “Árbol Huluppu”, un elemento clave que crece a medida que el jugador avanza, desbloqueando nuevas áreas y habilidades. Este vínculo busca ofrecer una experiencia de juego inmersiva donde cada acción tiene repercusión en el desarrollo global.

Para conseguir este objetivo, hemos seguido una metodología basada en el uso del motor Godot para el desarrollo, combinada con control de versiones y trabajo colaborativo mediante GitHub. También hemos implementado pruebas constantes para garantizar la calidad del juego, así como sesiones de análisis para ajustar el balance entre las mecánicas de exploración y gestión.

Las conclusiones hasta ahora indican que esta combinación de géneros y mecánicas permite crear una experiencia rica y atractiva, con un ritmo que alterna momentos de acción intensa y pausa estratégica, proporcionando a los jugadores una motivación constante para explorar y desarrollar su entorno.

Palabras clave:

Faeterna

Videojuegos de gestión de recursos

Videojuegos metroidvania

Kotasan Studio

Kotasan

Juegos Kotasan

Abstract

The main theme of this project is the development of a video game that combines the Metroidvania genre with a resource management system, creating an experience that balances exploration, combat, and strategic growth. Players can switch between different characters, each with unique abilities that help overcome obstacles and uncover hidden areas, adding depth and variety to the gameplay.

The project's main objective is to design and implement a game that seamlessly connects action and management mechanics through a narrative centered on the "Huluppu Tree." This central element evolves alongside the player's progress, unlocking new areas, abilities, and gameplay possibilities. The goal is to create an immersive experience in which the player's actions directly influence the world's development.

To achieve this, the project is being developed using the Godot engine, alongside collaborative workflows and version control through GitHub. Continuous testing and balancing sessions are carried out throughout development to refine the interaction between exploration, combat, and management systems.

Current results suggest that the combination of these mechanics creates an engaging gameplay loop, alternating between moments of intense action and strategic planning while maintaining the player's motivation to explore and expand their environment.

Keywords (between 4 and 8):

Faeterna

Resource management games

Metroidvania games

Kotasan Studio

Kotasan

Kotasan games

Índice

1 Introducción

Este proyecto consiste en el desarrollo de un videojuego del género Metroidvania que incorpora mecánicas de gestión de recursos y construcción estratégica, con el objetivo de ofrecer una experiencia que combine exploración, combate y progresión dinámica. A lo largo de la aventura, el jugador podrá alternar entre distintos personajes, cada uno con habilidades únicas que permitirán acceder a nuevas zonas, resolver obstáculos y descubrir secretos ocultos dentro del mundo del juego.

La propuesta busca diferenciarse mediante la integración de un sistema de crecimiento vinculado al “Árbol Huluppu”, un elemento central tanto a nivel narrativo como jugable. A medida que el jugador avanza, recolecta recursos y supera desafíos, el árbol evoluciona y desbloquea nuevas habilidades, áreas y posibilidades de exploración, generando una conexión directa entre las acciones del jugador y la evolución del entorno.

El desarrollo del proyecto se realiza utilizando el motor Godot junto con programación en C#, aplicando metodologías de trabajo colaborativo mediante GitHub para el control de versiones y la organización del desarrollo. Además, se llevan a cabo pruebas constantes y ajustes de equilibrio para garantizar una experiencia fluida y satisfactoria.

Con esta combinación de géneros y sistemas, el proyecto pretende crear un ritmo de juego variado, alternando momentos de acción intensa con fases de planificación y gestión, incentivando al jugador a explorar, mejorar sus capacidades y expandir progresivamente el mundo que lo rodea.

1.1 Contexto

En el contexto actual de la industria del videojuego, la gran cantidad de títulos lanzados cada año ha provocado una creciente homogeneización de propuestas y mecánicas, dificultando la aparición de experiencias realmente innovadoras. Muchos géneros tradicionales han alcanzado un punto de saturación creativa, donde las fórmulas establecidas se repiten constantemente y reducen la capacidad de sorprender al jugador.

Ante esta situación, este proyecto surge como una propuesta que busca aportar una visión diferente mediante la combinación de dos géneros con gran potencial complementario: el Metroidvania y la gestión de recursos. Mientras que el género Metroidvania destaca por su exploración progresiva, el descubrimiento de secretos y la obtención de habilidades que transforman la movilidad y el acceso al mundo, la gestión de recursos añade una dimensión estratégica basada en la planificación, el crecimiento y la toma de decisiones.

La integración de ambos sistemas permite construir una experiencia más profunda y dinámica, donde la exploración no solo sirve para avanzar en la aventura, sino también para obtener recursos y desarrollar el entorno del jugador. De esta forma, cada acción tiene un impacto tangible en la progresión del mundo y en las posibilidades jugables, creando un ciclo constante de exploración, mejora y expansión.

Con esta propuesta, el proyecto pretende ofrecer una experiencia que combine acción, estrategia y narrativa de manera cohesionada, aportando frescura dentro de un mercado cada vez más competitivo y estandarizado.

1.2 Justificación

La fusión entre los géneros metroidvania y gestión de recursos representa un desafío creativo que busca explorar la convergencia de dos sistemas de juego fundamentalmente distintos en una única experiencia.

1.3 Objetivos

1.3.1 Objetivo general

Nuestros objetivos generales son los siguientes:

- Implementar correctamente las mecánicas principales del videojuego en Godot utilizando C#.
- Utilizar de forma eficiente herramientas como Visual Studio Code, JetBrains Rider y GitHub durante el desarrollo del proyecto.
- Mejorar la coordinación y organización del trabajo en equipo mediante tareas y revisiones periódicas.
- Desarrollar una versión funcional del videojuego con exploración, combate, gestión de recursos y progresión.
- Realizar pruebas continuas para detectar errores y equilibrar las mecánicas del juego.

1.3.2 Objetivos específicos

Nuestros objetivos específicos son los siguientes:

- Diseñar y desarrollar un prototipo funcional del videojuego aplicando buenas prácticas de programación.
- Crear escenarios y personajes 2D con efecto parallax para el entorno del juego.
- Diseñar niveles interconectados siguiendo la estructura de un Metroidvania.
- Implementar mecánicas de exploración, combate y gestión de recursos.
- Aplicar técnicas de narrativa y diseño de niveles para mejorar la experiencia del jugador.
- Optimizar el rendimiento del videojuego dentro de Godot.
- Integrar correctamente gráficos, sonidos y animaciones en el proyecto.
- Organizar tareas y roles del equipo para cumplir los plazos de desarrollo.

1.4 Estrategia y planificación del proyecto

En este apartado se presenta la visión general del proyecto, respondiendo a la pregunta: **¿Cómo se pretende alcanzar los objetivos del proyecto?**

Las estrategias más comunes utilizadas en la gestión de proyectos son las siguientes:

- **Estrategia tradicional**: El proyecto se ejecuta paso a paso, siguiendo un plan estructurado y rígido, en el que los objetivos se alcanzan de forma secuencial.

- **Estrategia adaptativa**: El proyecto se divide en pequeñas metas incrementales, lo que permite avanzar de manera flexible y adaptarse a cambios o retroalimentación constante.

- **Estrategia por escenarios**: Se plantean diferentes escenarios posibles para el desarrollo del proyecto, evaluando y seleccionando las alternativas más viables (*citado textualmente*).

- **Estrategia jerárquica**: Las tareas se distribuyen entre los miembros del equipo, delegando responsabilidades de forma clara y organizada.

- **Estrategia centralizada**: Una única persona asume el rol de coordinador principal, supervisando el avance del proyecto y tomando las decisiones clave.

- **Estrategia colaborativa**: Se basa en una comunicación activa y en la toma de decisiones conjunta entre todos los miembros del equipo, siendo especialmente útil en fases creativas como el diseño y las pruebas del videojuego.

- **Estrategia iterativa**: Se desarrollan prototipos funcionales en ciclos de trabajo cortos (sprints), permitiendo evaluar, ajustar y mejorar el producto de forma continua, de acuerdo con metodologías ágiles como Scrum.

Teniendo en cuenta la naturaleza del desarrollo de videojuegos —una actividad que combina creatividad, trabajo multidisciplinario e iteración constante—, se ha optado por adoptar principalmente la **estrategia adaptativa**. Este enfoque permitirá avanzar de forma progresiva, flexible y receptiva a los cambios, facilitando la mejora continua del proyecto a lo largo de su desarrollo.

1.5 Metodología de trabajo

En este proyecto de desarrollo de videojuegos se ha adoptado una metodología híbrida conocida como Scrumfall o Water-Scrum-Fall. Este enfoque combina la estructura del modelo tradicional en cascada, utilizada en las fases de planificación y documentación, con metodologías ágiles basadas en Scrum, permitiendo un desarrollo más flexible y adaptado a las necesidades del proyecto.

Durante la fase inicial se realiza una planificación detallada del videojuego, definiendo los objetivos, mecánicas, alcance y requisitos principales antes de comenzar el desarrollo. Una vez establecida esta base, el trabajo se organiza mediante sprints iterativos, donde se implementan y prueban las diferentes funcionalidades del juego de forma progresiva.

Para la organización y seguimiento de tareas se utiliza HacknPlan, una herramienta especializada en desarrollo de videojuegos que permite gestionar objetivos, asignar tareas, planificar mecánicas y realizar un seguimiento del progreso del proyecto. Además, el control de versiones y la colaboración entre los miembros del equipo se realizan mediante GitHub.

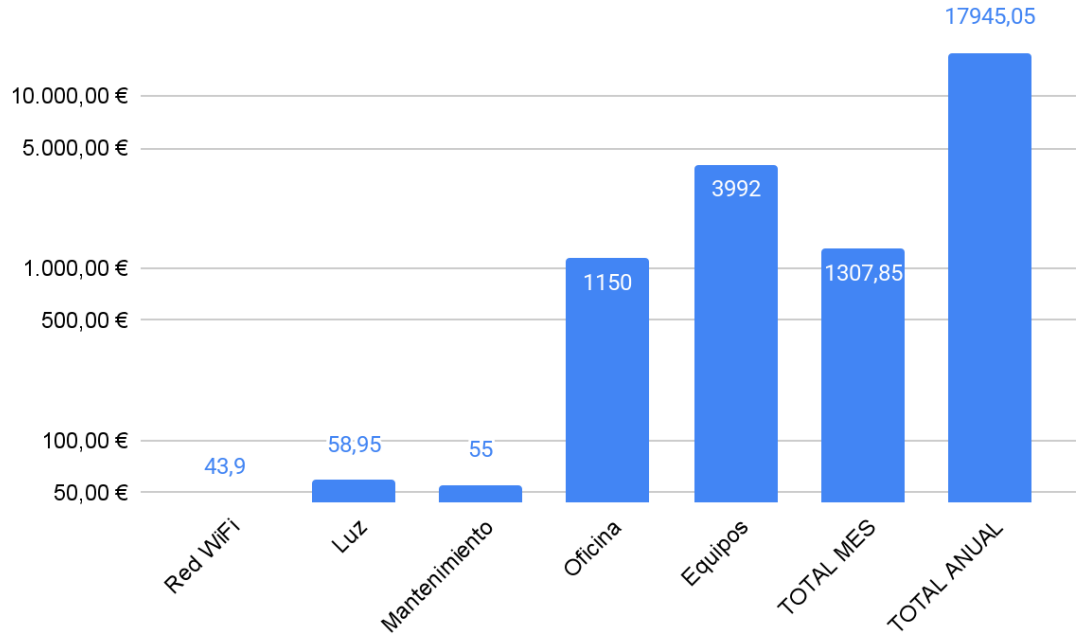
Finalmente, una vez completadas las fases principales de desarrollo, el proyecto entra en una etapa de pruebas, optimización y mantenimiento orientada a garantizar la estabilidad y calidad de la experiencia final.

Este enfoque híbrido permite combinar una planificación organizada con la flexibilidad necesaria para adaptarse a cambios y mejoras durante el desarrollo, algo especialmente importante en un proyecto creativo y dinámico como un videojuego.

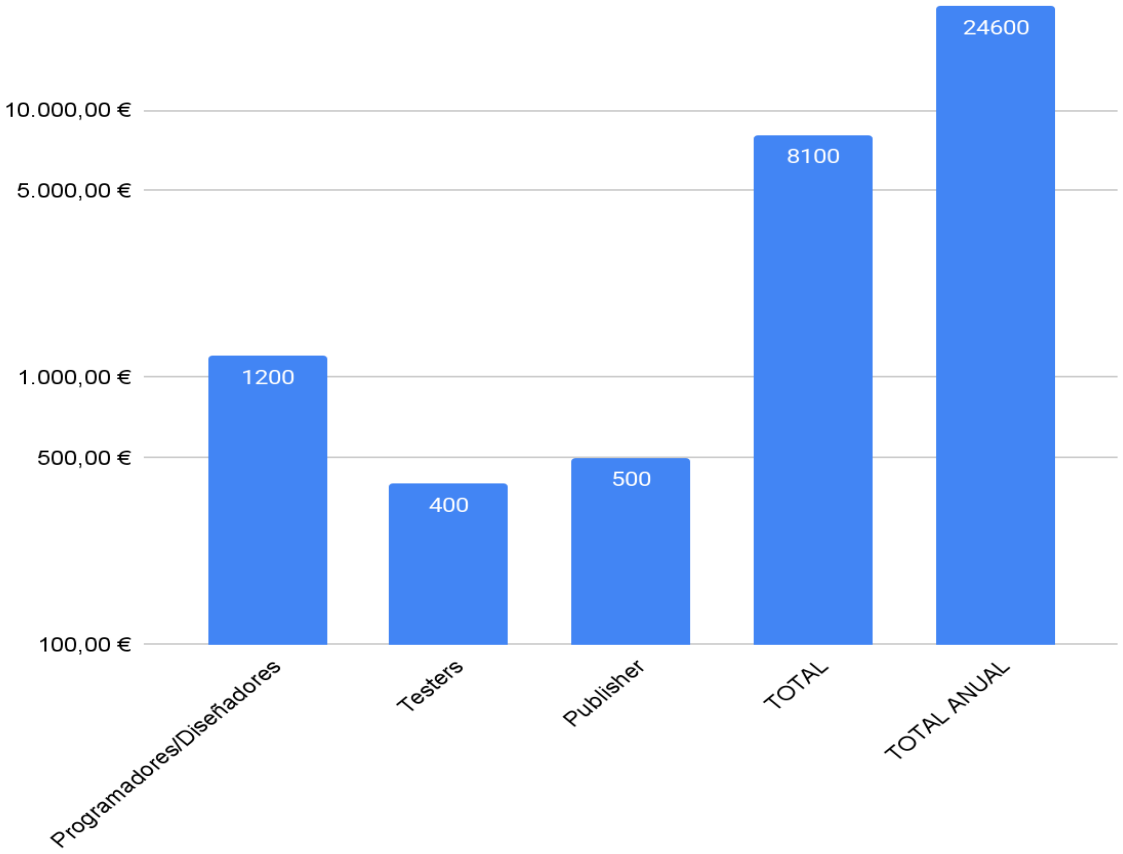
Representación gráfica:

1.6 Estudio económico y presupuestario

En este apartado se explica cómo han gestionado la economía y el presupuesto total del proyecto.



El equipo ha realizado un estudio de los costos de infraestructura para este proyecto, calculando el costo total mensual; también hay una columna de equipos que se debería pagar el primer mes para tener equipos con los que trabajar.



Para finalizar con los gráficos, aquí se puede ver los gastos del equipo en contratar personal; los programadores también son los diseñadores, así que el sueldo por programador también sería el de diseñador. Estos reciben 14 pagas, contadas las dos extraordinarias; los testers solo serán contratados los dos últimos meses del proyecto y, para finalizar, el equipo nos comenta que tienen un publisher especializado en el mercado que estará contratado todo el año, haciendo que también tenga 14 pagas, dos de ellas extraordinarias. Por último, se puede observar una columna que pone "total", pero no es del todo así, ya que es el total de un mes donde todos los trabajadores están cobrando. La última columna de todas es el gasto total de los trabajadores, contando que el proyecto solo va a durar un año.

2 Descripción del proyecto

2.1 Análisis de los requisitos

Para el desarrollo del proyecto ha sido necesario utilizar diferentes conocimientos técnicos y herramientas relacionadas con la programación y el diseño de videojuegos. Entre ellas destacan los fundamentos de programación orientada a objetos, el uso del motor gráfico Unity y conocimientos básicos sobre físicas, matemáticas aplicadas y diseño de videojuegos.

Además, el proyecto requiere una correcta planificación de mecánicas, sistemas y estructura de juego para garantizar una experiencia jugable estable y coherente.

2.1.1 Requisitos funcionales

Los requisitos funcionales definen las características y sistemas que el videojuego debe implementar:

- Sistema de movimiento y control del personaje.
- Sistema de habilidades desbloqueables.
- Sistema de economía y gestión de recursos.
- Diseño de niveles con plataformas, obstáculos y zonas explorables.
- Sistema de enemigos e interacción con el entorno.
- Sistema de progresión del jugador.
- Guardado y carga de partida.

2.1.2 Requisitos no funcionales

Los requisitos no funcionales definen las condiciones de calidad y funcionamiento del videojuego:

- Controles intuitivos y accesibles para el jugador.
- Rendimiento estable en ordenadores de sobremesa.
- Interfaz clara y fácil de comprender.
- Diseño visual coherente con la estética del proyecto.
- Experiencia de juego fluida y sin errores críticos.
- Curva de dificultad equilibrada.
- Compatibilidad con teclado y mando.

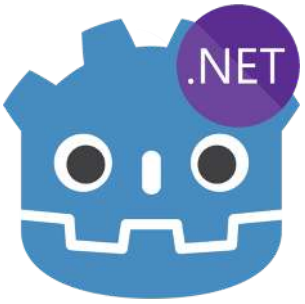
2.2 Tecnologías

2.2.1 Comparativa de las tecnologías valoradas

Godot

Ventajas:

- **Gratuito y de código abierto:** Totalmente libre, sin royalties ni licencias. Puedes modificar el motor si es necesario.
- **Motor ligero y de carga rápida:** Ideal para equipos pequeños o medianos.
- **GDScript (similar a Python):** Lenguaje muy accesible para desarrolladores principiantes.
- **Sistema de nodos muy modular:** Fácil de combinar 2D y 3D dentro de una misma escena.
- **Soporte nativo para proyectos 2.5D:** Muchos juegos indie lo han utilizado con éxito.
- **Exportación sencilla a múltiples plataformas:** Sin costes adicionales.
- **Comunidad en crecimiento y muchas herramientas creadas por la comunidad.**



Desventajas:

- **Rendimiento 3D inferior a Unity o Unreal:** Aunque ha mejorado mucho con Godot 4.x, no es tan óptimo para proyectos muy exigentes en 3D.

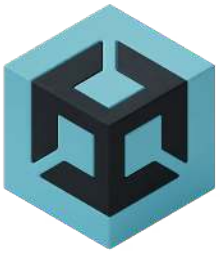
- **Menos recursos y tutoriales avanzados que Unity,** especialmente para proyectos híbridos como el tuyo.
- **GDScript puede ser una limitación** si estás acostumbrado a C# u otros lenguajes más potentes.
- **Menor compatibilidad con plugins y herramientas comerciales** (como shaders avanzados, servicios de análisis, monetización, etc.).

Unity

Ventajas:

- **Motor potente y versátil para 2D, 3D y 2.5D:** Muchos juegos comerciales 2.5D lo utilizan.
- **Rendimiento 3D superior:** Especialmente útil si el entorno 3D es complejo.
- **Gran comunidad y soporte:** Miles de tutoriales, recursos y documentación.
- **Asset Store muy completa:** Muchas herramientas preconstruidas que aceleran el desarrollo.
- **C# como lenguaje:** Potente y común en entornos profesionales.
- **Soporte oficial para proyectos comerciales y empresariales.**

Desventajas:



- **Cambios en la política de licencias** (como el caso de *Unity Runtime Fee* en 2023): han generado desconfianza en la comunidad.
- **Puede ser pesado y lento de cargar o compilar,** especialmente en equipos más modestos.
- **Exigencia de licencias de pago en ciertos casos,** si tu proyecto genera ingresos elevados.

2.2.2 Tecnologías escogidas

Después de analizar diferentes motores de desarrollo de videojuegos, finalmente se ha elegido **Godot** como la tecnología principal para estudiarla en profundidad y utilizarla durante la investigación y el desarrollo del proyecto. Esta decisión se ha tomado tras valorar diversas opciones, entre ellas **Unity**, que inicialmente se consideraba una alternativa viable.

Godot ha sido escogido por varios motivos:

- **Motor de código abierto y gratuito:**
  Godot es un motor completamente **open source**, lo que permite utilizarlo sin costes de licencia ni restricciones comerciales. Esto facilita el desarrollo del proyecto y resulta especialmente interesante en un contexto académico, donde el acceso libre a las herramientas y su funcionamiento interno puede ser una ventaja durante el proceso de investigación.

- **Ligereza y eficiencia del entorno de desarrollo:**
  Godot destaca por ser un motor ligero y rápido, tanto en la ejecución del editor como en los tiempos de compilación y prueba. Esto permite realizar iteraciones de forma más ágil durante el desarrollo del juego, facilitando la experimentación y la implementación progresiva de nuevas funcionalidades.

- **Buen soporte para desarrollo en 2D:**
  El proyecto se desarrolla en un entorno **bidimensional**, por lo que resulta fundamental disponer de un motor que ofrezca herramientas sólidas para este tipo de juegos. Godot cuenta con un sistema 2D muy optimizado que permite gestionar fácilmente elementos como sprites, animaciones, colisiones y cámaras.

- **Implementación de efectos visuales como parallax:**
  El diseño visual del juego incluye el uso de **efectos de paralaje (parallax)** para generar sensación de profundidad en los escenarios 2D. Godot ofrece herramientas específicas que facilitan la implementación de este tipo de efectos, permitiendo organizar distintas capas de fondo que se desplazan a diferentes velocidades respecto a la cámara.

- **Uso del lenguaje de programación C#:**
  Para el desarrollo de la lógica del juego se utilizará **C#**, un lenguaje ampliamente utilizado en el desarrollo de software y videojuegos. Godot ofrece compatibilidad con este lenguaje a través de su integración con .NET, lo que permite estructurar el código de forma clara, mantenible y escalable.

- **Arquitectura basada en escenas y nodos:**
  La estructura de Godot se basa en un sistema de **escenas y nodos**, que permite organizar los distintos elementos del juego de forma modular y reutilizable. Este enfoque facilita la gestión del proyecto y permite ampliar el juego de manera progresiva sin comprometer la organización del código.

Aunque **Unity** sigue siendo uno de los motores más utilizados en la industria del videojuego y ofrece un amplio ecosistema de herramientas y recursos, se ha considerado que **Godot se adapta mejor a las necesidades concretas de este proyecto**, especialmente por su ligereza, su enfoque en el desarrollo 2D y su facilidad para implementar efectos visuales como el parallax.

Por todos estos motivos, Godot se ha considerado la opción más adecuada tanto para la fase de investigación como para el desarrollo práctico del proyecto.

2.3 Estructura del proyecto

La estructura del proyecto se dividirá en 3 partes con el mismo nivel de importancia: Diseño, lógica y arte.

El diseño de los niveles está pensado para ofrecer una experiencia intuitiva y fácil de comprender para el jugador, permitiendo explorar y desplazarse de forma natural por el entorno. Al mismo tiempo, también se incorporan zonas más desafiantes y obstáculos que requieren habilidad, precisión y un buen dominio de las mecánicas del juego para poder superarlos. En esta parte se deciden cosas como la sensación que queremos dar al movimiento del jugador (libertad, uso de habilidades obtenidas), la forma en la que los enemigos se comportan acorde a las acciones del jugador, los obstáculos y trampas con los que se podrá encontrar el jugador.

La lógica del videojuego es, en pocas palabras, la sección del proyecto en la que se programa cómo se comportan todos los elementos de este acorde al diseño que se ha elegido (ej.: jugador, enemigos, objetos).

El arte es el proceso en que se crea el arte del juego, donde se buscará un equilibrio entre fluidez y que sea llamativo visualmente.

Se eligió este tipo de estructura por la especialidad del equipo.

2.4 Descripción de los componentes

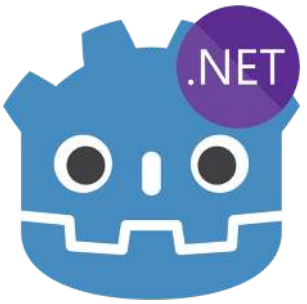
2.4.1 Godot

Es el núcleo de nuestro desarrollo, ya que es el motor de videojuegos que hemos optado por utilizar. Este es de código abierto, publicado bajo la Licencia MIT. Dicho motor funciona en una amplia gama de plataformas: Linux, Windows, macOS y BSD, además de tener soporte experimental para Android y HTML.

Godot tiene muchas gamas de características y herramientas que permiten a los desarrolladores crear y desarrollar videojuegos en 2D y en 3D, destacándose sobre todo en el campo de 2D.

Las herramientas que ofrece para crear un juego completo son:

- Motor de físicas 2D y 3D.
- Sistema de animación con curvas, interpolaciones y más.
- Editor visual intuitivo.
- Scripting con autocompletado.
- Depurador y gestor de rendimiento.



Godot utiliza como lenguaje principal GDScript, inspirado en Python, lo que lo hace muy accesible para principiantes. **También soporta C++, C#, y VisualScript** para usuarios más avanzados. Al ser un motor multiplataforma, permite ejecutar y probar los proyectos fácilmente tanto en PC como en dispositivos móviles o navegador web. Utiliza efectos de postprocesado como el mapeo tonal, niebla y ajuste de curvas de color para lograr resultados visuales profesionales incluso con recursos limitados.

2.4.2 C#

C# es un lenguaje de programación orientado a objetos desarrollado por Microsoft. Se utiliza principalmente para el desarrollo de aplicaciones, videojuegos y software multiplataforma dentro del entorno Microsoft .NET.

En el proyecto se utiliza C# para programar la lógica del juego o aplicación. Esto incluye el control de personajes, la gestión de eventos, la interacción con el usuario y el funcionamiento general del sistema.

C# permite estructurar el código mediante clases, métodos y objetos, lo que facilita la organización y el mantenimiento del proyecto.

2.4.3 LibreSprite

LibreSprite es un programa de código abierto utilizado para crear gráficos pixel art y sprites para videojuegos. Permite diseñar personajes, objetos y animaciones que posteriormente se utilizan dentro del juego.

En el proyecto, LibreSprite se emplea para crear los elementos visuales en formato de sprites, que son imágenes utilizadas por el motor del juego para representar personajes, enemigos, objetos o efectos.

El programa permite trabajar con capas, animaciones y paletas de colores, facilitando la creación de gráficos optimizados para juegos 2D.

2.4.4 Tiled

Tiled (map editor) es una herramienta de código abierto utilizada para crear mapas 2D basados en tiles. Permite diseñar niveles o escenarios para videojuegos organizando pequeñas imágenes llamadas tiles.

En el proyecto, Tiled se utiliza para diseñar los niveles del juego, colocando los diferentes elementos del escenario como suelos, paredes, obstáculos o decoraciones.

El editor permite trabajar con diferentes capas (layers), lo que facilita separar elementos visuales, colisiones y objetos interactivos.

2.4.5 GitHub

GitHub es una plataforma de desarrollo colaborativo que utiliza el sistema de control de versiones Git. Permite almacenar, gestionar y compartir el código fuente de un proyecto.

En el proyecto, GitHub se utiliza para guardar el código, controlar los cambios y colaborar en el desarrollo. Cada modificación realizada en el proyecto puede registrarse mediante commits, lo que permite mantener un historial de versiones y recuperar versiones anteriores si es necesario.

También facilita el trabajo en equipo mediante ramas (branches), donde cada desarrollador puede trabajar en nuevas funciones sin afectar directamente al código principal.

2.5 Definición de las funcionalidades

A continuación se detallan las principales funcionalidades que ofrece la solución “Faeterna”. Para cada una, se explica qué permite hacer, el proceso conceptual para lograrlo y el estado de la implementación actual.

2.5.1 Sistema de movimiento y control del personaje principal

Descripción:

Permite al jugador controlar al personaje principal en el entorno del juego, incluyendo desplazamiento, salto y doble salto, así como la interacción con la física del entorno.

Proceso conceptual:

- Captura de las entradas del jugador (teclado o mando).
- Procesamiento de estas entradas para mover y animar al personaje según las colisiones y el entorno.

Estado:

Implementado completamente.

2.5.2 Sistema de combate

Descripción:

Permite los enfrentamientos entre el jugador y enemigos mediante ataques, habilidades y defensa.

Proceso conceptual:

- Detección y gestión de colisiones (hitboxes) entre el personaje y los enemigos.
- Secuencias de ataque y habilidades según el progreso del jugador.
- Gestión de la vida tanto del jugador como de los enemigos.

Estado:

Implementado parcialmente, pendiente de mejoras y ampliaciones en la IA enemiga.

2.5.3 Sistema de inventario

Descripción:

Permite al jugador recoger, almacenar y gestionar distintos objetos encontrados en el juego (armas, consumibles, llaves, etc.).

Proceso conceptual:

- Registro de los objetos recogidos mediante colisiones en el escenario.
- Interfaz de usuario para visualizar y manipular el inventario.
- Uso o eliminación de objetos directamente desde el inventario.

Estado:

Implementado parcialmente, con las funciones básicas disponibles.

2.5.4 Gestión de niveles y escenas

Descripción:

Gestiona la transición entre diferentes escenarios o niveles del juego, así como los menús y pantallas especiales (Game Over, pausa, etc.).

Proceso conceptual:

- Carga y descarga de escenas según el estado del juego.
- Animaciones de transición y control de información persistente como la puntuación.

Estado:

Implementado completamente.

2.5.5 Sistema de diálogos y narrativa

Descripción:

Permite mostrar diálogos entre personajes y avanzar en la narrativa del juego.

Proceso conceptual:

- Motor de diálogos, de tipo secuencial o con opciones de respuesta.
- Representación visual de los diálogos, textos y retratos de los personajes.

Estado:

Implementado parcialmente; previsto ampliar en futuras versiones.

2.5.6 Sistema de audio y música

Descripción:

Gestiona la reproducción de efectos sonoros y música ambiental según el contexto del juego.

Proceso conceptual:

- Asignación de pistas musicales y sonidos a diferentes escenas o eventos.
- Control dinámico de efectos de sonido en función de las acciones del jugador.

Estado:

Implementado completamente.

2.5.7 Guardado y carga de partida

Descripción:

Permite guardar el progreso de la partida para poder reanudarla posteriormente.

Proceso conceptual:

- Serialización del estado del juego (nivel, inventario, puntuación, etc.).
- Recuperación de los datos almacenados al iniciar o cargar una partida.

Estado:

Funcionalidad reservada para una futura ampliación.

3 Diario de desarrollo

El diario de desarrollo recoge el proceso seguido durante la creación del videojuego, mostrando la evolución de las ideas iniciales hasta la implementación final. Este apartado permite documentar las decisiones tomadas durante el proyecto, los cambios realizados y los problemas encontrados a lo largo del desarrollo.

Dentro de la industria del videojuego, la documentación tiene un papel fundamental, especialmente en proyectos desarrollados por equipos multidisciplinares. Uno de los documentos más importantes es el Game Design Document (GDD), ya que actúa como una guía general del videojuego. En él se detallan aspectos como las mecánicas, la narrativa, el diseño de niveles, la dirección artística, la progresión del jugador y los sistemas principales del proyecto. Gracias a este documento, todos los miembros del equipo pueden trabajar siguiendo una misma visión y mantener la coherencia del desarrollo.

Además, el GDD facilita la organización interna, la planificación de tareas y la comunicación entre departamentos, reduciendo errores y evitando malentendidos durante la producción. También resulta útil para presentar el proyecto a colaboradores externos o posibles inversores, ya que resume de forma estructurada las características principales del videojuego.

Toda esta documentación complementaria, incluyendo el GDD, archivos de gestión interna, concept arts y sprites finales utilizados en el videojuego, se encuentra adjunta en el apartado de anexos de esta memoria.

3.1 Conclusiones

En este apartado se realiza una valoración general del proyecto desarrollado, analizando tanto los resultados obtenidos como el proceso seguido durante la creación del videojuego. También se revisa el grado de cumplimiento de los objetivos planteados inicialmente, la metodología utilizada durante el desarrollo y las posibles mejoras o ampliaciones planteadas de cara al futuro del proyecto.

3.1.1 Conclusiones generales del proyecto

Este proyecto nos ha permitido entender mucho mejor cómo funciona realmente el desarrollo de un videojuego dentro de un entorno de producción. A lo largo del proceso hemos aprendido tanto a nivel técnico como organizativo, descubriendo la importancia de la planificación, la comunicación y la capacidad de adaptación ante los problemas que van surgiendo durante el desarrollo.

También nos ha servido para conocer mejor nuestros propios límites y capacidades como equipo. Aunque no hemos podido alcanzar todo lo que planteábamos al inicio del proyecto, principalmente porque la idea original era demasiado ambiciosa para el tiempo y recursos disponibles, consideramos que el resultado final ha sido muy positivo. Gran parte de las mecánicas principales y de la identidad del videojuego han conseguido implementarse correctamente, permitiendo mostrar una demo sólida y representativa de la visión inicial del proyecto.

3.1.2 Consecución de los objetivos

La mayoría de los objetivos planteados al inicio del proyecto se han completado satisfactoriamente. Se ha conseguido desarrollar una base jugable funcional, implementar las mecánicas principales del metroidvania y establecer una dirección artística coherente con la temática del videojuego.

Además, también se han alcanzado objetivos relacionados con la organización del proyecto, la documentación y el trabajo en equipo. Aunque algunas características quedaron fuera de esta primera versión por falta de tiempo o complejidad técnica, los resultados obtenidos han sido superiores a los esperados en varios apartados del desarrollo.

3.1.3 Valoración de la metodología y planificación

La metodología utilizada durante el proyecto ha sido clave para mantener una organización constante y un seguimiento del progreso realizado. El uso de herramientas como HacknPlan ha facilitado la distribución de tareas, la planificación de objetivos y el control del trabajo pendiente en cada sprint.

Por otra parte, las reuniones diarias y semanales han ayudado a mantener una buena comunicación entre los miembros del equipo, permitiendo detectar problemas rápidamente y tomar decisiones de forma conjunta. Gracias a esta organización, el proyecto ha podido avanzar de forma estable y llegar a una versión final mucho más pulida y completa.

3.1.4 Visión a futuro

La visión de futuro del proyecto incluye todas aquellas mecánicas y sistemas que no han podido implementarse en esta demo debido a las limitaciones de tiempo y alcance del proyecto. Entre ellas destacan los sistemas completos de gestión de recursos, automatización, economía y expansión de zonas explorables dentro del mundo del juego.

Además, también se plantea seguir mejorando aspectos relacionados con el equilibrio de gameplay, la variedad de enemigos, el diseño de niveles y la progresión del jugador. Todo esto permitiría convertir la demo actual en una experiencia mucho más completa y cercana a la idea original planteada al inicio del desarrollo.

4. Glosario

2D: Tipo de entorno gráfico bidimensional utilizado en videojuegos, basado en ancho y alto sin profundidad real.

2.5D: Estilo visual que combina elementos 2D con efectos o entornos 3D para generar sensación de profundidad.

API (Application Programming Interface): Conjunto de herramientas y funciones que permiten la comunicación entre distintos programas o sistemas.

Asset Store: Tienda digital integrada en motores como Unity donde se pueden adquirir recursos y herramientas para el desarrollo.

Branch: Rama de desarrollo dentro de Git que permite trabajar en nuevas funcionalidades sin afectar directamente al código principal.

BSD (Berkeley Software Distribution): Familia de sistemas operativos derivados de Unix compatibles con Godot.

C#: Lenguaje de programación orientado a objetos desarrollado por Microsoft y utilizado en el proyecto para programar la lógica del videojuego.

Commit: Registro de un conjunto de cambios realizados dentro de un repositorio Git.

Framework .NET: Plataforma de desarrollo de Microsoft que permite ejecutar aplicaciones creadas con lenguajes como C#.

FPS (Frames Per Second): Medida de rendimiento gráfico que indica el número de imágenes mostradas por segundo.

GDD (Game Design Document): Documento de diseño del videojuego donde se definen mecánicas, historia, personajes y funcionalidades.

GDScript: Lenguaje de programación propio de Godot, inspirado en Python.

Git: Sistema de control de versiones distribuido utilizado para gestionar los cambios del proyecto.

HacknPlan: Herramienta de gestión de proyectos especializada en desarrollo de videojuegos.

Hitbox: Área de colisión de un personaje u objeto utilizada para detectar impactos o interacciones.

HTML (HyperText Markup Language): Lenguaje estándar utilizado para crear páginas web.

IA (Inteligencia Artificial): Sistema que permite simular comportamientos inteligentes en enemigos o personajes del videojuego.

Indie: Término utilizado para referirse a videojuegos desarrollados por equipos independientes o pequeños estudios.

Interpolación: Proceso matemático utilizado en animaciones para generar movimientos fluidos entre distintos puntos.

MIT License: Licencia de software libre que permite utilizar, modificar y distribuir el código sin grandes restricciones.

Metroidvania: Subgénero de videojuegos centrado en la exploración, la progresión mediante habilidades y los mapas interconectados.

Nodo: Elemento básico dentro de la arquitectura de Godot que representa funcionalidades u objetos dentro de una escena.

Open Source: Software de código abierto que permite acceder y modificar su código fuente.

Parallax: Efecto visual donde diferentes capas del fondo se desplazan a velocidades distintas para crear profundidad.

Pixel art: Estilo artístico digital basado en píxeles, habitual en videojuegos 2D.

Publisher: Empresa encargada de publicar, promocionar o distribuir un videojuego.

Shader: Programa gráfico utilizado para generar efectos visuales avanzados.

Sprint: Período corto de desarrollo dentro de metodologías ágiles en el que se trabaja en objetivos concretos.

Sprite: Imagen 2D utilizada para representar personajes, objetos o efectos dentro de un videojuego.

Tiled: Editor de mapas 2D basado en tiles utilizado para crear niveles del videojuego.

Tile: Pequeña pieza gráfica reutilizable utilizada para construir escenarios 2D.

UI (User Interface): Conjunto de elementos visuales con los que interactúa el jugador.

VisualScript: Sistema de programación visual compatible con Godot.

5. Bibliografía

1.4
https://www.aicad.es/estrategias-de-un-proyecto

https://es.linkedin.com/pulse/7-secretos-estrat%C3%A9gicos-para-el-%C3%A9xito-de-tu-proyecto-garc%C3%ADa-burgue%C3%B1o

https://raulrodriguezchaparro.es/7-cosas-que-puedes-hacer-para-mejorar-la-ejecucion-de-tu-proyecto/

2.4
https://es.wikipedia.org/wiki/Godot

https://www.luisllamas.es/dotnet-framework-core-y-standard/

https://github.com/Kiamo2/YATI

6 Anexos

https://github.com/KotasanStudio/Faeterna

- GDD
- Financiamientos e Inversores
- Plan de Pruebas_Faeterna
- El mercado de los videojuegos
- Estudio de viabilidad económica del projecto
- Proceso creativo
- Desarrollo Historia de Faeterna
- Diseño del gameplay
- Tabla de estadísticas
- Concept arts
- Sprites