Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Creación de un Juego para el Aprendizaje del Manejo de Punteros en C

Alumno:
Sergio Lamata López

Tutores:
Alma Pisabarro Marrón
Carlos Enrique Vivaracho Pascual

...

Agradecimientos

Quiero agradecer a mi familia, amigos y a todos los que han estado a mi lado durante los meses de trabajo.

En especial a mis tutores Alma y Carlos, y también a Diego por darme tantas facilidades y hacer este proceso mucho más ameno.

Resumen

El presente Trabajo de Fin de Grado se enmarca en la ampliación de la plataforma educativa del grupo GREIDI, cuyo propósito es fomentar el aprendizaje a través de videojuegos serios.

Este proyecto ha consistido en el desarrollo de un videojuego utilizando la plataforma Unity 2D, con el objetivo de asistir a alumnos de diversas disciplinas que podrían no tener una amplia experiencia en programación.

El videojuego desarrollado se enfoca en explicar el funcionamiento de los punteros en el lenguaje de programación C, poniendo especial enfoque en las operaciones que se pueden hacer con ellos, como el asterisco y el ampersand, facilitando así la comprensión de este concepto fundamental a través de una metodología interactiva y lúdica.

Abstract

The present Final Degree Project is framed within the expansion of the educational platform of the GREIDI group, whose purpose is to promote learning through serious video games.

This project has consisted of developing a video game using the Unity platform, with the aim of assisting students from various disciplines who may not have extensive programming experience.

The developed video game focuses on explaining the functioning of pointers in the C programming language, with a special emphasis on the operations that can be performed with them, such as the asterisk and ampersand, thus facilitating the understanding of this fundamental concept through an interactive and playful methodology.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

El objetivo de este proyecto es el desarrollo de un videojuego mediante el motor Unity, orientado al ámbito educativo. Este será un juego serio que formará parte de la iniciativa GREIDI, destinada a gamificar los conceptos enseñados en asignaturas de diversos estudios de grado y máster. La aplicación se desplegará en la plataforma web del grupo anterior mencionado, la cual alberga diversos videojuegos con el mismo propósito.

A través del desarrollo de este proyecto, se pretende explorar el uso de videojuegos como herramientas de aprendizaje en centros educativos. La gamificación consiste en la utilización de técnicas, elementos y dinámicas propias de los juegos y actividades recreativas con el fin de fomentar la motivación, mejorar la productividad y facilitar el aprendizaje. Esta técnica busca aplicar conceptos de los videojuegos a tareas ajenas a ellos, de manera que se faciliten el logro de objetivos, comúnmente relacionados con el aprendizaje.

En nuestro caso, todo esto se llevará a cabo mediante la creación de un minijuego que se centrará en explicar el concepto de puntero en el lenguaje de programación C, así cómo los distintos operadores que se le pueden aplicar.

1.1. Motivación

Existen ciertos conceptos de la programación que pueden ser más complejos de entender que otros, sobre todo para los alumnos de primer año, Con este proyecto se pretende preparar a los estudiantes en uno de los conceptos más desafiantes de C desde el enfoque de los videojuegos. Este enfoque permite a los estudiantes hacer en vez de simplemente ver, todo ello desde un entorno más amigable y conocido dado que la gran mayoría de estudiantes han jugado antes a un videojuego. Mediante el uso de los videojuegos, se quiere dar al alumno un aprendizaje ameno y estimulante, que lo inste a aprender activamente en vez de ser un sujeto pasivo en este proceso.

1.2. Objetivos

- **Objetivo 1:** Este proyecto tiene como objetivo principal la creación de un videojuego, *Puntero Aventura*, con la intención de enseñar los punteros del lenguaje de programación C. Para ello tendrá diversos niveles que aumentarán su dificultad y los conceptos abarcados.
- **Objetivo 2:** Además del objetivo principal, en este proyecto también se abarca la modificación de un juego ya existente en la plataforma GREIDI, "Mars Miners", para así adaptarlo al lenguaje de programación C y cambiar sus niveles, ya que se consideraba que la dificultad y los conceptos que este juego enseñan son insuficientes.

1.3. Contexto

La industria de los videojuegos ya no es algo nuevo ni en la vida cotidiana, ni en las aulas. Al contrario, cada vez está creciendo más. Este concepto de llevar los videojuegos al aula se conoce como gamificación y tiene diversos beneficios [1]:

- **Aprendizaje interactivo:** convierte algo "obligado" como puede ser aprender en algo que el alumno busca activamente
- **Disminuye el miedo al fracaso:** no es lo mismo que al alumno piedra un nivel que puede volver a intentar las veces que sea a recibir una mala nota en bolígrafo rojo.
- **Aplicaciones en el mundo real:** Al ser los videojuegos más inmediatos, el alumno puede ver en qué afectan sus elecciones en el juego
- **Visibiliza el aprendizaje:** Muchas veces el alumno se pregunta cuánto le queda para terminar. Con los videojuegos, puede ver exáctamente cuánto le queda para terminar su objetivo y cuál es su siguiente paso.

Capítulo 2

Plan de Proyecto

El plan de proyecto es una sección crucial que detalla el esquema general del desarrollo del videojuego. Esta sección abarca la metodología, los recursos necesario, el plan deriesgos, y la planificación necesaria para llevar a cabo el proyecto con éxito.

2.1. Metodología utilizada

Para el desarrollo de este proyecto se ha decidido utilizar el marco de trabajo Ágil [2], debido a que se ha considerado útil en el ámbito del desarrollo de los videojuegos, y específicamente en un TFG, al requerir retroalimentación por parte de los tutores. Más concretamente, se usará la *Scrum*, un marco de trabajo ágil, con sprints de duración variable. El marco de trabajo Ágil se usará para el ciclo de vida del producto. Sin embargo, durante el ciclo de vida del proceso de desarrollo del software nos basaremos en los 5 grandes *workflows* del Proceso Unificado. Es decir, en cada *sprint* seguiremos un proceso de Requisitos, Análisis, Diseño, Implementación y Pruebas. En la figura 2.1 se puede apreciar el flujo de trabajo de un proyecto que usa la metodolgía ágil

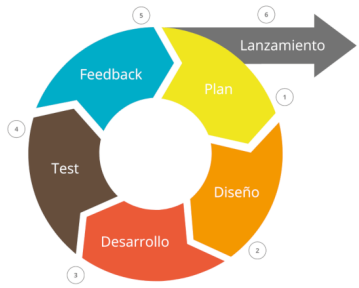

Figura 2.1: Representación del marco de trabajo ágil

Con este método, y al disponer de un equipo de trabajo tan pequeño (2 tutores y 1 alumno), los tutores harían el papel de *Scrum Master*, mientras que el alumno sería el *equipo de desarrollo* al completo

2.2. Presupuesto

El sueldo de un diseñador junior de videojuegos en España está en torno a los 18.000€ brutos anuales [3]. Eso se traduce en un coste de 9'37€/hora de trabajo, y con 300 horas de trabajo, el coste de personal asciende a 2811€

Los costes de material suponen el ordenador portátil que se ha usado para trabajar, estimado en 500€.

Las herramientas software utilizadas son gratuitas. En el caso de Unity, tiene una licencia para estudiantes que permite su uso gratuito. Otros recursos, como pueden ser los *assets* utilizados, se han tenido en cuenta, pero no afectan al coste final del proyecto al haber utilizado recursos gratuitos.

Se puede ver el total de los costes en la tabla 2.1.

Tipo de coste	**Cantidad (€)**
Trabajador	2811€
Material	500€
Recursos	0€
TOTAL	3311€

Tabla 2.1: Tabla de costes

2.3. Plan de riesgos

Un riesgo [4] en un proyecto se refiere a cualquier evento o condición incierta que, si ocurre, puede afectar positiva o negativamente los objetivos del proyecto. Los riesgos pueden surgir de diversas fuentes y pueden impactar diferentes aspectos del proyecto, incluyendo el cronograma, el costo, la calidad, el alcance y los recursos.

Un plan de riesgos cuenta con las siguientes fases:

- Identificación y análisis
- Priorizar riesgos

- Realizar un plan de riesgos
- Supervisar los riesgos
- Respuesta al riesgo (en caso de manifestarse)

A continuación se detallan los riesgos identificados, su descripción, probabilidad, impacto, medidas preventivas y el plan de contingencia.

Riesgo 01	Enfermedad
Descripción	El alumno contrae una enfermedad que le impida trabajar
Probabilidad	Baja
Impacto	Medio
Medidas Preventivas	Un estilo de vida saludable
Plan de contingencia	Dejar tiempo de margen entre actividades

Tabla 2.2: Riesgo 01

Riesgo 02	Fallo en equipo de trabajo
Descripción	El equipo en el que se trabaja deja de funcionar y se imposibilita el trabajar
Probabilidad	Baja
Impacto	Alta
Medidas Preventivas	Hacer copias de seguridad con regularidad
Plan de contingencia	Obtener un segundo equipo de trabajo y restaurar las copias en él

Tabla 2.3: Riesgo 02

Riesgo 03	Cambio de requisitos
Descripción	Se debe revisitar trabajo ya hecho pues los requisitos han sido modificados
Probabilidad	Alta
Impacto	Baja
Medidas Preventivas	El uso de metodología ágil permite flexibilidad en requisitos
Plan de contingencia	Dejar tiempo de margen entre actividades

Tabla 2.4: Riesgo 03

Riesgo 04	Dificultad de uso de las herramientas
Descripción	Se emplea más tiempo del esperado en aprender a usar las herramientas
Probabilidad	Baja
Impacto	Media
Medidas Preventivas	Las herramientas utilizadas ya se conocen
Plan de contingencia	Dejar tiempo de margen entre actividades

Tabla 2.5: Riesgo 04

Riesgo 05	Fallo en el canal de comunicación
Descripción	El medio de comunicación falla y es imposible comunicarse
Probabilidad	Baja
Impacto	Alta
Medidas Preventivas	Establecer canales de comunicación claros y regulares
Plan de contingencia	Buscar otro canal de comunicación

Tabla 2.6: Riesgo 05

2.4. Planificación Inicial

La planificación inicial es una fase crucial en cualquier proyecto, ya que establece las bases para el desarrollo exitoso del mismo. Permite establecer una hoja de ruta clara y estructurada, facilitando la gestión eficiente del proyecto y asegurando que todos los miembros del equipo y stakeholders estén alineados con los objetivos y expectativas.

Al seguir un marco de trabajo ágil, no se identifican fases fijas del proyecto como pueden ser el diseño o la implementación, sino que cada sprint tendrá se compondrá de todas esas fases, en mayor o menor medida.

Habiendo hecho un estudio preliminar de los videojuegos encontrados en la plataforma GREIDI y sus características, se estiman 6 sprints de duración variable en este proyecto: 2 para la modificación del juego mencionado anteriormente y 4 para la creación del nuevo juego.

2.4.1. Resumen planificación

Planificación del Proyecto	
Fecha Inicio	29/04/2024
Fecha Fin	01/07/2024
Tiempo Sprints	Entre 1 y 2 semanas
Carga Trabajo Diaria	Entre 5 y 7 horas
Horas Totales Previstas	Aproximadamente 300 horas

Tabla 2.7: Planificación Inicial

Capítulo 3

Seguimiento del Proyecto

En esta sección se detallarán los esfuerzos realizados en cada sprint, si este fue exitoso y el tiempo invertido. Para una clasificación de grano más fino del tiempo, este se desglosará en las siguientes actividades:

- **Diseño:** Se refiere al tiempo empleado para crear el modelo que seguirán las clases del videojuego.
- **Código:** Se refiere al tiempo empleado en escribir las clases y scripts del videojuego.
- **Búsqueda e Investigación:** Tiempo empleado en aprender sobre el lenguaje de programación, las herramientas usadas, o entender código.
- **Cambios en requisitios/Solución de errores:** El tiempo empleado en tratar con los cambios de requisitos entre sprints o solucionar lo que no se consiguió en sprints anteriores.

3.1. Sprint 1

Este primer *sprint* tuvo una semana de duración, y se centró en el inicio del desarrollo del proyecto. Las tareas incluían la instalación y aprendizaje del software a utilizar. Durante este *sprint* se asigna el juego ya existente a modificar, Mars Miners. Este juego ocupa la mayor parte del tiempo del *sprint*, tratando de entender el código y las estrucuras que lo componen. Tras este sprint se alcanzó un gran entendimiento del juego, por lo que se considera **exitoso**.

Resumen Sprint 1	
Fecha Inicio	29/04/2024
Fecha Fin	06/05/2024
Diseño	5 horas
Código	0 horas
Búsqueda e Investigación	30 horas
Cambios en Requisitos/Solución de errores	0 horas
TOTAL	35 horas

Tabla 3.1: Resumen Sprint 1

3.2. Sprint 2

Este *sprint* se centra en, una vez entendido el código de Mars Miners, realizar las modificaciones necesarias. Durante el *sprint* se realizaron la mayor parte de las modificaciones requeridas. Sin embargo, hubo un imprevisto en el funcionamiento del código y se requirió de una reunión adicional. En ese momento se manifestó el riesgo 05, al fallar el servidor de correo. Este riesgo ocasionó que no se pudieran realizar todas las modificaciones durante este *sprint*, pero con el tiempo que no se pudo trabajar en Mars Miners, se pasó a diseñar el juego original. Por lo tanto, el éxito de este *sprint* se considera **neutral**.

Resumen Sprint 2	
Fecha Inicio	06/05/2024
Fecha Fin	14/05/2024
Diseño	5 horas
Código	26 horas
Búsqueda e Investigación	6 horas
Cambios en Requisitos/Solución de errores	5 horas
TOTAL	42 horas

Tabla 3.2: Resumen Sprint 2

3.3. Sprint 3

Durante la reunión del *sprint* se acordó cómo debía funcionar el juego y el primer nivel. Los siguientes niveles se comentaron por encima, dando ideas de cómo podría aumentar la dificultad, pero no quedó nada definido. Tras esta reunión, el primer trabajo realizado es el de resolver los problemas del anterior *sprint*. Se averigua la causa del fallo en las comunicaciones

y ya se puede evitar ese riesgo. Una vez terminadas las modificaciones de Mars Miners, se procede a desarollar el GDD de Puntero Aventura y a desarrollar los requisitos de dicho juego. El primer nivel de Puntero Aventura sienta las bases para los siguientes, y es durante este *sprint* en el que se crea la base del juego: El menú inicial y la base para los niveles. La base para los niveles consiste en crear al jugador, los límites del mapa, los elemntos del juego a recoger (celdas de memoria) y la lista con esos objetivos. La mayoría de estas ideas fueron aceptadas o requirieron de cambios mínimos, por lo que este *sprint* se considera un **éxito total**.

Resumen Sprint 3	
Fecha Inicio	14/05/2024
Fecha Fin	31/05/2024
Diseño	18 horas
Código	46 horas
Búsqueda e Investigación	14 horas
Cambios en Requisitos/Solución de errores	8 horas
TOTAL	86 horas

Tabla 3.3: Resumen Sprint 3

3.4. Sprint 4

En la reunión de este *sprint* se acordó qué dificultad y elementos añadidos debería tener el segundo nivel, así cómo los cambios que debían realizarse sobre el trabajo del *sprint* anterior. Las modificaciones a realizar sobre el trabajo anterior son simples: cambiar el diseño de las celdas de memoria y hacer la lista de objetivos más visible. El segundo nivel resulta más sencillo de hacer, pues el primero había sentado la mayoría de bases, y la creación de este consistía en ampliar y añadir funcionalidad sobre lo ya hecho. La facilidad de adición de dificultad en este *sprint* implica más sencillez técnica respecto al anterior, y se considera este como un **éxito**.

Resumen Sprint 4	
Fecha Inicio	31/05/2024
Fecha Fin	07/06/2024
Diseño	6 horas
Código	22 horas
Búsqueda e Investigación	4 horas
Cambios en Requisitos/Solución de errores	8 horas
TOTAL	40 horas

Tabla 3.4: Resumen Sprint 4

3.5. Sprint 5

En la reunión de este *sprint* se acordó qué dificultad y elementos añadidos debería tener el tercer nivel. Además, surgieron otros cambios. La lista de objetivos debía desglosarse en 2 listas más simples: Una haciendo las veces de mapa de memoria y otra solo con los objetivos (los identificadores de las variables y lo que hay recoger de ellas) Dada la rapidez con la que se ha podido terminar este *sprint*, y la falta de cambios sugeridos en lo entregado, se considera un **éxito total**.

Resumen Sprint 5	
Fecha Inicio	07/06/2024
Fecha Fin	11/06/2024
Diseño	6 horas
Código	11 horas
Búsqueda e Investigación	3 horas
Cambios en Requisitos/Solución de errores	7 horas
TOTAL	27 horas

Tabla 3.5: Resumen Sprint 5

3.6. Sprint 6

En este *sprint* se define y realiza el último nivel del juego. Este nivel tiene una peculiaridad pues añade una nueva mecánica: La asignación de valores a otras posiciones de memoria. Este nivel, además, se considera como un nivel de desafío y aumenta la dificultad del juego. En este *sprint* surgen varios errores que dificultan el avance, pero por suerte, se pudo compensar al no necesitar hacer arreglos sobre los *sprints* anteriores. En cuanto al éxito de este *sprint*, la expectativa de rapidez que habían generado los 2 *sprints* anteriores, las modificaciones requeridas y el fallo en la planificación inicial que hará necesario otro *sprint* suena desalentador. Sin embargo, se dejó tiempo de margen entre las distintas actividades, por lo que aún queda tiempo suficiente. Por tanto, este *sprint* se considera **neutral**

Resumen Sprint 6	
Fecha Inicio	11/06/2024
Fecha Fin	17/06/2024
Diseño	5 horas
Código	26 horas
Búsqueda e Investigación	3 horas
Cambios en Requisitos/Solución de errores	0 horas
TOTAL	34 horas

Tabla 3.6: Resumen Sprint 6

3.7. Sprint 7

En este *sprint* se realizan los cambios que no fueron aprobados sobre el último nivel, la terminación de los menús y mecánicas de puntuación, y la selección del aspecto visual del juego. Los cambios consisten en pulir la generación de objetivos, que no sea tan aleatoria, ya que es un nivel de desafío. Las mecánicas de puntuación se refieren cómo consigue el jugador los puntos y cuántas oportunidades tiene antes de perder el nivel. Ya que se consigue terminar el trabajo a tiempo, este *sprint* se considera **exitoso**.

Resumen Sprint 7	
Fecha Inicio	17/06/2024
Fecha Fin	01/07/2024
Diseño	18 horas
Código	15 horas
Búsqueda e Investigación	10 horas
Cambios en Requisitos/Solución de errores	5 horas
TOTAL	48 horas

Tabla 3.7: Resumen Sprint 7

Capítulo 4

Documento de Diseño de Juego (GDD)

4.1. Introducción

La aplicación que se desarrollará en este proyecto es un juego didáctico cuya finalidad es que el jugador aprenda sobre los punteros y las operaciones que pueden realizar en el lenguaje de programación C.

El objetivo del juego es que el jugador recoja los datos que se le piden. Estos datos se encuentran en una estructura que emula la memoria de un programa, y el jugador puede ver en todo momento cuáles son los datos que se le piden.

4.2. Audiencia y Plataforma

Este juego forma parte del la iniciativa GREIDI descrita con anterioridad. Por lo tanto, los destinatarios de este juego son alumnos que cursen algún grado o máster universitario que esté abarcado por el grupo.

La plataforma de este grupo es una página web, por lo que este juego podrá ser jugado desde un navegador.

4.3. Mecánicas y controles

La mecánica principal de este juego es la de recoger los datos que se piden en los objetivos. Para ello, al jugador se le proporcionarán 2 estructuras: Un mapa de la memoria y una lista

de objetivos. En el mapa de memoria (Fig. 4.1) se puede ver cómo cada entrada referencia un identificador de variable y la posición de memoria en la que se encuentra dicha variable.

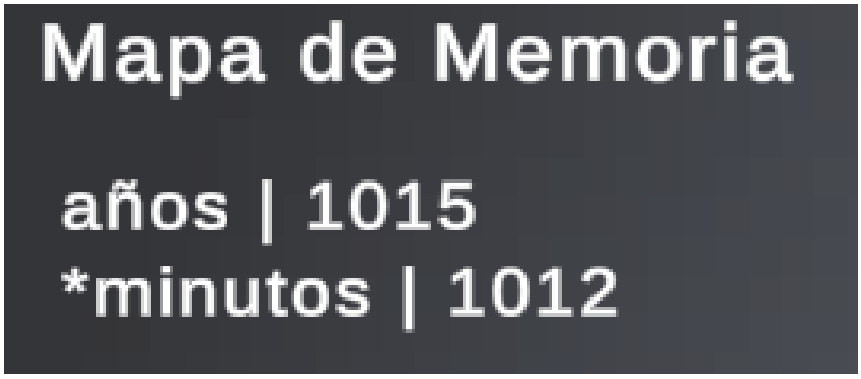

Figura 4.1: Entrada del mapa de la memoria

Por otro lado, en la lista de objetivos (Fig. 4.2) se muestra el identificador de la variable y un operador del lenguaje de programación C. Este operador da una pista al jugador sobre qué es lo que se le pide exactamente acerca de dicha variable.

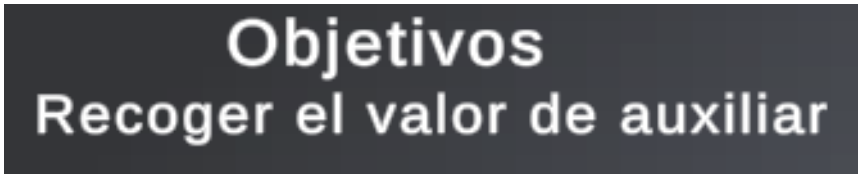

Figura 4.2: Entrada de la lista de objetivos

Para poder recoger estos datos, el jugador podrá moverse por el mapa con las flechas de dirección o mediante las teclas WASD. Cuando se acerque a una celda de memoria, está se iluminará para indicar al jugador que puede interactuar con ella. (Fig. 4.3)

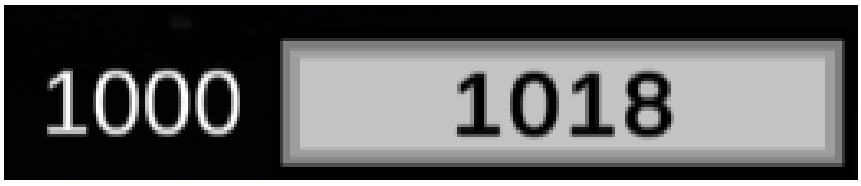

Figura 4.3: Celda de memoria con dirección y valor

Por último, las celdas de memoria se generarán aleatoriamente en cada nivel. Cada celda se compone de 2 partes: su dirección de memoria y su valor. Ya que el juego intenta emular

la memoria de un programa, las direcciones de memoria siempre serán las mismas y tendrán valores contiguos. Por otra parte, los valores se elegirán aleatoriamente. Estos valores pueden o no coincidir con las direcciones de memoria de otras celdas, con la finalidad de confundir al jugador. De las celdas de memoria se puede recoger tanto el valor que almacena como su dirección de memoria. La razón de esto se explicará más tarde.

El siguiente es un listado de las acciones que el jugador puede realizar:

- Moverse por el escenario, con las flechas de dirección o las teclas WASD.
- Procesar un objetivo: esto es lo que se entiende al recoger un valor. El jugador recoge el valor con el que está interactuando mediante la tecla "espacio" y se procesa como un objetivo recogido
- Guardar un valor: En el nivel de desafío, usando la tecla "j" el jugador podrá guardarse un valor para usarlo más tarde
- Usar un valor: Tras haber guardado un valor, podrá usarlo con la tecla "k" en otra posición de memoria, que contará como procesar un objetivo.

El jugador tiene una cantidad limitada de intentos para conseguir todos los objetvos, pero sí tiene un tiempo ilimitado, bajo el riesgo de que pierda puntos. Cuando el jugador empieza un nivel, lo empieza con todos los puntos. Cada cierto tiempo, perderá puntos de manera visible, para incentivar una velocidad y entendimiento de los conceptos. Sin embargo, también dispondrá de un número de errores que tiene permitido cometer, para desincentivar el procesar todas las celdas como objetivos sin pensar. Cada vez que el jugador procese un objetivo que no sea el que se le pide, perderá una vida, y con ello puntos. Si el jugador pierde un cierto número de vidas, perderá ese nivel, teniendo que intentarlo de nuevo.

4.4. Niveles

El juego dispondrá de 4 niveles, cada uno aumentando la dificultad en cuanto a conceptos de manera incremental, y siendo el último un nivel de desafío. Durante el primer nivel, el jugador tendrá acceso al tutorial donde se le explicarán las normas y las acciones que puede tomar dentro del juego.

En todos los niveles los objetivos, las celdas de memoria y el mapa de memoria se generan aleatoriamente, siempre que haya al menos un objetivo que corresponda a la dificultad del nivel. Los cambios incrementales en cada nivel son los objetivos que se piden al jugador, salvo para el nivel 4, del cuál se hablará más tarde. En cada nivel hay 4 objetivos a recoger, y el jugador dispondrá de 3 vidas

4.4.1. Nivel 1

Este es el nivel más sencillo, que sirve de introducción a los punteros. Las variables que se encuentran pueden ser variables estáticas o punteros.

Los objetivos que se muestran en este nivel siempre coinciden con el identificador de la variable. Si es una variable estática, se pide el valor que almacena. En cambio, si es un puntero, se pide el valor que almacena **la variable a la que está apuntando**.

El temporizador de este nivel tiene una duración de 2 minutos, y los puntos se van decrementando cada 15 segundos.

4.4.2. Nivel 2

Este segundo nivel introduce el operador conocido como "Ampersand" &. Este operador, cuando se usa en una variable, devuelve su posición de memoria.

En la lista de objetivos, además de incluir todo lo del nivel 1, puede aparecer un "&" delante del identificador de la variable. Cuando esto pase, se pide al jugador que interactúe con la **dirección de memoria de la variable**. Si es una variable estática, se pide su dirección, y si es un puntero, reemplaza al asterisco característico de los punteros por un ampersand y **también se pide su dirección**, no la dirección a la que apunta.

El temporizador de este nivel tiene una duración de 2 minutos y 30 segundos, y los puntos se van decrementando cada 15 segundos.

4.4.3. Nivel 3

En el tercer nivel se ahonda sobre el operador "Ampersand". Ahora puede aparecer en combinación con el asterisco de los punteros.

La lista de objetivos amplía lo que tenía el nivel 2 con la combinación de ampersand y asterisco. Este tipo de objetivo solo puede aparecer en las variables de tipo puntero, y cuando lo hace, se está pidiendo al usuario **el valor que almacena el puntero**, es decir, la dirección de memoria a la que apunta.

El temporizador de este nivel tiene una duración de 2 minutos y 45 segundos, y los puntos se van decrementando cada 15 segundos.

4.4.4. Nivel 4

Este nivel cambia un poco respecto a los anteriores descritos. En los anteriores niveles, simplemente se debía recoger un valor. En este nivel, lo que se pide es hacer una asignación. En este nivel, el jugador deberá recoger un valor y depositarlo en la dirección de memoria pedida.

Los objetivos de este nivel son pseudoaleatorios. Las variables elegidas siempre son aleatorias, pero con ciertas condiciones (Fig. 4.4):

- En el primer objetivo, el jugador deberá asignar a una variable estática el valor al que apunta un puntero. Es decir, el lado derecho de la asignación es un puntero, y el izquierdo una variable estática
- En el segundo objetivo, el jugador deberá asignar el valor de una variable estática a la dirección a la que apunta un puntero. Es decir, el lado derecho de la asignación es una variable estática, y el izquierdo un puntero
- En el tercer objetivo, ambos lados de la asignación son punteros
- En el cuarto objetivo, el lado derecho de la asignación contiene un ampersand. Se deberá asignar una posición de memoria a una variable estática.

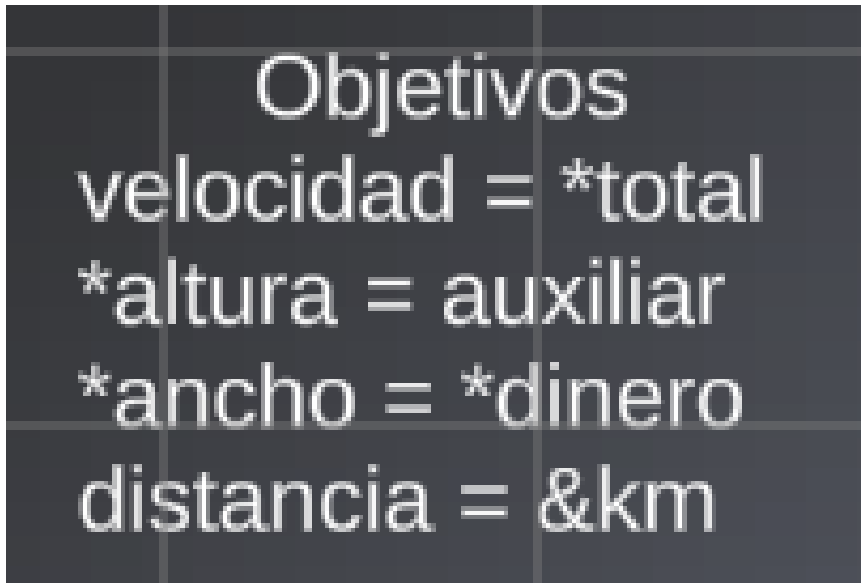

Figura 4.4: Lista de objetivos del desafío

Se ha decidido no añadir el operador conjunto ampersand y asterisco del nivel 3 en este nivel, pues se considera que la dificultad es suficiente. Este nivel tiene un temporizador de 3 minutos, y la puntuación decrementa cada 20 segundos.

4.5. Historia

El juego no tiene una historia concreta, pero está ambientado en un mundo futurista, en el espacio. Esto es así porque temas como la memoria de un programa y los punteros son de lo más complejo en el lenguaje de programación en C, y se cree que una ambientación futurista puede ayudar mejor al jugador a conectar con el mundo de los punteros. El personaje principal es una nave espacial, la lista de objetivos y el mapa de memoria son monitores futuristas, y de fondo se puede ver la luna y el espacio.

4.6. Recursos

Para este juego se han empleado diversos recursos: sonidos para los menús, imágnes para el fondo y las pantallas, y *sprites* para el personaje principal y las celdas de memoria. Todo lo que no se ha creado a mano tiene una licencia de uso gratuito.

El personaje principal se ha sacado de la Unity Asset Store [12]

Figura 4.5: Representación del personaje

Las pantallas para los objetivos y el mapa de memoria se han sacado de una página de uso libre [13]

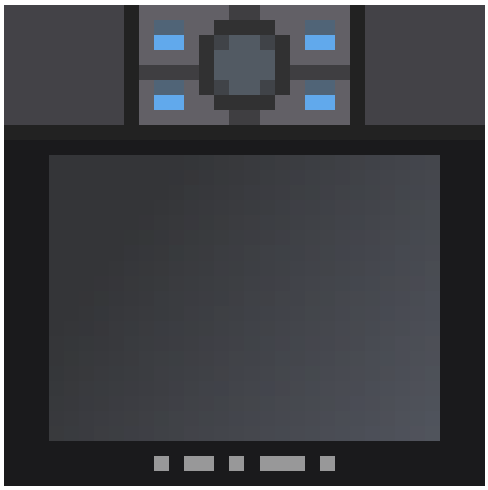

Figura 4.6: Fondo objetivos

Figura 4.7: Fondo mapa de memoria

El fondo de espacio se ha sacado de una plataforma online que ofrece recursos tanto gratuitos como de pago [14]

Figura 4.8: Fondo del juego

La imagen de batería para las vidas se ha obtenido de una página web que ofrece recursos tanto gratuitos como de pago [10]

Figura 4.9: Vidas del jugador

Las celdas de memoria se han creado a mano para este proyecto usando Paint. [15]

Figura 4.10: Celda de memoria

Los efectos de sonido se han sacado de la página Freesound [16]

Capítulo 5

Requisitos

Esta sección se dedica a definir y describir las necesidades fundamentales que debe cumplir el videojuego. Aquí se especifican tanto los requisitos funcionales, que detallan las funcionalidades y comportamientos esperados del sistema, como los requisitos no funcionales, que establecen las condiciones y restricciones sobre cómo debe operar el sistema. Esta sección es esencial para asegurar que el desarrollo del videojuego esté alineado con las expectativas y necesidades de los usuarios, proporcionando una base sólida para el diseño y la implementación del proyecto. Además, se mostrarán los casos de uso, los cuales ilustran los distintos escenarios en los que los usuarios interactuarán con el sistema.

5.1. Requisitos

5.1.1. Requisitos funcionales

La tabla 5.1 recoge los requisitos funcionales del videojuego. En esta tabla se incluyen el identificador del requisito, su nombre y una descripción.

ID	Nombre	Descripción
RF-01	Interfaz de inicio	El sistema deberá tener una pantalla de inicio con los diferentes niveles y un botón para salir del juego
RF-02	Jugar nivel	El sistema deberá permitir seleccionar un nivel y jugarlo
RF-03	HUD (Heads-Up Display)	Durante el juego, debe mostrar información relevante como puntuación, tiempo restante y vidas.
RF-04	Interactuar con el entorno	El sistema deberá permitir al jugador interactuar con las celdas de memoria
RF-05	Conseguir objetivo	El sistema deberá indicar al jugador cuándo ha conseguido un objetivo y cuándo ha fallado
RF-06	Ganar nivel	El sistema deberá mostrar una pantalla de fin de nivel cuando el jugador gane el nivel
RF-07	Perder nivel	El sistema deberá mostrar una pantalla de fin de nivel cuando el jugador pierda el nivel
RF-08	Control de Personaje	El jugador debe poder mover el personaje principal usando las teclas de flecha o WASD.
RF-09	Niveles y Progresión	El juego debe tener múltiples niveles, cada uno con una complejidad creciente en el uso de punteros.
RF-10	Sistema de Puntuación	Debe haber un sistema de puntuación basado en la rapidez y precisión con la que se completan los niveles.

Tabla 5.1: Requisitos funcionales

5.1.2. Requisitos no funcionales

La tabla 5.2 recoge los requisitos no funcionales del videojuego. En esta tabla se incluyen el identificador del requisito, su nombre y una descripción.

ID	Nombre	Descripción
RNF-01	Motor de juego	El videojuego deberá ser desarrollado usando **Unity**.
RNF-02	Compatibilidad WebGL	El videojuego deberá desarrollarse de forma que pueda ejecutarse en plataformas compatibles con WebGL.
RNF-03	Interfaz Intuitiva	La interfaz debe ser intuitiva y fácil de navegar para usuarios sin experiencia en programación.
RNF-04	Tutorial	Debe incluir un tutorial que explique los controles del juego y cómo jugar.

Tabla 5.2: Requisitos no funcionales

5.2. Casos de uso

En esta sección se tratarán los casos de uso para los distintos escenarios que tiene el juego: El menú de inicio, dentro de un nivel y la pantalla de final de nivel.

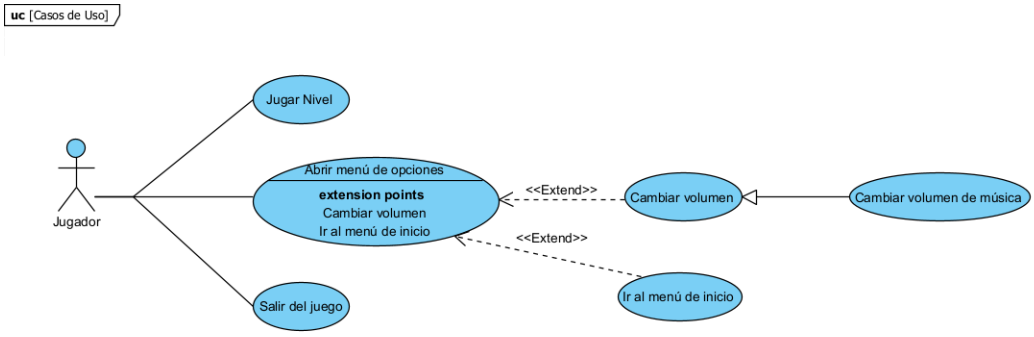

Figura 5.1: Diagrama de casos de uso del menú de inicio

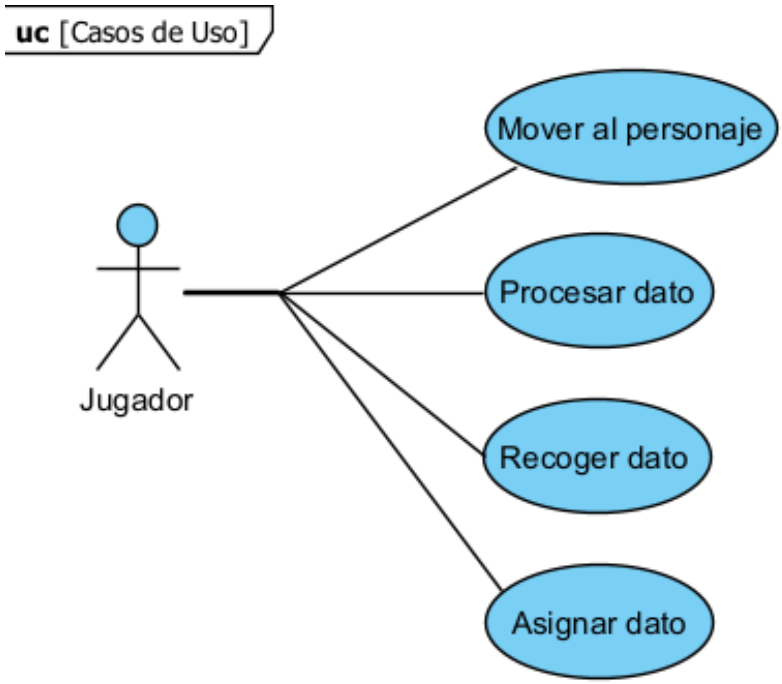

Figura 5.2: Diagrama de casos de uso de un nivel

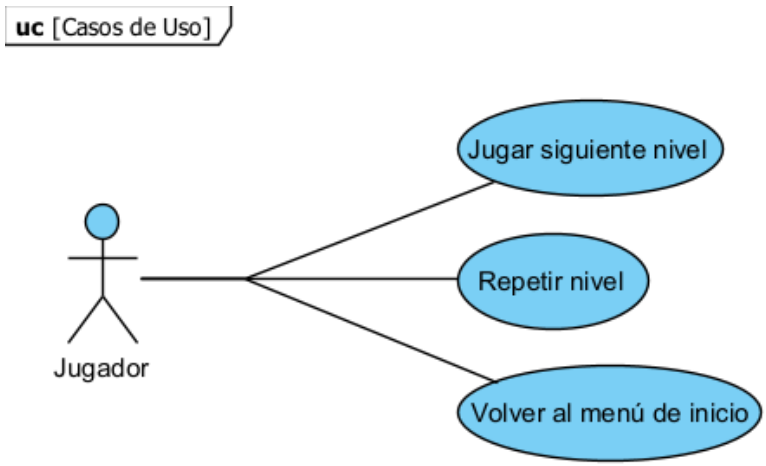

Figura 5.3: Diagrama de casos de uso de la pantalla de fin de nivel

Capítulo 6

Análisis

En esta sección se identificarán y describirán las principales entidades y relaciones que conforman el mundo del videojuego. Este modelo proporciona una representación conceptual del sistema, facilitando la comprensión de sus componentes fundamentales y cómo interactúan entre sí. Al definir claramente las entidades, sus atributos y las relaciones entre ellas, el modelo de dominio establece una base sólida para el diseño detallado y la implementación del juego. Además, ayuda a asegurar que todos los requisitos del sistema se comprendan y se gestionen adecuadamente a lo largo del ciclo de desarrollo.

6.1. Modelo de Dominio

En la figura 6.1 se puede apreciar el modelo de dominio de la aplicación. Para una mejor comprensión del problema, se proceden a dar más detalles de las clases.

- **LevelLoader** es la clase encargada de crear los niveles. Esta clase emplea el patrón Singleton, y solo habrá un objeto de esta clase.
- **AbsLevel** es la clase base para todos los niveles. Gestiona todo lo relacionado con los niveles, desde la creación y procesamiento de los objetivos hasta la creación del personaje.
- **Easy0, Medium0, Hard0 y Challenge0** son las distintas especializaciones de cada nivel de dificultad. Heredan de *AbsLevel* y modifican lo necesario para jugar el nivel específico.
- **ShipControl** es la clase encargada de gestionar el movimiento del personaje principal.
- **PlayerLives** se encarga de gestionar las vidas del jugador
- **MemoryCell** representa cada celda de memoria.

- **ChangeColor** se encarga de mostrar feedback visual al jugador cuando este puede interactuar con una celda de memoria.
- **Objective** representa los objetivos que el jugador debe recoger.
- **MemoryMapEntry** representa cada entrada del mapa de memoria.
- **CollectValue** se encarga de que el jugador pueda recoger y dejar valores en el nivel de desafío.

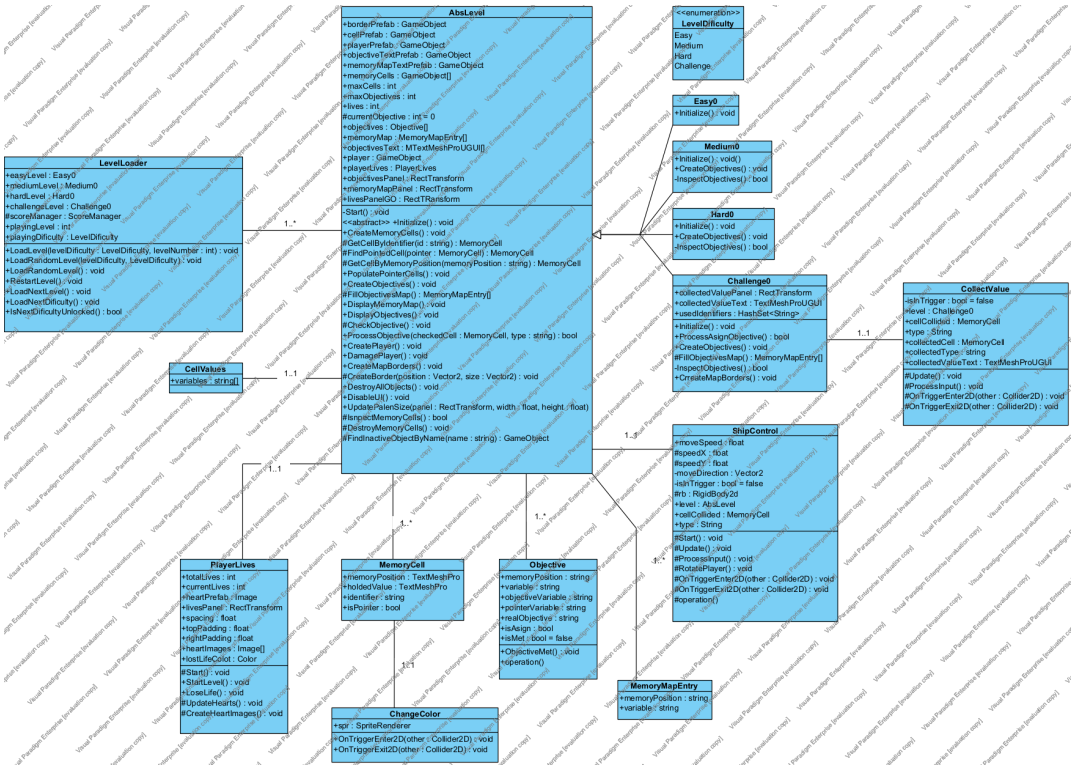

Figura 6.1: Modelo de dominio

Capítulo 7

Pruebas

El capítulo de pruebas se centra en garantizar la calidad y la funcionalidad del videojuego. A través de un enfoque sistemático y estructurado, se validarán todos los aspectos del juego para asegurar que cumple con los requisitos especificados y ofrece una experiencia de usuario óptima. En cada tabla se detallarán el identificador de la prueba, su nombre y descripción. Las pruebas se dividirán en tres secciones principales: Pruebas Unitarias, Pruebas de Integración y Pruebas de Aceptación.

7.1. Pruebas Unitarias

En esta sección, se validarán individualmente los componentes más pequeños del código, como funciones y métodos, para asegurar que cada unidad funciona correctamente de manera aislada. En la tabla 7.1 se detallan todas las pruebas unitarias realizadas.

ID	Nombre	Descripción
PU-01	Límites del mapa	El jugador no podrá abandonar el mapa.
PU-02	Generar celdas de memoria	Las celdas de memoria se generarán con un identificador y dirección de memoria distinta cada una.
PU-03	Generar objetivos	Los objetivos deberán generarse de acuerdo a las necesidades de cada nivel y no podrán repetirse.
PU-04	Generar mapa de memoria	El mapa de memoria deberá contener las direcciones de todos los objetivos, y no podrá repetirse ninguna entrada.
PU-05	Perder nivel	Si el jugador se queda sin vidas, perderá el nivel.
PU-06	Ganar nivel	Si el jugador obtiene todos los objetivos, ganará el nivel.

Tabla 7.1: Pruebas unitarias

7.2. Pruebas de Integración

Aquí se verificará que las diferentes unidades de código funcionen correctamente cuando se combinan entre sí. En la tabla 7.2 se detallan todas las pruebas de integración realizadas.

ID	Nombre	Descripción
PI-01	Vidas del jugador	Las vidas del jugador se reiniciarán en cada nivel.
PI-02	Pantalla de Inicio	Cuando el jugador termine un nivel, de manera exitosa o no, podrá volver a la pantalla de inicio.
PI-03	Interactuar con celda correcta	El jugador podrá interactuar con las celdas de memoria, y si interactúa con la celda que corresponde al objetivo actual, el objetivo se marcará como conseguido.
PI-04	Interactuar con celda incorrecta	El jugador podrá interactuar con las celdas de memoria, y si interactúa con una celda que no corresponde al objetivo actual, perderá una vida.
PI-05	Recoger valor	En el nivel de desafío, el jugador podrá interactuar con los valores y recogerlos. Cuando interactúe con un valor, recibirá feedback por pantalla.
PI-06	Dejar valor	En el nivel de desafío, cuando un jugador haya recogido un valor, podrá dejarlo en otra celda de memoria. Si el valor y la celda de memoria coinciden con el objetivo, se marcará como correcto. Si no, el jugador perderá una vida.

Tabla 7.2: Pruebas de Integración

7.3. Pruebas de Aceptación

Esta sección se enfocará en validar que el sistema completo cumple con los requisitos funcionales y no funcionales especificados. Involucra la evaluación del juego en un entorno realista para garantizar que los usuarios finales obtengan una experiencia satisfactoria. En la tabla 7.3 se detallan todas las pruebas de integración realizadas.

ID	Nombre	Descripción
PI-01	Objetivos de cada nivel	Para cada nivel, al menos hay un objetivo de su dificultad destinada
PI-02	Estética	El juego deberá tener una misma estética

Tabla 7.3: Pruebas de Aceptación

Capítulo 8

Mantenimiento Mars Miners

Este capítulo estará dedicado al segundo objetivo de este proyecto: modificar el juego ya existente en la plataforma GREIDI, *Mars Miners*. *Mars Miners* es un juego que pretende enseñar a sus jugadores sobre las estructuras de control, tales como *if*, *while* y *for*. Para esto, el jugador controla a un robot que sigue unas direcciones proporcionadas por una pantalla que muestra líneas de código. El objetivo del juego es que el robot consiga minerales y que no choque con las diversas rocas que se encuentran por el escenario, las cuales podrá mover.

8.1. Modificaciones requeridas

Las modificaciones que se pidieron realizar en este juego eran 3:

- **Modificar el código mostrado de Java a C.** Este juego muestra líneas de código por pantalla que el usuario debe interpretar. Dichas líneas están escritas en Java. Para que se apadte a estudiantes de C, se pedía traducir dichas líneas.
- **Añadir dificultad.** En muchos niveles, la única condición para pasarlos exitosamente es quitar las rocas de en medio, sin pensar en qué hace el código.
- **Asegurar la comprensión del jugador**

8.2. Modificaciones realizadas

Para conseguir el primer requerimiento, las acciones a tomar fueron inmediatas: cambiar el texto de un lenguaje de programación por otro.

Para añadir dificultad, se modificaron las condiciones de las estructuras de control para que incluyesen a las rocas que se podían mover. Ahora, para pasar el nivel no solo hace falta

eliminar las rocas del camino, sino que si no se colocan de cierta manera que ayuden al robot, no se pasará el nivel.

Para asegurar la comprensión del jugador, se creó un nuevo mineral con distintas condiciones de recogida. Los minerales ya existentes eran estáticos. El robot pasaba por ellos, y al recogerlo, obtenía puntuación. En ciertos niveles se han añadido minerales de otros colores con la capacidad de ser movidos y que deberán recogerse en el último movimiento del robot. Si no se recogen en el último movimiento, el nivel se dará por perdido. Esto es de especial utilidad a los jugadores en los últimos niveles, en los que el jugador deberá entender qué hace cada línea de código y cuándo se cumplen las condiciones para salir de la estructura de control.

Capítulo 9

Tecnologías utilizadas

9.1. Unity

Unity [7] es un motor de videojuegos muy utilizado, que requiere muy poca experiencia para aprender a usar y que cuenta con gran potencia, versatilidad y apoyo. Alguna de las herramientas y funcionalidades que posee son:

- **Motor gráfico multiplataforma:** Unity permite crear juegos y aplicaciones capaces de ejecutarse en múltiples plataformas, como pueden ser PC, consolas y dispositivos móviles.
- **Gráficos y renderización:** Unity cuenta con un motor gráfico que soporta juegos tanto en 2D como en 3D, y usa tecnologías de renderizado basadas en físicas
- **Comunidad:** Este punto es con diferencia el mayor atractivo de Unity. Cuenta con un sinfín de tutoriales y de creadores de contenido a su lado, así cómo una *Asset Store* [8] en donde los creadores pueden vender todo tipo de cosas, como pueden ser imágenes para los personajes del juego, como menús o incluso *scripts*, que son trozos de código.

Figura 9.1: Logo de Unity

9.2. C#

C-Sharp [6] es un lenguaje de programación, y es el lenguaje que soporta Unity por defecto. El código escrito se conoce como *script*, y se añade a los objetos dentro del juego para controlar su comportamiento y las acciones que puede tomar.

Figura 9.2: Logo de C-Sharp

9.3. Blender

Blender [9] es una herramienta de software libre y código abierto cuya característica principal es el modelado 3D.

Esta es una herramienta bastante potente y de amplio uso por la comunidad para juegos 3D, ya que a parte del modelado, ofrece la oportunidad de animar, añadir esqueletos a los modelos, renderizarlos y añadir y crear materiales, texturas y sombras para los modelos.

Figura 9.3: Logo de Blender

9.4. Overleaf y LaTeX

LaTeX y Overleaf [6] son herramientas relacionadas con la creación y edición de documentos utilizando el sistema de composición tipográfica LaTeX.

LaTeX es un sistema de composición de textos que se centra en la estructura y el contenido del documento, permitiendo a los usuarios concentrarse en el contenido mientras LaTeX se encarga del diseño y la maquetación.

Por otro lado, Overleaf es una plataforma en línea que permite a los usuarios escribir, editar y colaborar en documentos LaTeX sin necesidad de instalar software localmente. Es una interfaz basada en la web que facilita la creación y edición de documentos LaTeX en tiempo real, y proporciona herramientas para la colaboración y el trabajo en equipo.

Figura 9.4: Logo de Overleaf

9.5. Visual Paradigm

Visual Paradigm [11] es una completa herramienta de modelado y gestión de proyectos que facilita el diseño y desarrollo de software mediante el uso de diversos diagramas y metodologías. Esta plataforma se destaca por su capacidad de soportar una amplia gama de lenguajes de modelado como UML (Unified Modeling Language), BPMN (Business Process Model and Notation), y ERD (Entity-Relationship Diagram), entre otros.

Esta herramienta se ha utilizado para crear el modelo de dominio y los diagramas de caso de uso.

Figura 9.5: Logo de Visual Paradigm

9.6. Paint

Microsoft Paint, comúnmente conocido simplemente como Paint, es un programa básico de edición gráfica incluido en todas las versiones de Microsoft Windows.

Este programa se ha utilizado para crear un *asset* del juego.

Figura 9.6: Logo de Paint

Capítulo 10

Conclusiones

Tras la finalización de este proyecto, se puede concluir que ha sido completado de manera exitosa, cumpliendo los requisitos propuestos. Se ha sido capaz de crear un videojuego que enseñe una parte tan relevante de la programación en C como son los punteros.

10.1. Conocimientos adquiridos

Entre otros, los conocimientos adquiridos a destacar son:

- Desarrollo de un videojuego mediante la división de trabajo en *sprints* dentro de un marco de trabajo ágil.
- Se han afianzado conocimientos sobre el motor de videojuegos *Unity* y el lenguaje de programación *C#*, que sin duda serán útiles para futuros proyectos.
- Se ha aprendido a realizar pruebas de aceptación en las reuniones de planificación de los *sprints*.
- Se ha adquirido una gestión del tiempo necesaria para finalizar los *sprints* a tiempo y planificar la duración de cada uno con antemano.

10.2. Trabajos futuros

En esta sección se hará una lista de posibles mejoras y adiciones que se podrían añadir al videojuego:

- Mejora de la inmersión del jugador, añadiendo más efectos de sonido y visuales en el fondo.

- Añadido de niveles con aumento de dificultad. Si bien el juego cubre las operaciones básicas con punteros, aún quedan conceptos que pueden enseñarse, como la suma de punteros o los punteros a punteros
- Dar más libertad al jugador, permitiéndolo cambiar los controles
- Expansión a diversas plataformas, como pueden ser los dispositivos móviles.

-

Bibliografía

[1] Gamificación https://www.questionpro.com/blog/es/gamificacion-en-el-aula/

[2] Manifiesto ágil https://agilemanifesto.org/iso/es/manifesto.html

[3] Sueldo https://www.tokioschool.com/formaciones/cursos-videojuegos/diseno/sueldo/

[4] Riesgos https://asana.com/es/resources/project-risk-management-process

[5] Overleaf www.overleaf.com

[6] C# https://dotnet.microsoft.com/es-es/languages/csharp

[7] Unity https://unity.com/es

[8] Unity Asset Store https://assetstore.unity.com/

[9] Blender https://www.blender.org/

[10] Vidas Jugador www.vecteezy.com

[11] Visual Paradigm https://www.visual-paradigm.com/

[12] Personaje Puntero Aventura https://assetstore.unity.com/packages/2d/characters/2d-pixel-spaceship-two-small-ships-131545

[13] Panel Memoria y Objetivos https://opengameart.org/content/screen-32x26-and-tablet-32x32

[14] Fondo Puntero Aventura https://www.freepik.com/free-photos-vectors/space-moon-background

[15] Paint https://www.microsoft.com/es-es/windows/paint

[16] FreeSound https://freesound.org/