Un dibujo de un perro

Descripción generada automáticamente con confianza media

**UNIVERSIDAD DE CASTILLA-LA MANCHA**

**ESCUELA SUPERIOR DE INFORMÁTICA**

GRADO EN INGENIERÍA INFORMÁTICA

Videojuego de ayuda al diagnóstico temprano de Trastorno del Déficit de Atención e Hiperactividad en niños y adolescentes

Sandra Ciudad Moreno

Julio, 2024

# **RESUMEN**

*Prometheus-9* es un videojuego narrativo de ciencia ficción desarrollado en Unity que combina exploración, puzles y decisiones morales para contar una historia inmersiva en un entorno espacial. El jugador despierta solo en una nave averiada, sin rastro de la tripulación, y debe restaurar sistemas, descubrir grabaciones del pasado y enfrentarse a una decisión final que determinará el desenlace de la misión. A lo largo de 10 misiones, se desbloquean fragmentos narrativos en forma de cintas de audio/vídeo que revelan una historia de sabotaje, contacto alienígena y dilemas éticos sobre el conocimiento y la supervivencia.

El proyecto fue concebido como una experiencia jugable completa con enfoque en la integración entre narrativa y mecánicas. Cada misión representa tanto un reto jugable como una pieza de la historia, construyendo un flujo progresivo donde jugabilidad y narrativa se retroalimentan. El jugador explora distintas zonas de la nave, resuelve desafíos técnicos, recolecta objetos y escucha grabaciones que humanizan a los personajes ausentes, culminando en una toma de decisión con tres posibles finales.

Desde el punto de vista técnico, se aplicaron buenas prácticas de ingeniería de software, incluyendo una arquitectura modular en C#, patrones de diseño como Singleton, controladores independientes por misión y sistemas de inventario, UI y cámaras bien estructurados. El desarrollo siguió metodologías ágiles adaptadas a un entorno unipersonal, lo que permitió iteraciones frecuentes, pruebas constantes y corrección de errores durante todo el proceso.

El juego está acompañado por un sistema de gamificación sutil: barras de energía, recompensas narrativas, penalizaciones lógicas y una interfaz clara que guían al jugador sin romper la inmersión. La ambientación sonora y visual refuerza la sensación de soledad y misterio, y el ritmo está cuidadosamente medido para mantener la tensión narrativa.

En conclusión, *Prometheus-9* es un proyecto que demuestra cómo, con recursos limitados pero planificación y visión clara, se puede construir una experiencia interactiva rica en contenido narrativo, emocionalmente impactante y técnicamente sólida. El resultado final no solo cumple sus objetivos, sino que deja abierta la puerta a futuras ampliaciones y evoluciones dentro del mismo universo de ficción.

# **ABSTRACT**

*Prometheus-9* is a narrative science fiction video game developed in Unity, blending exploration, puzzles, and moral decision-making to tell an immersive story set in a deep-space environment. The player wakes up alone aboard a damaged spaceship, with no sign of the crew, and must restore systems, uncover past events through security recordings, and ultimately face a final decision that will determine the fate of the mission. Across 10 missions, players unlock narrative fragments through video/audio logs that reveal a story of sabotage, alien contact, and ethical dilemmas regarding knowledge and survival.

The project was conceived as a complete playable experience, with a strong focus on the integration between gameplay mechanics and storytelling. Each mission offers both a mechanical challenge and a narrative piece, building a progressive flow where gameplay and narrative enhance each other. The player explores different areas of the ship, solves technical problems, collects items, and listens to recordings that bring depth to the absent characters, culminating in a meaningful final choice with three different endings.

From a technical perspective, the game applies solid software engineering practices, including a modular architecture in C#, design patterns like Singleton, independent mission controllers, and well-structured systems for inventory, UI, and camera control. The development followed agile methodologies adapted to a solo developer workflow, enabling constant iteration, testing, and debugging throughout the process.

Gamification elements are subtly implemented: energy bars, narrative rewards, logical penalties, and a clean user interface guide the player without breaking immersion. The audiovisual atmosphere reinforces a sense of isolation and mystery, with carefully measured pacing to maintain narrative tension.

In conclusion, *Prometheus-9* demonstrates that with limited resources but strong planning and a clear creative vision, it is possible to build an interactive experience that is narratively rich, emotionally impactful, and technically robust. The final result not only fulfills its objectives but also opens the door to future expansions and deeper explorations within the same fictional universe.

**ÍNDICE**

[RESUMEN 3](#_Toc194533361)

[ABSTRACT 5](#_Toc194533362)

[CAPÍTULO 1. INTRODUCCIÓN 8](#_Toc194533363)

[**1.1. EL AUGE DE LOS VIDEOJUEGOS Y SU CONTEXTO CULTURAL Y TECNOLÓGICO** 8](#_Toc194533364)

[**1.2. CONTEXTO DEL PROYECTO** 8](#_Toc194533365)

[**1.3. ALCANCE DEL PROYECTO** 8](#_Toc194533366)

[**1.4. MOTIVACIÓN Y SOLUCIÓN PROPUESTA** 8](#_Toc194533367)

[**1.5. ESTRUCTURA DE LA MEMORIA** 8](#_Toc194533368)

[CAPÍTULO 2. OBJETIVOS DEL PROYECTO Y HERRAMIENTAS UTILIZADAS 9](#_Toc194533369)

[**2.1. OBJETIVOS TÉCNICOS, NARRATIVOS Y METODOLÓGICOS DEL DESARROLLO** 9](#_Toc194533370)

[**2.2. HERRAMIENTAS UTILIZADAS** 9](#_Toc194533371)

[CAPÍTULO 3. FUNDAMENTOS TEÓRICOS Y ESTADO DEL ARTE 10](#_Toc194533372)

[**3.1. ¿QUÉ SON LOS JUEGOS NARRATIVOS?** 10](#_Toc194533373)

[**3.1.1. Definición y evolución histórica** 10](#_Toc194533374)

[**3.2. GAMIFICACIÓN** 10](#_Toc194533375)

[CAPÍTULO 4. METODOLOGÍA DE TRABAJO 11](#_Toc194533376)

[**4.1. METODOLOGÍAS ÁGILES APLICADAS** 11](#_Toc194533377)

[**4.1.1. Contexto y necesidad de un enfoque ágil** 11](#_Toc194533378)

[**4.2. ARQUITECTURA DEL SISTEMA** 11](#_Toc194533379)

[**4.2.1. Filosofía de diseño** 11](#_Toc194533380)

[**4.3. EVOLUCIÓN DEL PROYECTO** 11](#_Toc194533381)

[CAPÍTULO 5. RESULTADOS 12](#_Toc194533382)

[**5.1. DISEÑO DEL JUEGO** 12](#_Toc194533383)

[**5.1.1. Historia, universo y tono** 12](#_Toc194533384)

[**5.1.3. Estructura de juego progresiva** 12](#_Toc194533385)

[**5.1.5. Grabaciones como núcleo narrativo** 12](#_Toc194533386)

[**5.2. ARQUITECTURA FINAL DEL JUEGO** 12](#_Toc194533387)

[**5.2.1. Escenas y estructuras del proyecto en Unity** 12](#_Toc194533388)

[**5.2.3. Optimizaciones técnicas** 12](#_Toc194533389)

[**5.2.4. Pruebas realizadas y balance** 12](#_Toc194533390)

[CAPÍTULO 6. CONCLUSIONES Y PROPUESTAS DE TRABAJO FUTURO 13](#_Toc194533391)

[**6.1. CONCLUSIONES GENERALES** 13](#_Toc194533392)

[**6.1.1. El valor de unir narrativa y jugabilidad** 13](#_Toc194533393)

[**6.1.3. La narrativa como motor emocional** 13](#_Toc194533394)

[**6.2. OBJETIVOS ALCANZADOS** 13](#_Toc194533395)

[**6.2.1. Revisión exhaustiva de los objetivos propuestos** 13](#_Toc194533396)

[**6.3. COMPETENCIAS ADQUIRIDAS** 13](#_Toc194533397)

[**6.3.1. Competencias técnicas avanzadas** 13](#_Toc194533398)

[**6.3.2. Competencias creativas y de diseño narrativo** 13](#_Toc194533399)

[**6.3.3. Competencias organizativas y de gestión de proyecto** 13](#_Toc194533400)

[**6.4. LIMITACIONES TÉCNICAS Y NARRATIVAS** 13](#_Toc194533401)

[**6.4.1. Alcance narrativo limitado** 13](#_Toc194533402)

[**6.4.2. Recursos visuales genéricos** 14](#_Toc194533403)

[**6.4.3. Linealidad estructural** 14](#_Toc194533404)

[**6.4.4. Inteligencia Artificial limitada** 14](#_Toc194533405)

[**6.4.5. Duración corta** 14](#_Toc194533406)

[**6.5. TRABAJO FUTURO** 14](#_Toc194533407)

[**6.5.1. Ampliación narativa** 14](#_Toc194533408)

[**6.5.2. Mejora de la estructura jugable** 14](#_Toc194533409)

[**6.5.3. Evolución audiovisual** 14](#_Toc194533410)

[**6.5.4. Enriquecimiento del sistema de Inteligencia Artificial** 14](#_Toc194533411)

[**6.5.5. Ampliacion del universo del juego** 14](#_Toc194533412)

[**6.5.6. Portabilidad y accesibilidad** 14](#_Toc194533413)

[**6.5.7. Conclusión final** 14](#_Toc194533414)

# **CAPÍTULO 1. INTRODUCCIÓN**

El propósito de este capítulo es detallar el contexto en el que se va a desarrollar este Trabajo de Fin de Grado, a fin de comprender el problema que hay que abordar y la solución propuesta.

## **1.1. EL AUGE DE LOS VIDEOJUEGOS Y SU CONTEXTO CULTURAL Y TECNOLÓGICO**

El

De cara al futuro, se espera que estos adquieran incluso un mayor alcance, debido a la influencia en su desarrollo de nuevas tendencias, como la inteligencia artificial y los servicios en la nube.

## **1.2. CONTEXTO DEL PROYECTO**

Este proyecto está sumamente relacionado con las áreas de investigación y desarrollo de los juegos serios y la evaluación psicológica.

## **1.3. ALCANCE DEL PROYECTO**

## **1.4. MOTIVACIÓN Y SOLUCIÓN PROPUESTA**

## **1.5. ESTRUCTURA DE LA MEMORIA**

# **CAPÍTULO 2. OBJETIVOS DEL PROYECTO Y HERRAMIENTAS UTILIZADAS**

## **2.1. OBJETIVOS TÉCNICOS, NARRATIVOS Y METODOLÓGICOS DEL DESARROLLO**

## **2.2. HERRAMIENTAS UTILIZADAS**

# **CAPÍTULO 3. FUNDAMENTOS TEÓRICOS Y ESTADO DEL ARTE**

## **3.1. ¿QUÉ SON LOS JUEGOS NARRATIVOS?**

### **3.1.1. Definición y evolución histórica**

**3.1.2. Narrativa emergente vs. narrativa estructurada**

**3.1.3. Diseño narrativo en videojuegos**

**3.1.4. Elementos clave de un juego narrativo**

## **3.2. GAMIFICACIÓN**

**3.2.1. Origen y concepto de gamificación**

**3.2.2. Principales mecánicas de gamificación**

**3.2.3. Tipos de motivación en gamificación**

**3.2.4. Gamificación como diseño consciente en Prometheus-9**

# **CAPÍTULO 4. METODOLOGÍA DE TRABAJO**

## **4.1. METODOLOGÍAS ÁGILES APLICADAS**

### **4.1.1. Contexto y necesidad de un enfoque ágil**

**4.1.2. Aplicación específica de Scrum en el proyecto**

## **4.2. ARQUITECTURA DEL SISTEMA**

### **4.2.1. Filosofía de diseño**

**4.2.2. Componentes principales del sistema**

**4.2.3. Comunicación entre componentes**

## **4.3. EVOLUCIÓN DEL PROYECTO**

# **CAPÍTULO 5. RESULTADOS**

## **5.1. DISEÑO DEL JUEGO**

### **5.1.1. Historia, universo y tono**

**5.1.2. Jugabilidad centrada en la narrativa**

### **5.1.3. Estructura de juego progresiva**

**5.1.4. Misiones con identidad propia**

### **5.1.5. Grabaciones como núcleo narrativo**

**5.1.6. Decisión final. Ramificación narrativa.**

## **5.2. ARQUITECTURA FINAL DEL JUEGO**

### **5.2.1. Escenas y estructuras del proyecto en Unity**

**5.2.2. Módulos y scripts clave**

### **5.2.3. Optimizaciones técnicas**

### **5.2.4. Pruebas realizadas y balance**

# **CAPÍTULO 6. CONCLUSIONES Y PROPUESTAS DE TRABAJO FUTURO**

## **6.1. CONCLUSIONES GENERALES**

### **6.1.1. El valor de unir narrativa y jugabilidad**

**6.1.2. La fuerza del diseño modular y escalable**

### **6.1.3. La narrativa como motor emocional**

## **6.2. OBJETIVOS ALCANZADOS**

### **6.2.1. Revisión exhaustiva de los objetivos propuestos**

## **6.3. COMPETENCIAS ADQUIRIDAS**

### **6.3.1. Competencias técnicas avanzadas**

### **6.3.2. Competencias creativas y de diseño narrativo**

### **6.3.3. Competencias organizativas y de gestión de proyecto**

## **6.4. LIMITACIONES TÉCNICAS Y NARRATIVAS**

### **6.4.1. Alcance narrativo limitado**

### **6.4.2. Recursos visuales genéricos**

### **6.4.3. Linealidad estructural**

### **6.4.4. Inteligencia Artificial limitada**

### **6.4.5. Duración corta**

## **6.5. TRABAJO FUTURO**

### **6.5.1. Ampliación narativa**

### **6.5.2. Mejora de la estructura jugable**

### **6.5.3. Evolución audiovisual**

### **6.5.4. Enriquecimiento del sistema de Inteligencia Artificial**

### **6.5.5. Ampliacion del universo del juego**

### **6.5.6. Portabilidad y accesibilidad**

### **6.5.7. Conclusión final**

**Prometheus-9: Documentación del Proyecto de Videojuego Narrativo**

**Capítulo 1. Introducción**

**1.1. El auge de los videojuegos y su contexto cultural y tecnológico**

En las últimas décadas, el sector de los videojuegos ha pasado de ser una forma de entretenimiento de nicho a convertirse en una de las industrias culturales y tecnológicas más influyentes del mundo. Su crecimiento ha sido impulsado por avances tecnológicos, la expansión global del acceso a internet y cambios en las dinámicas sociales y culturales que han normalizado el juego como una actividad legítima para todas las edades.

Actualmente, el mercado de los videojuegos supera ampliamente a otras industrias del entretenimiento: según reportes recientes, genera más ingresos que el cine y la música combinados. Este fenómeno se debe no solo al número creciente de jugadores (se estima que superan los 3.500 millones a nivel global), sino también a la diversificación de plataformas (PC, consolas, móviles, realidad virtual) y de géneros, lo que ha permitido alcanzar públicos muy distintos.

Además, los videojuegos ya no son únicamente productos de consumo pasivo, sino medios de expresión creativa, educación y comunicación. Gracias a su interactividad, permiten experiencias personalizadas, inmersivas y con un alto grado de participación del usuario. Esta evolución ha propiciado el surgimiento de nuevos géneros, como los **juegos narrativos**, donde la historia y la toma de decisiones adquieren un papel central. Estos juegos representan una convergencia entre el videojuego tradicional y otras formas narrativas como el cine, la literatura o el teatro, dando lugar a una nueva manera de contar historias.

**1.2. Contexto del proyecto *Prometheus-9***

En este marco de transformación del videojuego como medio expresivo nace *Prometheus-9*, un proyecto de videojuego narrativo de ciencia ficción con elementos de misterio, exploración y puzles. Este trabajo surge dentro de un entorno académico, con el objetivo de aplicar tanto conocimientos técnicos (programación, arquitectura de software, diseño de interfaces) como creativos (guion, narrativa, ambientación sonora).

El proyecto se plantea como una propuesta independiente, sin los recursos de una gran producción, pero con la ambición de demostrar cómo la narrativa y la jugabilidad pueden integrarse eficazmente en un producto coherente y funcional. Para ello, se ha desarrollado un entorno jugable completo utilizando el motor Unity, y aplicando principios de diseño centrado en el jugador, gamificación y metodologías ágiles.

*Prometheus-9* no es solo un juego, sino también un experimento narrativo y un estudio de caso en desarrollo interactivo. Al estar centrado en un único jugador que explora una nave abandonada mientras reconstruye los eventos pasados a través de grabaciones, ofrece una experiencia íntima y envolvente que prioriza la inmersión por encima de la acción.

**1.3. Alcance del proyecto (narrativa, misiones, jugabilidad, técnica)**

El alcance del proyecto cubre todas las fases del desarrollo de un videojuego desde su concepción hasta su implementación y pruebas. Específicamente, se desarrolló:

* **Narrativa:** Una historia original con narrativa fragmentada en 10 grabaciones de audio/vídeo, tres finales distintos y un trasfondo de ciencia ficción cuidadosamente elaborado. Cada grabación revela parte de la trama y juntas conforman un rompecabezas narrativo que el jugador debe recomponer.
* **Misiones:** Un conjunto de 10 misiones interconectadas, cada una con mecánicas únicas (puzles lógicos, minijuegos, desafíos de exploración, secuencias de sigilo, etc.). Las misiones se articulan con la historia, de modo que cada desafío completado desbloquea nueva información o áreas de la nave.
* **Jugabilidad/Interfaz:** Un sistema de interfaz de usuario adaptativa, que incluye gestión de vida y energía del personaje, reproducción de grabaciones (cintas) encontradas, un inventario básico para objetos clave, e indicadores visuales de objetivos y estado. La interfaz está integrada diegéticamente (pantallas dentro del juego, HUD coherente con la ambientación) para no romper la inmersión.
* **Arquitectura técnica:** Una arquitectura de software modular en C# para Unity, con scripts especializados para misiones (MissionXController), un controlador general de estado (GameManager), controladores de cámara (CameraController), scripts de interacción (CheckPlayerPosition, CollectiblesController), entre otros. Esta arquitectura soporta la escalabilidad y el fácil mantenimiento del código.
* **Ambientación:** Una ambientación lograda mediante el uso de modelos 3D (en su mayoría gratuitos o de libre uso, adaptados al contexto), efectos de sonido envolventes, música de fondo atmosférica y un sistema de cámaras virtuales para crear momentos **cinemáticos**. Se diseñó la iluminación de la nave con tonos tenues y parpadeos eléctricos para reflejar su estado de avería, y se incorporaron detalles visuales (chispas, humo, cables sueltos) que cuentan parte de la historia de forma ambiental.

Cabe destacar que el proyecto **no** contempla funcionalidad multijugador, juego en red, ni opciones de guardado avanzado, dado que se trata de una experiencia pensada para una sola sesión relativamente corta (unas 2-3 horas de juego). Aun así, la calidad de la presentación, el diseño de las misiones y la cohesión narrativa permiten ofrecer una experiencia completa y significativa en sí misma.

**1.4. Motivación y solución propuesta**

La motivación principal del proyecto nace del interés por los videojuegos como medio narrativo, y por el potencial del género de ciencia ficción como vehículo para contar historias de exploración, contacto con lo desconocido y dilemas morales. El objetivo no era simplemente hacer un juego **divertido**, sino crear una experiencia narrativa interactiva que empujara al jugador a descubrir activamente una historia mientras interactúa con el entorno. Se buscaba generar en el jugador sensaciones de misterio, tensión y curiosidad intelectual, más allá del puro entretenimiento.

Para responder a esta motivación, se propuso una solución basada en varios pilares fundamentales:

* **Diseño de misiones coherente con la narrativa:** Cada reto u objetivo del jugador representa no solo una barrera lúdica, sino también una pieza del rompecabezas argumental. Por ejemplo, restablecer la electricidad de la nave no es solo un puzle técnico, sino la forma de revelar que alguien la había saboteado deliberadamente. Todas las mecánicas están justificadas por la historia.
* **Progresión narrativa mediante grabaciones desbloqueables:** La historia se descubre de forma gradual a medida que el jugador avanza. Tras completar cada misión, se desbloquea una nueva cinta de registro que revela otro fragmento de la trama. Este goteo de información asegura una inmersión gradual y mantiene la intriga, ya que el jugador siempre quiere saber qué ocurrió realmente.
* **Enfoque técnico sólido (modularidad y estabilidad):** Desde el inicio se priorizó una arquitectura modular y bien estructurada, que permitiera reutilización de código y facilidad de testeo. Esto garantiza la estabilidad del juego incluso en fases tempranas de prototipo, facilitando añadir contenido o modificar elementos sin introducir errores en cascada. Una base técnica robusta también reduce tiempos de depuración y permite centrarse en pulir la experiencia.
* **Toma de decisiones significativa al final del juego:** Se diseñó un momento culminante donde el jugador debe tomar una decisión que afecta el desenlace y refleja sus valores o interpretación de los eventos. Esta ramificación final da al jugador una sensación de agencia sobre la conclusión de la historia y refuerza la dimensión narrativa del proyecto.

De esta forma, *Prometheus-9* se convierte en una **historia jugada**, donde cada acción tiene un sentido dentro del relato y cada descubrimiento modifica la percepción del jugador sobre lo ocurrido. La solución propuesta equilibra narrativa y jugabilidad de tal modo que el jugador es a la vez protagonista y narrador: interactúa para avanzar y, al hacerlo, revela la trama que da contexto a sus acciones.

**1.5. Estructura de la memoria**

La documentación del proyecto se estructura en seis capítulos principales, que abarcan tanto los aspectos conceptuales como los técnicos del desarrollo:

* **Capítulo 1. Introducción:** Presenta el contexto general del proyecto, su motivación, los objetivos globales y explica brevemente cómo se organiza el documento (la memoria).
* **Capítulo 2. Objetivos del proyecto y herramientas utilizadas:** Enumera los objetivos específicos que se plantearon al iniciar el desarrollo, distinguiendo entre objetivos técnicos, narrativos y metodológicos. También detalla las herramientas tecnológicas y plataformas empleadas (motor de juego, lenguaje, IDE, control de versiones, herramientas gráficas y de audio, etc.).
* **Capítulo 3. Fundamentos teóricos y estado del arte:** Aborda las bases conceptuales relacionadas con el proyecto. Se profundiza en qué son los juegos narrativos (historia, clasificación, diseño narrativo, ejemplos clave en la industria) y se explica el concepto de gamificación, sus mecánicas y motivaciones, así como su aplicación concreta en *Prometheus-9*. También se mencionan obras de referencia que han inspirado o servido de comparación para el proyecto.
* **Capítulo 4. Metodología de trabajo:** Describe cómo se organizó y llevó a cabo el trabajo de desarrollo. Incluye las metodologías ágiles aplicadas (por ejemplo, la adaptación de Scrum a un desarrollo unipersonal), la arquitectura interna del sistema (cómo se estructuró el código, módulos, escenas, etc.) y la evolución del proyecto a través de sus distintas fases, desde la conceptualización inicial hasta el pulido final.
* **Capítulo 5. Resultados:** Detalla el producto final obtenido. Se describen tanto los aspectos de diseño del juego (narrativa, ambientación, jugabilidad, estructura de misiones, etc.) como los aspectos técnicos (arquitectura final, optimizaciones realizadas). También se resumen las pruebas de calidad efectuadas y se presenta el grado en que el juego logró integrar los elementos planteados. En este capítulo se incluye la descripción de cada misión y de cada grabación narrativa principal, mostrando cómo ambas facetas (reto jugable y trama) quedaron entrelazadas.
* **Capítulo 6. Conclusiones y propuestas de trabajo futuro:** Reflexiona sobre lo aprendido durante el desarrollo, evaluando el cumplimiento de los objetivos iniciales y extrayendo conclusiones generales. Además, se enumeran las limitaciones técnicas y narrativas identificadas en el producto final y se proponen mejoras o líneas de trabajo futuro para ampliar o pulir *Prometheus-9* más allá de lo conseguido en esta versión.

A continuación, se desarrolla cada uno de estos capítulos en detalle.

**Capítulo 2. Objetivos del proyecto y herramientas utilizadas**

**2.1. Objetivos técnicos, narrativos y metodológicos del desarrollo**

El desarrollo de *Prometheus-9* partió de una serie de objetivos específicos, tanto técnicos como narrativos, que guiaron todas las decisiones de diseño e implementación. Estos objetivos se pueden agrupar en cinco grandes bloques:

1. **Desarrollo de un videojuego narrativo funcional.** El objetivo principal fue concebir y construir un videojuego completo y funcional dentro del motor Unity. El juego debía ser jugable de principio a fin, sin errores críticos, e incluir una narrativa estructurada, un sistema de progresión claro y una secuencia lógica de retos. La estructura mínima esperada comprendía:
   * Un **tutorial implícito** al inicio para introducir las mecánicas básicas sin necesidad de explicaciones externas.
   * 10 **misiones** variadas, integradas en la historia general, ofreciendo diferentes tipos de desafío.
   * Un **clímax** narrativo y una toma de decisiones final con **múltiples desenlaces** (finales alternativos) para aumentar la profundidad y rejugabilidad.  
     Este objetivo implica, además, que el jugador pueda experimentar una sensación de flujo continuo sin interrupciones técnicas ni incoherencias narrativas de principio a fin de la experiencia.
2. **Integración efectiva entre narrativa y mecánicas de juego.** No se buscaba solo contar una historia por un lado y ofrecer desafíos por otro, sino **lograr una integración real** entre ambas dimensiones. Cada misión debía tener una justificación narrativa clara, y cada mecánica implementada debía aportar significado a la historia. Por ejemplo, al reparar el sistema eléctrico de la nave, el jugador no solo avanza en el juego a nivel mecánico, sino que reconstruye un hecho clave del sabotaje ocurrido en la trama. Este objetivo también requería cuidar el ritmo narrativo, de forma que las revelaciones de la historia se distribuyeran equilibradamente a lo largo del avance del jugador, manteniendo siempre el interés.
3. **Crear una experiencia inmersiva con múltiples finales.** Se planteó que el juego fuese más que una serie de retos aislados; debía ser una experiencia emocional en la que el jugador se sintiera realmente dentro de una nave espacial en crisis, tomando decisiones con consecuencias. Para lograrlo se diseñaron:
   * **Ambientes envolventes:** iluminación tenue, sonidos ambientales de maquinaria averiada, espacios claustrofóbicos que generan tensión.
   * **Revelación progresiva mediante grabaciones:** las cintas que el jugador encuentra incrementan el misterio y la tensión conforme se suceden.
   * **Tres posibles finales** distintos, condicionados por la decisión final que el jugador toma tras conocer toda la historia.  
     Este enfoque refuerza la **inmersión** (el jugador se siente protagonista real del suceso) y la **rejugabilidad** narrativa, dos elementos clave en los juegos narrativos modernos, invitando a experimentar diferentes desenlaces.
4. **Aplicar buenas prácticas de ingeniería de software y diseño ágil.** Desde el punto de vista técnico, se fijó como objetivo utilizar principios sólidos de ingeniería de software durante el desarrollo, para asegurar la calidad del código y la facilidad de mantenimiento. Entre estas buenas prácticas se incluyeron:
   * Separación clara de responsabilidades en los scripts (cada clase o módulo debe encargarse de una tarea específica).
   * Uso de patrones de diseño adecuados, como Singleton (ej. un GameManager global) para manejo de estado, y Controladores específicos por misión (Mission1Controller, etc.) para aislar la lógica de cada nivel.
   * Empleo de **prefabs** reutilizables para objetos interactivos comunes (coleccionables, paneles, puertas) y sistemas reutilizables (ej. un manejador genérico de minijuegos).
   * Iteraciones con **pruebas frecuentes**, integración progresiva de funcionalidades y uso riguroso de control de versiones con Git, de forma de detectar y corregir errores rápidamente.

Además, se decidió utilizar una metodología de desarrollo **ágil** (Scrum adaptado a un equipo reducido de una persona) con iteraciones cortas, revisiones semanales y evolución continua del backlog. Esto buscó garantizar entregas incrementales, reducir riesgos, y permitir ajustes sobre la marcha sin perder el control del proyecto.

1. **Documentar el proyecto de forma completa y profesional.** Un objetivo final, pero no menos importante, fue dejar constancia escrita de todo el trabajo realizado mediante una **memoria técnica** clara, estructurada y útil para terceros. Esta documentación debía servir como guía para:
   * Entender la arquitectura y diseño del proyecto.
   * Justificar las decisiones de diseño tomadas durante el desarrollo (por qué se eligieron ciertas mecánicas o soluciones técnicas).
   * Explicar la relación entre la historia narrada y las mecánicas de juego implementadas.
   * Reflexionar sobre los aprendizajes obtenidos y las dificultades superadas.

En síntesis, la memoria cumple una doble función: por un lado, es una presentación académica del proyecto (situándolo en contexto teórico y mostrando resultados), y por otro, es un **registro técnico** del proceso creativo y de desarrollo, que podría ser consultado para futuros trabajos o ampliaciones de *Prometheus-9*.

**2.2. Herramientas utilizadas**

A lo largo del desarrollo de *Prometheus-9* se utilizaron diversas **herramientas tecnológicas** y plataformas, seleccionadas en función de su versatilidad, compatibilidad con Unity y utilidad en el flujo de trabajo. A continuación, se describen las principales:

* **Unity 3D (versión 2021.x):** Es el motor principal del proyecto. Unity fue elegido por su robustez, su amplia documentación, una comunidad activa, y su capacidad para desarrollar experiencias 3D con relativa facilidad. Su sistema basado en componentes permitió diseñar scripts reutilizables, manejar escenas y transiciones, y gestionar eventos y estados del juego de forma modular. Se usaron intensivamente funcionalidades del motor como:
  + **Sistema de eventos** (clases UnityEvent) para modular la interacción entre objetos y sistemas sin acoplamiento fuerte.
  + **Canvas UI** para crear interfaces personalizadas diegéticas (barras de vida y energía, indicadores de cinta reproducida, etc.).
  + **Timeline** y **sistema de cinemáticas** para orquestar la introducción, finales y algunos eventos dramáticos intermedios.
  + **Prefabs** combinados con *physics raycasting* para implementar recolectables e interacciones con objetos del entorno de forma eficiente.
  + **VideoPlayer** integrado de Unity para la reproducción de vídeos de las cintas narrativas dentro del juego.
* **C# (lenguaje de programación):** Todos los scripts lógicos del juego se desarrollaron en C#, aprovechando su orientación a objetos, su claridad sintáctica y su excelente integración con Unity. Algunas características del lenguaje que resultaron útiles fueron:
  + **Delegados y eventos** propios de C# para una comunicación flexible entre scripts (por ejemplo, notificar al GameManager cuando se completa una misión).
  + **Interfaces** para generalizar comportamientos de controladores (ej. interfaces ICollectible o ITriggerable usadas para objetos recolectables o triggers, permitiendo tratarlos de forma polimórfica).
  + **Clases Singleton** (ej. GameManager, InventoryManager) para mantener estados persistentes entre escenas y garantizar una única instancia global accesible.
  + **Bucles de estado en Update()** para gestionar lógica de minijuegos (por ejemplo, PuzzleController, PinCodeController monitorean continuamente condiciones hasta que el jugador resuelve la prueba).
  + **Corutinas** (IEnumerator) para manejar esperas, temporizadores y secuencias asíncronas (por ejemplo, retrasar ciertos eventos, controlar la duración de efectos, etc.).
* **Visual Studio 2019/2022:** El entorno de desarrollo integrado (IDE) utilizado fue Visual Studio, por su compatibilidad con Unity y sus potentes herramientas de edición y depuración. VS ofreció ventajas como:
  + Autocompletado y marcación de errores en tiempo real, acelerando la codificación.
  + **Debugging** con breakpoints y watch variables para inspeccionar el estado del juego en ejecución y encontrar fallos lógicos.
  + Herramientas de **refactorización automática** (renombrado de variables/métodos, extracción de funciones, etc.) que ayudaron a mantener un código limpio a medida que crecía el proyecto.
  + Navegación rápida entre métodos y clases, facilitando el manejo de un código base cada vez más extenso.
* **Git (control de versiones):** Se utilizó Git para llevar el control de versiones de todo el proyecto (código fuente principalmente, pero también algunos recursos). Gracias a Git se pudo:
  + Registrar cada cambio realizado en el código con mensajes descriptivos, creando un historial detallado del desarrollo.
  + Revertir a versiones estables en caso de introducir errores graves, minimizando riesgos.
  + Trabajar con **ramas** separadas para nuevas funcionalidades experimentales (por ejemplo, probar mecánicas de una misión nueva) sin afectar la rama principal.
  + Realizar integraciones controladas de las ramas al comprobar que las nuevas funciones estaban testeadas.  
    En particular, se usó GitHub como repositorio remoto para tener un respaldo del proyecto y permitir mostrar el código a tutores/colaboradores. Esto también facilitó la gestión de *issues* y tareas pendientes de forma básica.
* **Herramientas de diseño gráfico y edición de sonido:** Aunque el foco principal fue la programación, fue necesario crear o adaptar recursos visuales y sonoros básicos:
  + **GIMP / Photoshop:** para edición de texturas (por ejemplo, paneles de la nave con desgaste, interfaces retro en monitores) y creación de elementos de la interfaz de usuario. Se emplearon para retoques sencillos en los gráficos obtenidos de librerías gratuitas, adecuándolos al estilo deseado.
  + **Audacity:** para grabación y edición de las cintas narrativas de audio. Se grabaron voces (con ayuda de colaboradores para interpretar a distintos personajes) y se aplicaron efectos de radio o distorsión para simular transmisiones antiguas. Audacity permitió limpiar el audio, cortar silencios y agregar efectos sonoros.
  + Bancos de sonidos en línea como **FreeSound.org** y la herramienta **Soundly:** se usaron para obtener efectos sonoros ambientales (zumbidos de máquinas, pasos metálicos, alarmas, respiración con traje espacial, etc.) libres de derechos o con licencias apropiadas.
* **Trello (Kanban personal):** Para organizar el trabajo y las tareas se utilizó un tablero estilo *Kanban* (mediante la herramienta Trello). Se definieron columnas como "Pendiente", "En progreso", "En revisión/pruebas" y "Terminado", junto a etiquetas por categoría (por ejemplo: *UI*, *Narrativa*, *Gameplay*, *Bugs*). Esto permitió tener una visión clara del estado del proyecto, priorizar las tareas más importantes y realizar retrospectivas ágiles al finalizar cada iteración, identificando qué fue bien y qué mejorar en el siguiente ciclo.

En conjunto, estas herramientas proporcionaron un ecosistema de desarrollo robusto. Unity y C# dieron soporte a la creación rápida del juego; Visual Studio y Git aseguraron calidad y control en el código; las herramientas gráficas/sonoras permitieron construir la atmósfera narrativa; y Trello sirvió para mantener organizado el progreso. La elección de herramientas fue coherente con el tamaño y objetivos del proyecto, evitando sobrecarga de procesos pero garantizando las funcionalidades necesarias para un desarrollo eficiente.

**Capítulo 3. Fundamentos teóricos y estado del arte**

**3.1. ¿Qué son los juegos narrativos? (historia, clasificación, diseño narrativo, ejemplos clave)**

**Definición y evolución histórica**

El término **juego narrativo** se refiere a una categoría de videojuegos en la que el desarrollo de la historia es el elemento central de la experiencia del jugador. A diferencia de otros juegos tradicionales, cuyo foco suele estar en las mecánicas de acción, en los reflejos o en la obtención de puntuaciones, en los juegos narrativos el jugador participa activamente en el desarrollo de una trama estructurada, a través de elecciones, exploración o interacción con el entorno. En esencia, la narrativa deja de ser un mero telón de fondo para la acción y pasa a ser el núcleo que guía la jugabilidad.

Históricamente, los primeros intentos de incorporar narrativas complejas en videojuegos se remontan a los años 70 y 80 con títulos pioneros como *Colossal Cave Adventure* (1976) o *Zork* (1980), que eran aventuras de **texto** puras. En esos juegos, el jugador leía descripciones y escribía comandos para realizar acciones, viviendo una historia interactiva a través de palabras. Más tarde, empresas como Sierra On-Line y LucasArts popularizaron el género con aventuras gráficas point-and-click, como *Monkey Island* o *King’s Quest*, donde el jugador debía resolver puzles e interactuar con personajes para avanzar en la historia. Estas obras sentaron las bases de cómo contar historias en un medio interactivo.

Con la evolución del hardware y de los motores gráficos en los 90 y 2000, la narrativa en videojuegos fue ganando en complejidad y se fusionó con otros géneros. Surgieron juegos como *Metal Gear Solid*, *Silent Hill*, *Bioshock*, *The Last of Us*, entre muchos otros, que ofrecían tramas dignas de cine o literatura, con alto contenido emocional y cinemáticas elaboradas, pero manteniendo al jugador en control durante buena parte del tiempo. Estos títulos demostraron que era posible equilibrar narrativa y jugabilidad activa, diluyendo la frontera entre juego de acción y juego narrativo.

**Narrativa emergente vs. narrativa estructurada**

Uno de los grandes debates en el diseño narrativo de videojuegos es la contraposición entre **narrativa estructurada** y **narrativa emergente**:

* **Narrativa estructurada:** La historia está predefinida y diseñada previamente; el jugador avanza a través de una trama planificada por los creadores. Se asemeja a experimentar una novela o película interactivamente: aunque el jugador influye en el ritmo o en detalles, los eventos principales ya están escritos. *Prometheus-9* sigue mayoritariamente este modelo, presentando una historia concreta que el jugador descubre gradualmente.
* **Narrativa emergente:** La historia no está determinada de antemano, sino que surge de la interacción del jugador con las reglas del mundo del juego. Aquí, los desarrolladores establecen sistemas y escenarios, pero las historias las “crea” el jugador con sus acciones. Ejemplos clásicos son *Minecraft*, *RimWorld* o *Dwarf Fortress*, donde a partir de mecánicas de simulación resultan anécdotas o aventuras únicas para cada jugador (por ejemplo, la caída de una fortaleza en Dwarf Fortress genera toda una saga espontánea).

*Prometheus-9* se alinea con la narrativa **estructurada**, ya que hay una trama concreta (el sabotaje de la nave y sus consecuencias) que el jugador debe reconstruir. Sin embargo, incluye elementos de exploración y fragmentación no lineal: aunque la historia global está escrita, el jugador puede descubrir las grabaciones (fragmentos narrativos) en distintos órdenes y ritmos, generando su propia **experiencia de descubrimiento**. Esto añade una pizca de narrativa emergente en tanto que cada jugador podría recomponer mentalmente la historia en un orden ligeramente distinto según cómo explore la nave.

**Diseño narrativo en videojuegos**

Tal como plantea Henry Jenkins en su ensayo *“Game Design as Narrative Architecture”* (2004), los juegos no narran historias de la misma forma que el cine o la literatura, sino que **crean espacios narrativos** donde el jugador puede **descubrir fragmentos** y recomponer la historia por sí mismo. Jenkins introduce la idea de que el entorno del juego puede ser el narrador, a través de lo que se conoce como **environmental storytelling**. Siguiendo esa línea, existen varias estrategias de diseño narrativo en juegos:

* **Narrativa ambiental:** El mundo y sus objetos cuentan la historia. Por ejemplo, mensajes escritos en paredes, diarios o notas encontrados, la disposición de cadáveres o destrucción que sugiere eventos previos, grabaciones de audio (como las cintas de *Prometheus-9*), etc. El jugador deduce la trama observando el entorno, sin necesidad de largas exposiciones verbales.
* **Narrativa modular:** La historia se revela en módulos o bloques que el jugador va desbloqueando a medida que avanza, pero que no necesariamente recibe en orden cronológico. Esto exige al jugador reconstruir la secuencia o llenar huecos. Es típico de juegos donde coleccionas fragmentos de memoria, audios, o donde visitas capítulos en desorden temporal.

*Prometheus-9* se apoya fuertemente en ambas estrategias. Por un lado, emplea narrativa ambiental: la nave abandonada está llena de pistas visuales (equipos rotos, puertas forzadas, laboratorios vacíos) que cuentan parte de lo ocurrido antes de que el jugador despierte. Por otro lado, utiliza narrativa modular a través de las 10 grabaciones de seguridad: cada grabación es un bloque de la historia que el jugador obtiene tras una misión, y aunque están numeradas de 1 a 10 (ordenadas para tener sentido), el jugador las experimenta intercaladas con la acción, asimilándolas progresivamente y muchas veces teniendo que deducir conexiones entre ellas.

Al conjugar estos métodos, *Prometheus-9* logra que el jugador se sienta como un **investigador** dentro del juego: explorando la nave (narrativa ambiental) y recogiendo evidencia fragmentada (narrativa modular) para finalmente entender el relato completo.

**Elementos clave de un juego narrativo**

Más allá de las clasificaciones, los juegos narrativos suelen compartir varios elementos fundamentales que les permiten construir una experiencia centrada en la historia. En un juego narrativo exitoso típicamente encontramos:

1. **Historia central (plot):** Una línea argumental clara con planteamiento, conflicto y desenlace. En el caso de *Prometheus-9*, la historia central podría resumirse así: el protagonista despierta solo en una nave espacial aparentemente averiada (planteamiento), descubre que la nave fue **saboteada** por un miembro de la tripulación (conflicto) y finalmente debe decidir cómo resolver la situación (desenlace, con varias posibilidades).
2. **Personajes con motivaciones creíbles:** Protagonistas y secundarios que no son meros avatares vacíos, sino que tienen deseos, miedos y objetivos propios que impulsan la trama. En *Prometheus-9*, aunque gran parte del juego el protagonista está solo, la historia emerge a través de personajes ausentes pero presentes en las grabaciones. Por ejemplo, el especialista en comunicaciones **Cooper** tiene un conflicto moral entre su ansia de descubrimiento (contactar alienígenas) y su lealtad a la misión original, lo que le lleva a tomar decisiones drásticas. Sus motivaciones creíbles dan peso a la narrativa.
3. **Decisiones significativas:** El jugador suele poder tomar decisiones (pequeñas o grandes) que afectan la historia o la percepción de la misma. No todos los juegos narrativos tienen narrativas ramificadas amplias, pero sí ofrecen decisiones que al menos cambian el final o ciertos matices. En *Prometheus-9*, aunque el recorrido de misiones es lineal, la **decisión final** del jugador determina cuál de los tres finales ve y ofrece interpretaciones distintas de los hechos.
4. **Ambiente inmersivo:** Un mundo de juego consistente y detallado que refuerza la historia. Música, sonidos, estilo visual, todo contribuye a sumergir al jugador en la atmósfera narrativa. *Prometheus-9* cuida mucho este aspecto: la ambientación claustrofóbica de la nave, los crujidos metálicos, la falta de música excepto en momentos clave, todo ello sumerge al jugador en la sensación de soledad en el espacio y el suspense de no saber qué pasó allí.
5. **Interacción contextualizada:** Las acciones que realiza el jugador tienen sentido dentro de la historia. Esto es crucial para que narrativa y jugabilidad no se sientan desconectadas. Cada rompecabezas o desafío en *Prometheus-9* está contextualizado (arreglar un reactor, desactivar una alarma, etc.), de modo que el jugador siente que lo que hace **forma parte de la historia**, no es un simple obstáculo artificial.

En conjunto, *Prometheus-9* es un ejemplo de narrativa interactiva de ciencia ficción espacial, donde el jugador:

* Participa activamente en el desarrollo de la historia (no es un espectador; debe explorar y accionar para saber más).
* Reconstruye la historia mediante descubrimiento progresivo (las grabaciones, los entornos).
* Decide el desenlace final (agencia en la conclusión).
* Interpreta los hechos a través no solo de diálogos, sino del entorno, documentos y sonidos, estimulando su imaginación.

Este enfoque permite una conexión emocional más fuerte con la historia, ya que el jugador no solo **juega** la historia, sino que la **descubre y reconstruye** personalmente. La implicación intelectual y emocional es mayor que en juegos donde la narrativa es lineal y frontal.

**3.2. Gamificación: definición, mecánicas, motivación, aplicación en *Prometheus-9***

**Origen y concepto de gamificación**

La palabra **gamificación** (del inglés *gamification*) fue popularizada a inicios de los 2000 por investigadores y diseñadores como Nick Pelling. Se refiere al uso de **mecánicas de juego en contextos no lúdicos** con el objetivo de motivar, enseñar o involucrar al usuario. En otras palabras, consiste en tomar elementos típicos de los juegos (puntos, retos, premios, niveles, etc.) y aplicarlos a ámbitos como la educación, la salud, el marketing, la productividad empresarial, entre otros, para hacer estas actividades más **atractivas y estimulantes**.

La teoría de la gamificación se sustenta en principios de psicología del comportamiento y motivación. Por ejemplo, se apoya en conceptos de condicionamiento operante (B.F. Skinner) y, muy especialmente, en teorías de motivación intrínseca y extrínseca (Deci y Ryan, autores de la teoría de la autodeterminación). Ha sido usada con éxito en aplicaciones tan variadas como plataformas educativas (ej. Duolingo gamifica el aprendizaje de idiomas con puntos y rachas diarias), en la salud y el ejercicio (Fitbit convierte hacer ejercicio en un juego de metas y recompensas), o en gestión del tiempo y hábitos (Habitica, que gamifica tareas cotidianas). En todos estos casos, la gamificación busca aumentar el **compromiso** del usuario y fomentar ciertos comportamientos deseados.

En los videojuegos en sí, aunque pudiera parecer un término redundante (ya que de por sí *son* juegos), la gamificación se refiere a la aplicación **consciente y dirigida** de ciertos principios que aumentan la motivación del jugador más allá de la acción básica. Es decir, diseñar deliberadamente sistemas de recompensas, feedback y objetivos internos al juego para potenciar el engagement, la satisfacción y la permanencia del jugador.

**Principales mecánicas de gamificación**

Según Deterding et al. (2011), algunos elementos clave empleados en gamificación son los siguientes (acompañados de ejemplos típicos y su aplicación concreta en *Prometheus-9*):

| **Mecánica** | **Ejemplo típico en juegos/Apps** | **Aplicación en *Prometheus-9*** |
| --- | --- | --- |
| **Recompensas** | Puntos, logros, premios coleccionables. | Energía recuperada al completar misiones, cintas narrativas desbloqueadas como recompensa. |
| **Penalizaciones** | Perder vidas, retroceder progreso, bloqueos temporales. | Pérdida de salud u oxígeno al fallar ciertos retos o al equivocarse en puzles. |
| **Barras de progreso** | Experiencia, medidores de nivel, barras de carga. | HUD con barra de energía y vida siempre visible, indicando el progreso/restantes del jugador. |
| **Misiones y retos** | Objetivos con instrucciones claras y recompensas definidas. | Misiones con objetivos explícitos; cada misión completada otorga más historia y habilita la siguiente. |
| **Feedback inmediato** | Indicadores visuales/sonoros al acertar o fallar (luces, sonidos, vibración). | Efectos visuales (destellos, chispas) y sonoros (alarma, confirmación) al completar un objetivo o cometer un error. |
| **Desbloqueo por progreso** | Desbloquear niveles o zonas ocultas al avanzar. | Salas de la nave que solo se abren tras completar ciertas misiones; nuevas herramientas que aparecen al lograr objetivos previos. |
| **Decisiones con impacto** | Elegir caminos, decisiones morales, alterar alianza con personajes. | Decisión final con 3 desenlaces narrativos distintos, afectando profundamente el final de la historia. |

Estos elementos, típicos de la gamificación, están diseñados para mantener al jugador motivado: las recompensas brindan satisfacción y sentido de logro; las penalizaciones introducen riesgo y desafío; las barras de progreso muestran claramente el avance hacia metas; las misiones dan estructura; el feedback inmediato refuerza el comportamiento; el desbloqueo añade curiosidad por lo que vendrá; y las decisiones con impacto dan agencia y peso a las acciones del jugador.

**Tipos de motivación en gamificación**

La gamificación trabaja con dos grandes tipos de motivación humana:

* **Motivación intrínseca:** Es el deseo interno de realizar una tarea por el mero gusto de hacerla, por el interés o satisfacción inherente que aporta (descubrir, aprender, superar un reto personal). *Prometheus-9* fomenta la motivación intrínseca mediante su misterio y exploración: el jugador quiere seguir jugando porque genuinamente desea **descubrir** qué ha sucedido en la nave y cómo termina la historia. La narrativa fragmentada, los enigmas y la atmósfera despiertan la curiosidad y el interés propios.
* **Motivación extrínseca:** Es el impulso por obtener recompensas externas o evitar castigos. Viene de fuera de la actividad en sí (puntos, premios, reconocimiento). En *Prometheus-9*, aunque no hay un sistema tradicional de puntos o logros, sí hay elementos extrínsecos: el jugador recibe **cintas** (fragmentos de historia como recompensa), **desbloquea** nuevas salas, y **recupera recursos** (energía, oxígeno) por su avance exitoso. También evita consecuencias negativas al jugar bien (no agotar su salud, no activar alarmas que le resten recursos).

En *Prometheus-9*, se genera así un **ciclo de motivación** bien calibrado: el jugador explora → resuelve un reto → recibe una recompensa (narrativa o recurso) → obtiene más información de la historia → quiere seguir explorando para saber más. Este bucle combina intrínseco (el descubrir la historia) y extrínseco (premios concretos por avanzar) para sostener el interés durante todo el juego.

**Gamificación como diseño consciente en *Prometheus-9***

En *Prometheus-9*, la gamificación no es un añadido superficial o un “pegote” sobre la experiencia, sino que está **profundamente entrelazada con la narrativa y la jugabilidad**. No se incluyen puntuaciones por matar enemigos ni tablas de clasificación (elementos que no tendrían sentido en este contexto narrativo); en su lugar, cada decisión, cada misión y cada objeto recogido tiene un peso funcional *y narrativo*.

Por ejemplo: cortar el cable correcto en un panel eléctrico no otorga “puntos” en una pantalla de score, sino que permite restablecer un sistema vital de la nave **dentro de la ficción** y, como consecuencia, el jugador desbloquea una nueva grabación de audio que aporta información de la trama. Del mismo modo, fallar un puzle o una tarea no resulta en una simple pantalla de "Game Over", sino que típicamente **impacta al jugador** reduciendo su salud o energía, aumentando la tensión y el realismo (el jugador siente las consecuencias del error en el propio sistema del juego, como lo sentiría el personaje en la historia). Esto refuerza la inmersión narrativa: las mecánicas gamificadas (recompensas/penalizaciones) se traducen en **eventos del mundo del juego** en lugar de métricas abstractas.

En síntesis, *Prometheus-9* utiliza la gamificación de manera orgánica: barras de vida/energía que refuerzan la tensión de supervivencia, misiones y desbloqueos que guían al jugador sin sacarlo de la ficción, y un diseño de decisiones que apela a la motivación del jugador por ver las distintas consecuencias. La gamificación está al servicio de la experiencia emocional y narrativa, incrementando la motivación del jugador para avanzar **dentro del mundo ficticio** (y no únicamente por lograr un número o trofeo). Esta integración consciente hace que el jugador casi no perciba “gamificación” como tal, sino que simplemente se siente inmerso en un juego cuya estructura le resulta estimulante y gratificante.

**Capítulo 4. Metodología de trabajo**

**4.1. Metodologías ágiles aplicadas (Scrum adaptado, backlog, sprints)**

**Contexto y necesidad de un enfoque ágil**

El desarrollo de un videojuego como *Prometheus-9*, que combina narrativa interactiva, mecánicas variadas y sistemas interdependientes, exige un enfoque de gestión que permita **flexibilidad**, adaptación constante y entrega incremental de resultados. A diferencia de proyectos de software con requisitos fijos y bien definidos desde el inicio, un videojuego (y más uno centrado en la narrativa) es un producto muy iterativo: muchas ideas se van probando sobre la marcha, algunas funcionan y otras deben descartarse; el **feedback** (retroalimentación) durante el desarrollo puede hacer pivotar ciertos aspectos; se descubren requisitos nuevos según se avanza; etc. Por tanto, resultaba natural adoptar algún modelo **ágil** de trabajo en lugar de una metodología rígida en cascada.

Ante esta realidad, se optó por utilizar una metodología ágil inspirada en **Scrum**, adaptada al entorno de desarrollo reducido (básicamente un desarrollador principal, con alguna colaboración puntual). Esta adaptación permitió mantener un ritmo de trabajo constante pero sin burocracia innecesaria. La metodología ágil tuvo un impacto directo en la calidad del proyecto, ya que permitió iterar sobre ideas narrativas y mecánicas a medida que se implementaban, corrigiendo el rumbo rápidamente cuando algo no funcionaba como se esperaba y refinando aquello que sí funcionaba.

**Aplicación específica de Scrum en el proyecto**

**Product Backlog y planificación inicial:** Desde el inicio se construyó una lista de tareas o *Product Backlog* que se dividió en categorías para cubrir todos los frentes del proyecto:

* Funcionalidades base del motor de juego (movimiento del jugador, sistema de cámara, interacciones básicas, interfaz mínima HUD).
* Módulos del sistema (p. ej.: gestor de estado global *GameManager*, sistema de inventario, sistema de gestión de misiones).
* Misiones (1 a 10) desglosadas en sus pasos clave, requisitos previos y necesidades de pruebas para asegurarse de su correcta implementación.
* Narrativa (escritura de las grabaciones, integración del audio de voz, escenas de cámara para reproducirlas).
* Feedback y QA (testeo en cada iteración, corrección de bugs, balance de dificultad de puzles).
* Finales alternativos y sus condiciones de desbloqueo.

Cada ítem se priorizó (alta, media o baja prioridad). Las tareas más críticas (por ejemplo, las relacionadas con las primeras mecánicas fundamentales o con la infraestructura narrativa como el sistema de reproducir cintas) se abordaron primero, en *sprints* cortos de aproximadamente 1-2 semanas.

**Sprints (ciclos de desarrollo iterativo):** Se definió que cada sprint tendría objetivos concretos. Por ejemplo, un sprint podía tener como objetivo “Implementar la misión 4 completa con su puzle y la grabación asociada”. Durante ese sprint, se estimaba el tiempo de cada tarea (días o horas) y se trabajaba en orden lógico: primero asegurar la interacción básica (el jugador puede activar el puzle), luego integrar la interfaz necesaria, más tarde añadir efectos o transiciones para pulir la presentación, etc. Al final del sprint se hacía una revisión y se integraba la funcionalidad con el resto del juego.

**Revisiones y retrospectivas:** Al término de cada sprint o ciclo, se realizaba una revisión funcional completa de lo añadido o modificado:

* ¿La misión implementada se puede jugar de principio a fin sin errores?
* ¿Los estados del juego (variables, objetos, puertas abiertas/cerradas) se reinician correctamente si el jugador falla o reinicia?
* ¿Las recompensas (p. ej. desbloqueo de una nueva cinta) se activan de forma coherente al completar la misión?
* ¿La interfaz refleja adecuadamente los cambios de estado (mensajes de misión cumplida, actualización de la barra de energía, etc.)?

En caso de que algo fallara o resultara poco intuitivo para el jugador, ese punto se anotaba en el backlog como tarea de mejora para el siguiente sprint. De este modo había una **retroalimentación constante**: el desarrollo avanzaba por iteraciones, pero siempre volviendo a pulir y arreglar detalles de lo ya hecho.

**Herramientas empleadas en la gestión ágil:** Aunque ya se mencionó en el apartado de herramientas, es importante destacar el rol de:

* *Trello* como tablero Kanban digital para visualizar las tareas en columnas ("Por hacer", "En progreso", "En revisión", "Hecho"). Se movían las tarjetas según progresaban las tareas, dando un vistazo rápido al estado del proyecto en cualquier momento.
* *Git* con ramas específicas: cada nueva misión o característica sustancial se desarrollaba en una rama separada, permitiendo trabajar en paralelo a la versión estable. Por ejemplo, mientras la misión 5 estaba en la rama mission5-dev, se podía corregir un bug crítico en la rama principal y luego integrar. Esto es un principio ágil de integrar frecuentemente pero sin romper la build jugable.
* Documentos compartidos (Google Drive) para narrativa: aquí se escribía el guion de las grabaciones, la cronología de la historia y se hacían revisiones de texto con colaboradores, independiente del código, asegurando que la historia estuviera bien pensada mientras se programaba.

**Resultados de adoptar una metodología ágil:** Algunos beneficios concretos que se observaron al aplicar este enfoque fueron:

* Se pudieron ajustar los niveles de dificultad **a mitad del proyecto** tras realizar pruebas con jugadores: por ejemplo, el puzle del reactor (misión 6) resultó inicialmente demasiado difícil, pero gracias a probarlo temprano se rediseñó para que fuera más intuitivo.
* Se reordenaron ciertas grabaciones narrativas tras detectar que el orden original revelaba demasiado pronto un dato clave de la historia. La flexibilidad del desarrollo permitió este cambio sobre la marcha.
* Se refactorizó el sistema de misiones en mitad del proyecto para hacerlo más **flexible**: inicialmente era más rígido y costaba reiniciar una misión individual sin reiniciar todo el juego; tras la refactorización, se pudo implementar checkpoints y reinicios por misión de forma más limpia.
* Se detectaron y corrigieron bugs de forma continua, evitando la acumulación de errores al final. Cada sprint incluía su propio mini-QA, lo que llevó a una versión final más estable.

En definitiva, la metodología ágil aplicada hizo posible que *Prometheus-9* creciera de forma controlada sin comprometer la visión original del juego. Incluso siendo un proyecto llevado por una sola persona, organizar el trabajo en sprints y backlog dio claridad de objetivos a corto plazo y permitió ir presentando avances tangibles (misiones jugables, características completas) periódicamente, manteniendo la motivación alta y facilitando ajustes a tiempo.

**4.2. Arquitectura del sistema (scripts, módulos, gestión de escenas, controladores)**

**Filosofía de diseño**

Desde la fase de diseño técnico se optó por una **arquitectura modular**, separando claramente las responsabilidades de cada componente para evitar dependencias innecesarias. Esta decisión de diseño de software permitió:

* Tener un sistema fácil de **depurar**, ya que los problemas podían aislarse a módulos específicos.
* Posibilidad de **reutilización de código** en futuras misiones u otros proyectos (por ejemplo, el módulo de inventario o el de reproducción de cintas se podrían extraer y usar en otro juego).
* **Aislamiento de errores:** un fallo en la lógica de una misión afecta solo a esa misión y no a todo el juego, puesto que cada una corre en su propio controlador.
* **Escalabilidad:** sería posible añadir nuevas misiones, objetos o interacciones sin reescribir la base del código, simplemente creando nuevos módulos siguiendo la misma estructura.

En términos de patrones, se aplicó principalmente el patrón **Singleton** para los gestores globales (un único GameManager persistente), y una variante del patrón **Controlador** para cada misión (similar al patrón *State* donde cada misión es un estado autocontenido del juego). También se siguió una arquitectura de **escenas** separadas en Unity para aislar contextos (menú, juego principal, finales), comunicándose a través del estado persistente.

**Componentes principales del sistema**

A alto nivel, los componentes o módulos de software principales de *Prometheus-9* son:

1. **GameManager.cs** – *Gestor global de estado:*
   * Implementado como Singleton global, persiste entre escenas (usando DontDestroyOnLoad).
   * Guarda estados esenciales del juego, como la vida y energía actuales del jugador, misiones completadas, grabaciones desbloqueadas y cuál final ha sido elegido.
   * Provee métodos públicos para que otros scripts modifiquen estos estados de forma controlada (por ejemplo GameManager.Instance.AddEnergy(int valor) para sumar energía).
   * Controla la carga y cambio de escenas (por ejemplo, cargar la escena de final apropiada cuando corresponde) y coordina la reproducción de cintas narrativas en el flujo de juego.
2. **MissionXController.cs (misiones 1 a 10)** – *Controladores de cada misión:*
   * Cada misión del juego tiene su propio script controlador que hereda de una clase base o implementa una interfaz común.
   * Dentro de cada controlador se define: condiciones iniciales de la misión (por ejemplo, que el jugador haya recogido cierto objeto para poder comenzarla), la activación del desafío o minijuego de esa misión, la detección de **éxito** o **fracaso** en la misión, y la aplicación de consecuencias.
   * Consecuencias típicas al completar una misión: aumento de energía disponible, desbloqueo de una nueva grabación (cinta) en el inventario de narrativas, apertura de puertas hacia nuevas secciones de la nave. Consecuencias por fallar: pérdida de vida, quizá reinicio de algunos elementos de la misión.
   * Estos controladores están diseñados para ser lo más autónomos posible, comunicándose con el GameManager solo para leer o actualizar el estado global (por ejemplo, marcar una misión como completada). Esto asegura bajo acoplamiento: cada misión maneja su lógica internamente.
3. **InventoryManager.cs y CollectiblesController.cs** – *Sistema de inventario y coleccionables:*
   * InventoryManager (posiblemente integrado en GameManager o aparte) gestiona la lista de objetos clave que el jugador ha recogido y que puede usar. Objetos como tarjetas de acceso, herramientas, etc., se registran aquí.
   * CollectiblesController se encarga de la lógica de recogida de objetos en escena: detecta cuando el jugador entra en contacto con un objeto recogible o pulsa una tecla cerca de él, muestra indicaciones (“Presiona E para recoger”) y notifica al InventoryManager cuando algo es recogido.
   * Los objetos tienen identificadores únicos (por ejemplo, tags como "Wrench" para una llave inglesa, "KeyCard1" para una tarjeta). Al recogerlos, el inventario puede marcar flags que ciertas misiones revisan (p.ej., la Misión 3 puede requerir que InventoryManager confirme que el jugador tiene la "KeyCard1" antes de permitir acceso a la sala).
   * Este sistema también se ocupa de mostrar iconos o nombres de objetos en la interfaz cuando son obtenidos, manteniendo al jugador informado de su equipo.
4. **Controladores de puzles/minijuegos (PuzzleController, PinCodeController, Puzzle8Controller, etc.)** – *Scripts dedicados a mecánicas específicas:*
   * Cada puzle complejo o minijuego dentro de las misiones tiene un script propio que maneja su lógica. Por ejemplo, PinCodeController podría gestionar un panel de introducción de código PIN: detectar los dígitos que el jugador introduce, comprobar si coinciden con la clave correcta, y comunicar éxito o fracaso.
   * Estos scripts activan interfaces de puzle (por ejemplo, muestran un panel en la pantalla), controlan el input específico del puzle (cancela el control normal del personaje mientras se resuelve el puzzle) y determinan las condiciones de victoria o derrota en el mismo.
   * Al completarse un puzle, suelen llamar a métodos del MissionController correspondiente o del GameManager para notificar que la misión puede avanzar. Si el jugador falla (por ejemplo, excede el tiempo, introduce un código erróneo demasiadas veces), informan al controlador de misión para aplicar penalizaciones.
   * Técnicamente, muchos de estos usan el sistema de **UI de Unity** (canvases y panels) para presentar el rompecabezas sin salir de la escena principal, y pueden aprovechar **corutinas** para manejar temporizadores o esperas.
5. **PlayerUI.cs y UI\_Controller.cs** – *Control de la interfaz de usuario in-game:*
   * Se encargan de dibujar y actualizar el **HUD** (Head-Up Display) y demás elementos de interfaz. Esto incluye: barras de vida y energía del jugador, indicadores de objetivos o dirección, iconos de objetos recogidos, mensajes contextuales (por ejemplo, “Acceso denegado” si intenta abrir una puerta sin llave, o “Reparación completada” al terminar una tarea).
   * Estos scripts escuchan eventos o consultan el estado en GameManager para reflejar la información actual. Por ejemplo, si GameManager reduce la vida, el PlayerUI actualiza la barra de vida gráfica.
   * También manejan menús en juego, como una posible pantalla para reproducir las grabaciones encontradas (un menú donde el jugador puede escoger escuchar nuevamente las cintas) o pantallas de pausa/objetivos.
   * Usan animaciones simples (fade in/out, parpadeo) para transiciones suaves, por ejemplo al mostrar una notificación en pantalla o actualizar un indicador.
6. **CameraController (ej. camera18Controller.cs, PlayerCameraController.cs)** – *Control de cámaras virtuales:*
   * Dado que *Prometheus-9* emplea varias escenas y situaciones, existen controladores para la cámara. Por ejemplo, PlayerCameraController controla la cámara principal durante la jugabilidad normal (posiblemente una cámara en tercera persona o primera persona según el diseño, con seguimiento del jugador).
   * Camera18Controller y similares pueden referirse a cámaras específicas usadas en cinemáticas o eventos (por ejemplo, una cámara estática mostrando una escena cuando se reproduce una grabación importante).
   * Estos scripts son responsables de *cambiar* la vista cuando es necesario: en cinemáticas, desactivar el control del jugador sobre la cámara y activar cámaras predefinidas que encuadran la acción dramática; o volver al control del jugador al terminar.
   * También pueden añadir efectos de cámara (temblores cuando hay una explosión, movimientos suaves en secuencias narrativas) para realzar la narrativa audiovisual.
7. **Scripts de triggers e interacción (CheckPlayerPosition.cs, GazeTrigger.cs, TriggerFinal.cs, etc.)** – *Detección de condiciones in-game:*
   * Muchos eventos en el juego ocurren cuando el jugador llega a cierto lugar o mira cierto objeto. Los scripts de tipo *Trigger* detectan estas situaciones usando mecanismos de Unity:
     + CheckPlayerPosition puede comprobar si el jugador está dentro de un área (collider trigger) específica; al entrar, puede activar un evento (p.ej., encender las luces de una sala, o lanzar un diálogo).
     + GazeTrigger podría referirse a detectar si el jugador está mirando (con la cámara) un objeto importante durante un tiempo, para entonces desplegar información (como leer un rótulo, o enfocar un objeto clave en una escena).
     + TriggerFinal específicamente detecta si el jugador realiza la acción final (por ejemplo, acciona una consola de comunicaciones) que desencadena la secuencia final; dependiendo de la elección del jugador en ese momento, carga una escena de final u otra.
   * Estos scripts suelen ser sencillos: al activarse, llaman a otros componentes (como GameManager o un MissionController) para efectuar cambios de estado (por ejemplo, marcar la misión final como completada y almacenar qué elección hizo el jugador para mostrar el final correcto).
8. **Controladores de cinemáticas y diálogo (VideoPlayerController.cs, Conversation.cs, etc.)** – *Manejo de secuencias narrativas:*
   * VideoPlayerController gestiona la reproducción de vídeos (las grabaciones y posiblemente las escenas final de cada ending). Cuando el jugador decide ver una cinta recogida, este controlador se encarga de reproducir el vídeo correspondiente en pantalla (o en la UI diegética, como un monitor del juego). También puede pausar el juego mientras tanto, y notificar cuándo termina la reproducción.
   * Conversation.cs podría manejar alguna conversación o monólogo del jugador. Sin NPCs presentes físicamente, puede que esto no se use mucho salvo para monólogos internos.
   * Ambos controladores pueden también encargarse de cambiar ciertos aspectos del entorno durante las secuencias narrativas: por ejemplo, atenuar la iluminación de la escena mientras se reproduce un vídeo o desactivar el movimiento del jugador.
   * Asimismo, al finalizar una cinta, estos scripts pueden lanzar eventos en el juego: habilitar una nueva misión ahora que se obtuvo la información, o actualizar un objetivo.

**Comunicación entre componentes:**  
Los distintos scripts y módulos se comunican entre sí de forma controlada mediante varias técnicas:

* Llamadas a métodos públicos de otros scripts (por ejemplo, GameManager.Instance.CompleteMission(id) puede ser llamado por un MissionController cuando su misión termina exitosamente).
* Uso de **eventos/delegados** de C# o eventos de Unity (OnTriggerEnter, OnMouseDown, eventos personalizados) para notificar acciones sin acoplar directamente los componentes.
* Uso de **flags** (variables booleanas) y enumeraciones de estado en GameManager para que otros scripts chequeen estados globales (por ejemplo, un DoorController puede revisar if(GameManager.Instance.Mission4Completed) para decidir si abrir una puerta).

Gracias a esta arquitectura, cada sistema conoce solo lo necesario y la mayoría de interacciones son orquestadas a través del GameManager o eventos. Esto resulta en un código más mantenible, donde agregar una nueva misión significa principalmente crear un nuevo MissionController y quizás nuevos puzles, pero sin tocar los otros sistemas.

**4.3. Evolución del proyecto: fases desde la conceptualización hasta el pulido final**

El desarrollo de *Prometheus-9* se dividió, de facto, en varias **fases** bien diferenciadas, aunque fluidas. A continuación se describen esas etapas, sus objetivos y los hitos alcanzados en cada una:

**Fase 1. Conceptualización (Diseño inicial):**

* Se definió la premisa básica del juego: *survival narrative* en una nave espacial abandonada, con un misterio a resolver mediante la exploración y resolución de puzles. Se decidió desde el principio que no habría combate, enfocando la tensión en lo psicológico y lo ambiental.
* Se elaboraron las fichas de los personajes que existirían en la historia (aunque no todos aparezcan directamente, sí a través de las grabaciones). Por ejemplo, se definieron perfiles para **Madison** (la capitana), **Cooper** (comunicaciones, el eventual saboteador), **Nicole** (científica), etc., incluyendo sus motivaciones y su papel en los eventos anteriores al juego.
* Se escribió una **cronología de los eventos** previos al juego: qué ocurrió en la nave *Prometheus-9* antes de que el jugador despierte. Esto sirvió como esqueleto narrativo para luego distribuir en las grabaciones y en el diseño de niveles (por ejemplo, saber que hubo un sabotaje en la sala de máquinas, o una confrontación en el puente, para reflejarlo en esas áreas).
* También en esta fase se hizo un **esquema de misiones** tentativo: se listaron 10 posibles misiones/escenarios que encajarían con la historia (restaurar energía, reparar comunicaciones, investigar laboratorio, etc.), aunque sujetos a cambio luego.

**Fase 2. Prototipado (vertical slice inicial):**

* Se crearon versiones iniciales muy básicas de las mecánicas principales: movimiento del jugador (controles WASD, colisiones, gravedad), mirada libre, etc.
* Se implementó un prototipo de **recolección de objetos** (un objeto sencillo en el suelo que el jugador podía tomar, y aparecía un mensaje de confirmación).
* Se implementó la interacción con **paneles** o consolas simples: por ejemplo, acercarse a un panel y pulsar una tecla para activarlo, que encendía una luz (esto sentó la base para futuros puzles).
* Se desarrolló el primer minijuego prototípico: un panel de interruptores donde el jugador debía activar una secuencia correcta (representando quizá restablecer la energía). Esto fue muy importante para entender el tipo de lógica que tendrían los puzles en Unity.
* En paralelo, se diseñaron los **scripts base fundamentales**: el GameManager se creó aunque fuera vacío al comienzo, así como el PlayerUI para dibujar una HUD básica, y el InventoryController. Esta cimentación permitió que las siguientes características se construyeran ya con una estructura sobre la que encajar.

**Fase 3. Iteración y desarrollo progresivo:**

* Usando la metodología ágil mencionada, se procedió a implementar **misión por misión**. Cada misión (del 1 al 10) se desarrolló de forma relativamente independiente, siguiendo la lógica conceptualizada pero ajustándola según surgían dificultades o nuevas ideas.
* Después de completar cada misión, se hacían pruebas internas de juego (jugando esa misión en contexto, desde el inicio hasta completarla, integrándola con las anteriores). Esto permitió comprobar la coherencia entre misiones y la estabilidad técnica a medida que el juego crecía.
* El feedback obtenido en estas iteraciones llevó a numerosos ajustes, por ejemplo:
  + Rediseñar la misión de los conductos de ventilación, que inicialmente era excesivamente extensa y laberíntica, volviéndola más breve y orientada (añadiendo luces de guía, por ejemplo).
  + Simplificar el puzle del código PIN en la misión de comunicaciones, ya que se observó que la combinación de pistas era demasiado obtusa para el jugador promedio.
  + Añadir **checkpoints automáticos** tras cada misión importante, al notar que sin guardados intermedios, un fallo tardío obligaría a repetir mucho. Con los checkpoints, si el jugador muere en la misión 7, puede reiniciar desde el inicio de esa misión en vez de recomenzar todo el juego.

**Fase 4. Integración narrativa y cinemática:**

* Una vez todas las misiones estaban funcionales, se centró el esfuerzo en integrar completamente la **narrativa**. Se incorporaron las cámaras virtuales y se escribieron/implementaron las **cinemáticas** necesarias: la intro (escena inicial mostrando al protagonista despertando, quizás con una voz de fondo de registro de la misión), y sobre todo las 3 cinemáticas final(es) diferentes para los desenlaces.
* Se grabaron y trataron los **audios de las 10 cintas**. Esto implicó coordinar con actores de voz o sintetizar voces para los personajes. Cada grabación de 20-30 segundos fue editada para tener calidad aceptable y se integró en el juego usando el sistema de video o audio (algunas podían ser solo audio con subtítulos, otras vídeo).
* Se revisó la **coherencia de la historia** ya dentro del juego: se probó jugar de principio a fin leyendo/escuchando todas las grabaciones en orden para asegurar que la historia se entiende y que todas las piezas encajan (se ajustaron textos de algunas cintas para aclarar puntos oscuros o eliminar redundancias).
* En esta fase también se implementaron pequeños **detalles narrativos** en gameplay: por ejemplo, quizás un efecto cuando el jugador recoge la última cinta (un sonido especial indicando que ya tiene todos los datos), o diálogos de voz en off breves cuando algo sorprendente ocurre.

**Fase 5. Pulido final, QA y documentación:**

* Se llevó a cabo una optimización general del código y los assets. Esto incluyó eliminar piezas de código no usadas, comentarios temporales, optimizar bucles, asegurar la destrucción de objetos innecesarios para liberar memoria, etc.
* Se corrigieron errores de la interfaz de usuario (alineaciones, textos cortados, traducciones si las hubiera), y en general de usabilidad. También se ajustó el **feedback visual** de algunas acciones para hacerlas más claras (por ejemplo, hacer parpadear un panel cuando una misión relacionada está activa, para guiar al jugador).
* Se realizaron pruebas de juego completas con personas ajenas al desarrollo (compañeros, amigos) para obtener impresiones frescas. Con este testing final se hicieron ajustes de balance de dificultad en un par de puntos, se arreglaron los últimos bugs reportados y se verificó que el juego corría fluidamente en distintos equipos.
* Finalmente, se **redactó esta documentación** (la memoria del proyecto), recopilando todos los aspectos teóricos, técnicos y de diseño aprendidos o implementados. Se incluyeron reflexiones del desarrollador sobre problemas encontrados y soluciones adoptadas, para dejar un registro completo del proceso.

Tras estas fases, *Prometheus-9* alcanzó un estado **beta** funcional en el que cumplía con todos los requisitos planteados inicialmente, listo para ser presentado como un juego narrativo completo. El proceso fue iterativo y enriquecedor, permitiendo que la visión inicial (que podría haberse quedado en una idea) cobrara vida en un producto jugable concreto, gracias a un trabajo planificado pero flexible.

**Capítulo 5. Resultados**

Este capítulo documenta los resultados obtenidos tras la implementación completa del videojuego *Prometheus-9*, analizando tanto el **diseño del juego** (aspectos narrativos, de ambientación y jugabilidad) como la **implementación técnica** y la arquitectura final, así como el impacto que tuvieron la planificación y metodología utilizadas en el producto acabado. El proyecto evolucionó desde una propuesta conceptual hasta convertirse en una experiencia interactiva funcional, inmersiva y coherente, tanto a nivel técnico como narrativo.

**5.1. Diseño del juego: narrativa, ambientación, jugabilidad, estructura de misiones**

**Historia, universo y tono**

La historia de *Prometheus-9* se basa en un conflicto de corte tecnológico, moral y existencial. El jugador/protagonista despierta solo en una nave estelar tras un aparente accidente, encontrando los sistemas críticos averiados y ningún rastro de la tripulación. A medida que explora y repara sistemas, va descubriendo grabaciones de audio/vídeo y elementos del entorno que revelan que la situación fue causada por una acción deliberada: **Cooper**, el especialista en comunicaciones, saboteó los sistemas de la nave en un intento desesperado por contactar con una inteligencia alienígena detectada en las cercanías. Este acto extremo puso en peligro a toda la tripulación y desencadenó la crisis que el jugador debe resolver.

El tono del juego es **serio, introspectivo y tenso**, inspirado en obras de ciencia ficción y suspense como *Alien* (1979) en su atmósfera claustrofóbica, *Interstellar* (2014) en sus dilemas científicos/morales, o el videojuego *SOMA* (2015) en su exploración de temas existenciales y de soledad. Se evita deliberadamente la acción frenética; en lugar de eso se prioriza una ambientación silenciosa y opresiva, donde el jugador debe prestar atención a detalles visuales y sonoros sutiles. El misterio y la incertidumbre se mantienen durante toda la experiencia, con algunos picos emocionales en momentos clave: ciertas **grabaciones** revelan traiciones y sacrificios, y la eventual confrontación con el resultado del contacto alienígena genera tensión máxima.

**Jugabilidad centrada en la narrativa**

La jugabilidad en *Prometheus-9* está diseñada al servicio de la historia. Cada mecánica o misión no es un “relleno”, sino que tiene un propósito dentro del relato. Algunos ejemplos de cómo los desafíos del juego están integrados con la narrativa:

* **Restablecer la energía de la nave (Misión 1):** No es solo un puzle técnico inicial, sino la acción lógica que cualquier persona haría al despertar en la oscuridad total. Narrativamente permite que el jugador abandone la zona inicial y descubra la magnitud del problema (¿por qué se apagó todo?).
* **Analizar una muestra biológica extraña (Misión 5):** No es una mecánica aislada de laboratorio; su resultado revela la **naturaleza del contacto alienígena** que Cooper buscaba, dando un giro a la comprensión de la historia.
* **Recalibrar la navegación (Misión 9):** No es solo resolver un panel de navegación; es lo que abre la posibilidad de **escapar o quedarse** en la nave. Es la antesala de la decisión final del jugador, justificando jugablemente que tenga esa opción.

En resumen, hay una **lógica interna** fuerte: los retos están contextualizados. Esto fortalece la coherencia narrativa y la motivación del jugador, pues siente que cada acción tiene un propósito más allá de “avanzar en el juego”; está avanzando en la **historia**.

**Estructura de juego progresiva**

El flujo del juego sigue una estructura de progresión narrativa lineal en términos de misiones, donde cada misión completada desbloquea nuevas zonas de la nave y nuevas piezas de la historia. No obstante, se incorpora una ramificación narrativa al final, permitiendo al jugador elegir entre tres desenlaces distintos según su interpretación de los hechos descubiertos y sus valores.

La progresión general de la experiencia es la siguiente:

1. **Zona inicial:** El juego comienza con un apagón total, sin acceso a casi ninguna sala. El jugador debe orientarse en la oscuridad con herramientas limitadas hasta reactivar la energía.
2. **Tras Misión 1:** Al completar la primera misión (restaurar la energía), se activan sistemas básicos de la nave y se **desbloquean** puertas hacia las primeras áreas (ej. las cubiertas principales).
3. **Misiones intermedias (2–9):** Cada misión adicional que el jugador completa repara algún subsistema o consigue información, lo que típicamente abre más puertas o funcionalidades. A su vez, cada misión otorga una nueva **grabación** que arroja más información narrativa. Así, a medida que avanza, el jugador gana acceso a niveles más profundos de la nave (laboratorio, sistemas de soporte vital, comunicaciones, etc.) y simultáneamente entiende mejor qué pasó.
4. **Clímax (Misión 10) y desenlace:** Al alcanzar el 100% de restauración de energía y completar las tareas críticas (misión 10), el jugador obtiene todos los datos necesarios y finalmente tiene la oportunidad de activar el sistema de despegue/comunicaciones. Allí enfrenta la **decisión final** que conducirá a uno de los finales.

Este diseño logra mantener el interés narrativo y la sensación de avance tangible. El jugador percibe claramente su progreso: más áreas explorables y más información en su poder según va completando misiones. La estructura lineal garantiza que el jugador no se pierda ninguna parte esencial de la historia, pero dentro de ello tiene libertad para explorar las áreas abiertas en el orden que quiera antes de avanzar a la siguiente misión, si así lo desea.

**Misiones con identidad propia**

Cada una de las 10 misiones fue concebida con una **identidad distinta** en términos de ambiente y mecánica, para evitar la monotonía y ofrecer una experiencia variada al jugador. Se estableció una curva de dificultad progresiva y una diversidad de estilos de juego, alternando entre misiones de distinto tipo:

* Puzles lógicos y de ingeniería (ej. recomponer circuitos, descifrar códigos).
* Búsqueda e interacción con objetos (tipo *fetch quests* sencillas o uso correcto de ítems en lugares adecuados).
* Pruebas de **atención y memoria** (recordar secuencias de sonidos, patrones vistos en monitores, etc.).
* Eventos con **presión temporal** o contrarreloj (escapar de un área antes de que se agote el oxígeno, estabilizar un reactor antes de que explote).
* Elementos de **sigilo o evasión** en al menos una sección, para añadir tensión de evitar un peligro activo.

*Ejemplo detallado: Misión 8 - Neutralización del alienígena.* En esta misión, se descubre finalmente que una forma de vida alienígena ha entrado a la nave:

* El jugador detecta movimiento no humano en la bodega de carga mediante sensores que activó en la misión previa.
* Comienza un segmento de **sigilo**: el jugador debe descender a la bodega y evitar ser detectado por la criatura alienígena que patrulla el área.
* La tarea es llegar hasta la compuerta de carga y activar manualmente la esclusa para expulsar al intruso al espacio.
* Si el jugador es visto por la criatura, esta lo ataca y muere *instantáneamente* (siendo probablemente el único caso de muerte inmediata en el juego), reiniciando la misión desde el último checkpoint.
* La tensión en esta misión es máxima, ya que combina mecánica (sigilo) con la culminación del misterio (el primer y único encuentro directo con lo desconocido). Representa el **clímax jugable** justo antes de los eventos finales de narrativa.

Este ejemplo ilustra cómo cada misión ofrece algo único: distinta mecánica, distinta emoción. En conjunto, la variedad de misiones mantiene la frescura y también sirve a la historia (pues cada misión revela una faceta distinta del conflicto: técnica, científica, de supervivencia, moral, etc.).

**Grabaciones como núcleo narrativo**

Las 10 **grabaciones de seguridad** (audio/vídeo) que se desbloquean con cada misión completada representan el recurso narrativo clave del juego. Se diseñaron como micro-cortos cinematográficos con fuerte carga dramática, y su función es triple:

1. **Exposición narrativa:** Brindan información que el jugador no podría obtener solo con la exploración. Son básicamente flashbacks o registros del pasado que rellenan lagunas de la historia (por ejemplo, una grabación muestra a la capitana discutiendo con Cooper, revelando sus intenciones).
2. **Construcción del misterio:** Al no presentarse en orden cronológico exacto, revelan detalles de forma fragmentada. Esto estimula la curiosidad del jugador, que tras cada cinta puede replantearse lo que creía saber. Las grabaciones están numeradas pero el jugador quizá imagine eventos ocurridos entre ellas.
3. **Emoción y conexión humana:** Los registros incluyen voces de los tripulantes, diálogos intensos, decisiones humanas con consecuencias trágicas. Esto humaniza a la tripulación ausente y genera empatía o antipatía. El jugador no ve personajes físicamente, pero los **conoce** a través de sus palabras y decisiones grabadas.

Cada cinta tiene una duración media de 20 a 40 segundos. Se presentan con audio (voces actoralmente interpretadas) y en algunos casos con video estático o animaciones sencillas (por ejemplo, imágenes de la tripulación antes del lanzamiento, o una visualización de la consola de comunicaciones con la transcripción). El jugador puede acceder a ellas desde un menú especial (un apartado tipo “Registro de Cintas” en la interfaz) o reproduciéndolas en una sala específica de la nave habilitada como sala de comunicaciones. Se procuró que las cintas fueran reproducibles nuevamente en caso de que el jugador quisiera revisarlas (por ejemplo, repasar la cinta 3 cuando ya vio la 7, para conectar pistas).

Este sistema de grabaciones mantuvo al jugador motivado narrativamente: siempre hay **ganas de conseguir la siguiente cinta** para entender más, funcionando casi como las páginas de un diario dispersas en un survival horror, o los logs de audio en juegos como *System Shock* o *BioShock*.

**Decisión final: ramificación narrativa**

Una vez reunidas todas las cintas y completadas todas las misiones, el jugador es **convocado** (dirigido mediante objetivos) a la sala de comunicaciones de la nave. Allí debe tomar una decisión crucial, eligiendo uno de tres cursos de acción posibles para resolver la situación:

* **Enviar una señal de auxilio y regresar a la Tierra.** (Final 1) – Implica dar por perdida a la tripulación desaparecida y priorizar informar a la Tierra de lo ocurrido, llevándose los datos del contacto alienígena.
* **Poner rumbo en búsqueda de la tripulación desaparecida.** (Final 2) – Implica suponer que quizá la tripulación escapó o fue llevada por el ente alienígena y decidir ir tras ellos en vez de volver a casa de inmediato.
* **Rastrear las señales alienígenas y permanecer investigando.** (Final 3) – Implica una decisión más arriesgada: en lugar de huir, el protagonista elige seguir el rastro del alienígena en el planeta o sector espacial cercano, buscando un contacto a costa de su propia seguridad.

Cada final se representa mediante una **cinemática** distinta y va acompañado de un mensaje o escena final de tono reflexivo, con cierre suficiente pero a la vez dejando espacio a la interpretación del jugador. No se califica ningún final como “bueno” o “malo” explícitamente; cada uno representa una filosofía (seguridad vs. deber vs. curiosidad) y tiene sus propias implicaciones morales que el jugador puede ponderar.

Esta introducción de **rejugabilidad narrativa** permite que el jugador, si lo desea, repita el juego tomando una decisión diferente al final para ver otro desenlace y quizás captar matices distintos (por ejemplo, ciertos diálogos en las grabaciones pueden cobrar otro significado sabiendo un final diferente). Aunque el juego en sí no cambia antes de la decisión final, la posibilidad de múltiples finales añade valor narrativo y extiende la experiencia más allá de una sola partida, invitando a reflexionar qué elección fue la más apropiada según lo vivido.

**5.2. Arquitectura final del juego: estructura técnica, optimizaciones, pruebas y control de calidad**

**Escenas y estructura del proyecto en Unity**

El juego completo se dividió en varias **escenas** en Unity, organizadas por su función, pero conectadas entre sí mediante el flujo del juego y el GameManager que persiste datos:

* **MainScene:** Contiene la jugabilidad principal. En esta única escena se ubicó la mayor parte de la nave espacial y donde transcurren las 10 misiones, con el jugador controlable. Es, por así decirlo, el “mundo” del juego. Se optó por integrar todas las secciones de la nave en una misma escena para evitar cargas intermedias durante la exploración. El GameManager maneja qué partes están accesibles en cada momento.
* **IntroScene:** Una escena separada para la introducción cinemática. Antes de dar control al jugador, se reproduce una corta secuencia que presenta la misión de la nave *Prometheus-9* y muestra quizás al protagonista despertando de criosueño o del golpe del accidente.
* **FinalScene1 / FinalScene2 / FinalScene3:** Tres escenas distintas, una para cada final. Tras la decisión final del jugador en la MainScene, se carga la escena correspondiente donde se ejecuta la cinemática de ese final (por ejemplo, FinalScene2 mostrará la secuencia del protagonista iniciando la búsqueda de sus compañeros en las coordenadas X).
* **MenuScene:** La escena de menú principal (pantalla de título), que incluye también opciones (ajustes básicos de audio, controles) y los créditos finales. Desde el menú se puede iniciar una nueva partida (cargando IntroScene) o salir.

Cada escena está conectada mediante el GameManager y llamadas de cambio de escena. El GameManager guarda información persistente necesaria al cambiar (por ejemplo, cuánto de vida/energía lleva el jugador al pasar a un final, o qué final se debe cargar tras la decisión) para que la transición sea consistente. Se procuró que las transiciones fuesen limpias, usando fundidos a negro antes de cargar la siguiente escena, para la presentación profesional.

**Módulos y scripts clave (resumen técnico final)**

La lógica del juego quedó dividida en módulos independientes cohesivos, escritos en C# como se detalló en la arquitectura de capítulo 4. A modo de resumen, estos son algunos de los **scripts principales** y su responsabilidad en la versión final:

* **GameManager.cs:** *Gestor global.* Sigue siendo el Singleton central. Mantiene el estado del jugador (posición, salud, energía), la misión actual en curso, las cintas narrativas desbloqueadas y la elección final. Controla la transición de escenas (Intro → Main → Finales), reproduce las cintas (lanzando el VideoPlayerController) y gestiona la lógica global de fin de juego.
* **Mission1Controller.cs ... Mission10Controller.cs:** *Controladores individuales de misiones.* Cada misión tiene su script independiente que maneja activación, condiciones de éxito/fallo y qué ocurre al completarla. Por ejemplo, Mission3Controller habilita un minijuego de reactivar oxígeno, espera a que PuzzleOxigenoController informe de éxito, luego notifica al GameManager para que incremente la reserva de aire y desbloquee la siguiente zona.
* **InventoryManager.cs & CollectiblesController.cs:** *Inventario y objetos.* Controlan la recogida, uso y visualización de objetos clave. CollectiblesController detecta colisiones con ítems en el mundo y comunica a InventoryManager la adquisición. InventoryManager actualiza la UI (por medio de PlayerUI) para mostrar el nuevo objeto. También expone métodos para comprobar posesión de objetos (usados por las misiones, p. ej. una puerta chequea InventoryManager.HasItem("KeycardA")).
* **UI\_Controller.cs y PlayerUI.cs:** *Interfaz de usuario.* Gestionan todo el HUD y elementos de interfaz interactiva. PlayerUI actualiza la barra de salud/energía, los indicadores de objetivo, y muestra u oculta los paneles de interacción (por ejemplo, abre el menú de cintas cuando el jugador lo solicita). UI\_Controller puede englobar menús completos o superponer pop-ups contextuales (como “Misión X completada!” por unos segundos).
* **CameraController.cs (y específicos como camera18Controller.cs):** *Control de cámara.* Manejan la lógica de qué cámara está activa. CameraController principal se asegura de seguir al jugador en gameplay normal. Otros scripts de cámara se activan en cutscenes: por ejemplo camera18Controller puede ser un script que activa una cámara fija #18 cuando el jugador llega a cierto punto, ofreciendo una vista dramática (un momento cinematográfico).
* **Conversation.cs y TriggerFinal.cs:** *Eventos narrativos.* Conversation.cs podría manejar algunas líneas de diálogo automáticas (monólogos del personaje al examinar algo, u oír una voz grabada). TriggerFinal.cs específicamente monitorea la decisión final: detecta qué botón/elemento elige el jugador en la consola final y setea la variable de final correspondiente en GameManager antes de llamar a cargar la escena final.
* **VideoPlayerController.cs:** *Reproductor de cintas.* Maneja la reproducción de los vídeos de las grabaciones y de las escenas finales. Usa el componente VideoPlayer de Unity. Cuando se activa, se encarga de reproducir el clip adecuado (según qué cinta solicita el jugador), mostrar subtítulos si están disponibles, esperar a que termine, y luego retornar el control al juego (reactivando la jugabilidad o cerrando la interfaz de cinta). También es responsable de pausar el juego mientras una cinta se reproduce para evitar que el jugador muera o se mueva inadvertidamente durante la visualización.

Esta lista no es exhaustiva (hay más scripts menores), pero cubre los **pilares** de la implementación final, evidenciando una correspondencia con la planificación de arquitectura hecha en etapas previas.

**Optimizaciones técnicas**

Para garantizar un buen rendimiento y estabilidad del juego en hardware estándar, se aplicaron varias optimizaciones en la fase final:

* Uso intensivo de **Prefabs** reutilizables para elementos repetitivos: puertas, consolas, paneles interactivos, luces de emergencia, etc. En lugar de tener objetos únicos para cada instancia, se creaba un prefab bien configurado y se reutilizaba, ahorrando memoria y facilitando cambios globales (por ejemplo, cambiar el material de todas las puertas en un clic).
* **Ocultamiento/Culling de objetos**: Las secciones de la nave no visibles (por estar tras puertas cerradas) desactivan sus elementos dinámicos para ahorrar CPU. Unity realiza frustrum culling por defecto para gráficos, pero adicionalmente se desactivaban scripts en zonas inactivas. Esto aseguró que el juego corriera a ~60 FPS constantes en equipos promedio, sin caídas notables al moverse por la nave.
* Reducción de luces en tiempo real: Se limitaron las luces realtime y se usaron luces estáticas horneadas cuando fue posible, para minimizar el coste gráfico. También se moderó el uso de efectos de post-procesado (bloom, filtros de color) a solo momentos clave o escenas cinematográficas, manteniendo la escena principal relativamente ligera.
* **Checkpoints automáticos**: Implementados no solo como feature de diseño sino como salvaguarda técnica. Al empezar cada misión clave, el juego guardaba el estado actual (misión, vida, variables) en memoria; si el jugador moría, se restauraba. Esto no solo mejora la experiencia, sino que evitó bugs de encadenamiento de misiones al siempre reiniciar en un estado limpio predeterminado cada misión.
* Pruebas de *memory leaks* y gestión de assets: se revisó que al cambiar de escenas se liberaran los assets no necesarios (ej. destruir objetos de MainScene al ir a un final), evitando consumo extra de memoria.

El resultado es un juego que corre de manera fluida y sin errores críticos. Durante las pruebas finales se comprobó que incluso en momentos de carga (p.ej. en la bodega con el alien, con niebla volumétrica activa y IA corriendo) el rendimiento se mantenía estable.

**Pruebas realizadas (QA) y balance**

A lo largo del desarrollo, y especialmente al final, se realizaron diversas pruebas de jugabilidad con **usuarios reales** (compañeros y testers externos voluntarios). Se facilitó incluso una versión *debug* del juego con interfaz extra para registrar métricas, tales como:

* Tiempo empleado por misión, para detectar cuellos de botella o misiones excesivamente largas.
* Cuál misión resultaba más difícil a cada tester (medido por cuántos intentos o muertes en ella).
* Número de errores o reinicios por parte de cada jugador.

Con estas pruebas se identificaron y corrigieron varios errores o comportamientos no deseados:

* En una versión inicial, la Misión 3 (reparar oxígeno) podía romperse si el jugador realizaba acciones fuera de orden esperado, dejando el juego en un estado inconsistente. Se corrigió añadiendo checks robustos de secuencia o bloqueando acciones hasta que otras previas estén hechas.
* Se descubrió que el alienígena de la Misión 8 no se **reseteaba** correctamente al reiniciar tras una muerte: seguía en la posición donde vio al jugador la última vez. Se arregló reiniciando su posición y estado de alerta en cada intento.
* La barra de energía en el HUD no reflejaba correctamente el progreso si el jugador volvía a una sala anterior y luego regresaba: era un bug de actualización que se solventó haciendo que al entrar en cada sala clave, la UI consulte el GameManager y refresque los valores.

El proceso de QA también condujo a pequeños *tweaks*: por ejemplo, se ajustó la velocidad de movimiento del jugador levemente a petición de testers (muy lenta al inicio, frustraba en la nave grande), y se añadieron más indicaciones sutiles en ciertos puzles para evitar atascos (como un parpadeo en el objeto interactivo relevante si tras mucho tiempo el jugador no progresa).

En general, el **balance** final del juego quedó afinado: ningún puzle es trivial pero tampoco imposible, la curva de aprendizaje funciona (los jugadores entendían las mecánicas básicas antes de enfrentar retos mayores gracias a la introducción escalonada) y la duración total promedio se situó entre 2 y 3 horas, cumpliendo lo previsto.

**Conclusión del capítulo 5 (Resultados):**  
El resultado final de *Prometheus-9* es un videojuego narrativo funcional y **altamente inmersivo**, con mecánicas variadas y una historia que evoluciona junto con la progresión del jugador. Se alcanzaron los objetivos planteados al inicio:

* Desarrollo de un juego completo con 10 misiones distintas, cohesionado y sin errores críticos.
* Integración profunda entre narrativa, entorno y jugabilidad: el jugador experimenta la historia a través de las acciones del juego de forma natural.
* Implementación de sistemas técnicos sólidos, organizados modularmente y eficientes en tiempo de ejecución.
* Inclusión de elementos de gamificación y de diseño emocional que aumentan la motivación (barras de vida, recompensas, tensión dramática).
* Uso de recursos multimedia (audio, video, entorno 3D) para enriquecer la historia más allá de texto o diálogos simples.
* Ofrecimiento de decisiones significativas al jugador (los finales múltiples) que aportan profundidad y personalización de la experiencia.

Además, el producto final es fácilmente **escalable**: la arquitectura permite añadir más grabaciones, nuevas misiones o incluso más finales sin tener que modificar la base existente, lo cual es un indicador de buen diseño de software. También se consideró la **portabilidad**: el juego está preparado para ser compilado en otras plataformas (por ejemplo, una versión WebGL jugable en navegador, o portarlo a Android con controles táctiles adaptados) si se desease en el futuro.

En resumen, *Prometheus-9* pasó de ser un concepto a un juego real que demuestra cómo un proyecto independiente, con recursos limitados pero con una visión clara, puede ofrecer una experiencia completa y significativa al jugador.

**Capítulo 6. Conclusiones y propuestas de trabajo futuro**

El desarrollo de *Prometheus-9* ha supuesto un proceso de aprendizaje exhaustivo sobre lo que implica diseñar, programar y pulir un videojuego narrativo con múltiples capas: historia, jugabilidad, ambientación, decisiones y tecnología. En este capítulo se recogen las **conclusiones** derivadas de la experiencia, así como los aprendizajes y competencias adquiridas, las limitaciones identificadas en el producto logrado y las rutas posibles para una evolución futura del proyecto.

**6.1. Conclusiones generales**

**El valor de unir narrativa y jugabilidad:**  
Uno de los principales logros de *Prometheus-9* ha sido demostrar cómo la narrativa no debe ser un añadido decorativo, sino un **núcleo estructural** del juego. En muchos títulos comerciales, la historia se presenta a través de cinemáticas separadas del gameplay, lo que a veces rompe la inmersión o se siente desconectado de lo que el jugador *hace*. En este proyecto, se procuró que cada mecánica, misión y objeto estuviera **integrado en la historia**: las acciones jugables se habilitan o cobran sentido cuando narrativamente es coherente, y la progresión del jugador en los niveles es simultáneamente progresión en la trama. Esta unión diseño-narrativa hizo que testers señalaran que se sentían motivados a resolver puzles porque querían saber qué pasaba, no solo por el reto mismo, lo cual era exactamente la intención.

**La fuerza del diseño modular y escalable:**  
Otro acierto significativo fue la implementación de una arquitectura de software **modular**, inspirada en principios sólidos de ingeniería. Gracias a ello:

* Cada misión es un bloque relativamente independiente que puede modificarse o mejorarse sin afectar a las demás (bajo acoplamiento).
* Sistemas como la interfaz, los datos globales, el inventario o las cámaras están desacoplados entre sí mediante el GameManager y eventos, lo que facilitó depurar y ajustar componentes individuales.
* El sistema resultante puede **escalar fácilmente**: añadir misiones nuevas, más objetos o incluso escenas adicionales (como nuevos finales, expansiones) no requeriría reescribir la base, sólo seguir extendiendo con nuevos módulos.

Esto convierte a *Prometheus-9* no solo en un juego puntual, sino en una **plataforma** sobre la que se podría construir más contenido. La escalabilidad lograda da margen para que el proyecto continúe creciendo si se desea, validando la decisión de invertir tiempo en una buena arquitectura.

**La narrativa como motor emocional:**  
La historia del juego fue construida para provocar emociones en el jugador: miedo ante lo desconocido, sentido de responsabilidad, dudas morales, sensación de aislamiento, etc. Se evitó deliberadamente el uso de combates o violencia explícita para generar emoción; en su lugar, se buscó que el jugador sintiera tensión real al:

* Quedarse sin energía y ver parpadear las luces (miedo a morir en la oscuridad).
* Fracasar una misión crítica y oír alarmas (angustia por las consecuencias).
* Escuchar una grabación con implicaciones morales fuertes (impacto emocional al enterarse de traiciones o sacrificios).
* Enfrentar **una decisión con consecuencias irreversibles** en el final (peso moral sobre sus hombros).

Este enfoque se alinea con una corriente de diseño narrativo más moderna que apuesta por lo **emocional y reflexivo**, más allá del reto mecánico tradicional. *Prometheus-9*, salvando las distancias, trata de situarse en esa categoría de juegos que dejan al jugador pensando después de terminarlos, más que en cuán difícil fue tal jefe final, en los dilemas que presentó y en qué habría hecho uno mismo en esa situación ficticia.

En conjunto, estas conclusiones refuerzan la idea de que incluso un equipo pequeño (o una sola persona) puede crear un juego con profundidad si se centra en integrar bien todos los elementos. *Prometheus-9* consigue involucrar al jugador intelectualmente (resolviendo misterios), emocionalmente (empatizando con la historia) y activamente (superando desafíos), lo cual era la meta deseada.

**6.2. Objetivos alcanzados**

**Revisión exhaustiva de los objetivos propuestos:**  
A continuación se enumeran los objetivos iniciales planteados al proyecto (descritos en el Capítulo 2) y cómo fueron materializados en el producto final:

1. Crear un videojuego narrativo jugable de principio a fin — **Completado:** se desarrolló una experiencia completa con introducción, 10 misiones de desarrollo, un clímax y un desenlace ramificado en tres finales.
2. Integrar historia y mecánicas de forma coherente — **Completado:** cada mecánica (puzles, exploración, sigilo, etc.) está justificada dentro de la narrativa y aporta a la comprensión de la historia.
3. Desarrollar múltiples tipos de interacción (mecánicas variadas) — **Completado:** el juego incluye calibración de sistemas, introducción de códigos, secciones de sigilo, puzles de lógica, minijuegos de habilidad y limpieza/recogida, cubriendo un amplio espectro de interacciones.
4. Aplicar elementos de gamificación para motivar al jugador — **Completado:** se incorporaron barras de salud y energía, penalizaciones por error, recompensas por éxito, un sistema de progresión por misiones y desbloqueo gradual de contenido, manteniendo al jugador comprometido.
5. Desarrollar una arquitectura escalable y mantenible — **Completado:** todos los módulos del juego son reutilizables, están bien documentados en el código y desacoplados entre sí; se pueden extender fácilmente con contenido nuevo sin cambiar la base.
6. Utilizar metodologías ágiles adaptadas al trabajo individual — **Completado:** se llevó un backlog en Trello, con sprints de implementación cortos, retrospectivas y ajustes iterativos constantes que guiaron el proyecto de forma flexible pero controlada.
7. Documentar el proceso de forma rigurosa y profesional — **Completado:** se generó una memoria técnica completa (este documento), que incluye la base teórica, los objetivos, el desarrollo, los resultados y reflexiones críticas sobre el proyecto.

Es satisfactorio comprobar que **todos** los objetivos planteados inicialmente fueron cumplidos. Esto no significa que no haya aspectos mejorables (como veremos en limitaciones), pero sí que el proyecto logró hacer realidad sus aspiraciones fundamentales.

**6.3. Competencias adquiridas**

El proceso de creación de *Prometheus-9* ha supuesto un entrenamiento intensivo en múltiples disciplinas relacionadas con el desarrollo de videojuegos. Entre las competencias más valiosas adquiridas por el desarrollador a lo largo del proyecto, se pueden destacar:

**Competencias técnicas avanzadas:**

* Programación orientada a objetos en C# aplicada a un contexto complejo de videojuego, incluyendo gestión de memoria, optimización y depuración en tiempo real.
* Desarrollo con Unity a nivel intermedio-avanzado: uso de **prefabs**, animaciones, físicas, *Canvas UI*, sistema de eventos y corutinas, integración de plugins (VideoPlayer) y manejo de escenas múltiples.
* Patrones de diseño aplicados a videojuegos: implementación de Singletons, gestores, controladores de estado, máquina de estados sencilla para misiones, etc., entendiendo sus beneficios y riesgos en un entorno de juego.
* Sistemas de interacción compleja: uso de *raycasting*, *triggers* de colisión, detección de mirada (*gaze*) y cómo encadenar condiciones lógicas entre objetos del mundo.
* Gestión de datos globales persistentes entre escenas con DontDestroyOnLoad y serialización simple de estado (aunque no haya un guardado persistente, se maneja la persistencia en la sesión de juego).
* Optimización de rendimiento en Unity: técnicas de culling, baking de iluminación, reducción de draw calls y control de complejidad de escena.

**Competencias creativas y de diseño narrativo:**

* Escritura de **narrativa interactiva**: elaborar una historia fragmentada en registros, diseñar personajes solo a través de audio, crear finales múltiples coherentes, etc. Se practicó la economía narrativa (contar mucho con poco, dado el formato breve de las grabaciones).
* Diseño emocional del jugador: aplicación de principios de **diseño emocional**, generando tensión y alivio alternados, manejando la incertidumbre, etc., con recursos mínimos (silencios, sonidos incidentales).
* Diseño de entornos narrativos: cómo ambientar visual y sonoramente un espacio para que cuente historia (la disposición de elementos en la nave, los detalles visuales de saboteo, etc. fueron un ejercicio de “nivel narrativo”).
* Integración de **temáticas adultas** en un juego (dilemas morales, la soledad), cuidando el tono para que sea adecuado y respetuoso, a la vez que impactante.

**Competencias organizativas y de gestión de proyecto:**

* Planificación ágil: aprender a organizarse con Scrum/Kanban uno mismo, estableciendo *sprints* semanales, priorizando tareas, y adaptándose a cambios sin perder de vista el panorama general.
* Manejo de control de versiones (Git) en solitario: creación de ramas para funcionalidades, merges, resolución de conflictos, mantenimiento de un repositorio limpio. Estas son habilidades transferibles a trabajo en equipo, pero practicadas en un entorno controlado.
* Documentación técnica clara y estructurada: la elaboración de este documento ha reforzado la capacidad de explicar decisiones de diseño y arquitectura, y reflexionar sobre el propio trabajo de manera objetiva y comunicativa.

Estas habilidades no solo son relevantes para el desarrollo de videojuegos, sino que muchas de ellas son transferibles a otros entornos de la ingeniería de software, narrativa interactiva, desarrollo de software educativo o incluso entornos inmersivos como VR/AR. *Prometheus-9*, como proyecto integrador, ha servido como una experiencia formativa completa que abarca desde lo puramente técnico hasta lo creativo y lo organizativo.

**6.4. Limitaciones técnicas y narrativas**

Aunque el proyecto ha cumplido sus objetivos principales, es importante reconocer sus **límites** actuales, tanto en el ámbito técnico como en el creativo, los cuales ofrecen espacio para mejoras:

**Alcance narrativo limitado:**

* La historia, si bien inmersiva, está centrada exclusivamente en el **punto de vista del protagonista** y los registros de la tripulación. No hay escenas donde interactúen personajes en tiempo real (todo sucede en pasado a través de grabaciones).
* No hay eventos que cambien en función de decisiones intermedias del jugador; la ramificación narrativa solo ocurre al final. Esto significa que durante el desarrollo del juego la historia es lineal y no reactiva a las posibles acciones del jugador (por ejemplo, no hay diferentes caminos narrativos en mitad del juego según cómo actúes).
* No existen **diálogos activos** ni sistemas de conversación ramificados. La interacción narrativa es unidireccional (el jugador recibe información de las grabaciones pero no puede influir en ellas más que en el orden de descubrimiento).

**Recursos visuales genéricos:**

* La mayoría de los modelos 3D de la nave y objetos son assets gratuitos o de base, lo que si bien ahorró tiempo, conlleva que la dirección artística no sea totalmente única. Hay cierta mezcla de estilos en modelos que puede restar cohesión visual.
* Falta una coherencia estética completa: los distintos módulos de la nave (laboratorio, ingeniería, puente) podrían tener identidades visuales más diferenciadas. Actualmente comparten muchos elementos genéricos, lo que empobrece un poco la sensación de lugar real.
* Las grabaciones narrativas se muestran al jugador como videos simples 2D superpuestos o audios, en vez de escenas animadas *in-game*. Esto crea una leve desconexión visual (pasar del motor del juego a un vídeo). Limitaciones de producción impidieron hacer, por ejemplo, cinemáticas con modelos 3D de los personajes dentro del propio Unity.

**Linealidad estructural:**

* El jugador **debe completar las misiones en un orden exacto** predefinido. No hay forma de saltar o alternar el orden de resolución, lo que reduce la agencia en la progresión.
* No existe exploración libre significativa ni regreso a áreas anteriores (*backtracking*) con cambios. Si el jugador vuelve a un área pasada, la encontrará prácticamente igual (salvo puertas abiertas). El juego no presenta cambios de estado global perceptibles hasta el final.
* No se puede influir en el **orden de revelación narrativa**: siempre se obtienen las mismas cintas en la misma secuencia, atadas a misiones concretas. Un diseño más abierto podría permitir conseguir pistas en distinto orden, pero no fue el caso por simplicidad.

**Inteligencia artificial limitada:**

* El único “enemigo” con IA, la criatura alienígena de la Misión 8, sigue un patrón simple de patrulla predefinido. No busca activamente al jugador ni aprende; solo reacciona si lo ve en su cono de visión. Esto es adecuado pero básico.
* No hay otros NPCs ni enemigos con comportamientos complejos o dinámicos. La mayor parte del juego la IA es inexistente porque el entorno está vacío de seres vivos (lo cual fue una elección narrativa pero también una limitación en gameplay).
* No se implementó ningún sistema de IA avanzada como guardias con estados de alerta variables, ni IA para compañeros, ni por supuesto algo como **IA emocional** o adaptativa. Todo el comportamiento del juego está *scripted*.

**Duración corta y falta de contenido opcional:**

* La experiencia completa se puede terminar en aproximadamente 2-3 horas una vez conocido el juego. Para un jugador nuevo quizá se extienda a 4 horas si explora lentamente, pero sigue siendo una duración modesta. Esto es apropiado para el alcance, pero en comparación con juegos comerciales es corto.
* No hay **rejugabilidad mecánica** más allá de ver distintos finales. Es decir, el núcleo jugable será igual en cada partida; el incentivo para rejugar es solo narrativo (ver finales diferentes), ya que no hay modos extra, logros, dificultades alternativas, etc.
* No existen logros, secretos escondidos ni caminos alternativos de resolución de puzles. Todo es bastante guiado y lo opcional es mínimo (por ejemplo, no hay cintas “secretas” fuera de las 10 principales que premien la exploración extra).

Estas limitaciones eran en parte esperadas dado el alcance (un proyecto académico/indie). Muchas surgen de compromisos tomados para poder finalizar el juego en tiempo y con recursos disponibles (por ejemplo, mantener la linealidad y no añadir más ramificaciones debido a coste de producción). Identificarlas claramente es útil para planear cómo podría evolucionar el proyecto para superarlas.

**6.5. Trabajo futuro: propuestas de mejora y expansión del universo**

Pensando en convertir *Prometheus-9* en un producto más completo, inmersivo y pulido, a continuación se proponen diversas **líneas de expansión y mejora** que podrían abordarse en trabajos futuros. Se organizan por categoría:

**Ampliación narrativa:**

* Incorporar **diálogos interactivos** en tiempo real: por ejemplo, mediante un sistema de comunicación por radio con algún superviviente o IA de la nave, donde el jugador pueda elegir respuestas. Esto añadiría dimensiones a la narrativa y rompería la soledad absoluta, permitiendo decisiones morales durante el juego (no solo al final).
* Añadir personajes supervivientes con los que interactuar, ya sea encontrando algún miembro de la tripulación herido o con via de comunicación. Incluso si son hologramas o inteligencia artificial de la nave, darían más presencia a otros personajes.
* Incluir decisiones morales **a lo largo del juego**, no solo al final. Por ejemplo, decidir entre dos cursos de acción en una misión (sacrificar X sistema para salvar Y información, etc.) afectando ligeramente eventos posteriores.
* Crear más finales, algunos **secretos o desbloqueables** tras ciertas condiciones. Por ejemplo, un final extra si se consigue una cinta oculta con información clave, o un final alternativo “malo” si el jugador falla en evitar una catástrofe hacia el final.

**Mejora de la estructura jugable:**

* Introducir un sistema de **exploración libre** una vez se completan ciertas misiones: por ejemplo, tras la mitad del juego, permitir que el jugador deambule y aborde algunas tareas en orden libre, dandole más agencia sobre el flujo (siempre que narrativamente se ajuste).
* Permitir **múltiples enfoques** para algunas misiones. Actualmente los puzles tienen una solución. Se podría diseñar que ciertas misiones tengan dos formas de resolverse (por ejemplo: reparar un sistema hackeando el panel o físicamente cambiando una pieza, a elección del jugador). Esto aumentaría la sensación de control y rejugabilidad.
* Crear contenido opcional: salas ocultas, logs adicionales, recursos o easter eggs que no sean necesarios para terminar el juego pero enriquezcan el lore o supongan un reto para completistas.

**Evolución audiovisual:**

* Diseñar una **identidad visual única** para cada módulo de la nave. Por ejemplo, que el laboratorio tenga una estética distinta (más limpia, científica) frente a la sala de máquinas (industrial, sucia) o el puente (tecnología avanzada). Actualmente se intuye pero podría reforzarse con modelos y texturas específicas.
* Sustituir las cinemáticas de grabaciones en vídeo por **eventos interactivos in-game**. Quizá para una versión mejorada, en vez de mostrar un vídeo al encontrar una cinta, se podría recrear la escena en el propio motor, como un holograma 3D que el jugador presencia dentro de la nave, haciendo las transiciones menos bruscas.
* Mejorar los efectos de sonido con capas reactivas: por ejemplo, que si la salud del jugador es baja se oigan latidos o respiración agitada, o si el oxígeno está bajo haya silbidos de aire. Actualmente el audio cumple su función pero podría enfatizar aún más el estado del juego.
* Implementar transiciones visuales narrativas: efectos en pantalla que reflejen la narrativa, como vibraciones y desenfoques si la nave sufre una sacudida, o un efecto de interferencia visual cuando el jugador ve una grabación dañada. Añadirían inmersión sensorial.

**Enriquecimiento del sistema de IA:**

* Desarrollar un sistema de **visión y oído** para enemigos, si se introducen más secciones de sigilo o algún enemigo adicional. Que reaccionen a pasos o ruidos generados por el jugador (por ejemplo, si corre o lanza un objeto).
* Introducir comportamientos adaptativos en la IA: por ejemplo, patrullas que cambien rutas si detectan puertas abiertas, o enemigos que busquen refugio si son atacados (si se incluyera combate no letal, por ejemplo).
* Implementar reacciones a eventos del entorno: que la criatura o posibles robots de la nave respondan a luces apagadas (busquen cómo encenderlas), a sonidos de alarmas, etc., para dar vida a la nave y hacerla más impredecible.

**Ampliación del universo del juego:**

* Crear un **DLC narrativo** contando la historia desde otro punto de vista, por ejemplo controlar a Cooper antes/durante el sabotaje, o vivir lo que ocurrió tras uno de los finales. Esto podría ser una mini-campaña adicional que expanda el lore.
* Diseñar una **secuela** directa: Por ejemplo *Prometheus-10*, donde el jugador regresa al planeta origen de la señal con un nuevo equipo o afronta las consecuencias del final elegido. Una secuela permitiría aprovechar la base pero introduciendo entornos nuevos y quizá más personajes.
* Expandir el lore con material externo: documentos, logs extensos tipo novela gráfica, o una base de datos dentro del juego con información de la misión, la compañía que envió la nave, la especie alienígena, etc., para contentar a jugadores que quieran profundizar más en la historia detrás de la historia.

**Portabilidad y accesibilidad:**

* Preparar versiones para otras plataformas (WebGL para jugar en navegador, builds para Android/iOS) adaptando los controles y optimizando recursos para dispositivos móviles. Sería interesante hacer el juego accesible en la web dada su duración corta, podría llegar a más gente.
* Adaptar los controles a pantallas táctiles o mandos de consola, con interfaz ajustable, de modo que la experiencia sea cómoda fuera de PC.
* Implementar opciones de **accesibilidad**: por ejemplo, posibilidad de ajustar el tamaño del texto de subtítulos, modo daltónico (cambiar ciertos puzzles de color), incluir un narrador opcional que describa sonidos importantes para jugadores con dificultades auditivas, etc. Esto haría el juego disfrutable por un público más amplio.

Estas propuestas conforman un plan de mejoras que, de llevarse a cabo, podrían convertir *Prometheus-9* de un muy buen prototipo/proyecto final, en un producto casi comercial con mayor profundidad y pulido. Por supuesto, cada mejora conlleva un trabajo significativo, pero la base construida durante el proyecto facilita muchas de ellas, gracias a la escalabilidad lograda.

**Conclusión final**

*Prometheus-9* no es solo un juego terminado para efectos académicos: se concibió y se concretó como una **experiencia emocional**, un proyecto técnico sólido, y una propuesta narrativa que demuestra cómo los videojuegos pueden contar historias con profundidad, tensión, dilemas y belleza. A lo largo del desarrollo se integraron disciplinas diversas para servir a una visión unificada, y el resultado es un mundo interactivo pequeño pero completo.

Este prototipo es solo el primer paso. La historia de la nave, de sus tripulantes y del contacto alienígena apenas ha comenzado a contarse. Las bases están sentadas. El sistema funciona. El universo de ficción existe. Lo que falta, de aquí en adelante, es ir más allá: dar vida a más historias en este universo, expandir los sistemas de juego, y buscar conectar aún más intensamente con los jugadores a nivel emocional y lúdico.

Como ocurre al final del juego, ahora es el desarrollador quien debe elegir: ¿regresar a lo conocido, dando por concluido el proyecto, o **aventurarse más allá de las estrellas** y seguir explorando las posibilidades que ofrece *Prometheus-9* en el futuro? En esas manos queda la próxima decisión.