Plataforma para recreación de estrategia basada en aprendizaje reforzado

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Julio de 2023

Autor

Erick José Mercado Hernández

Tutor/a

Vidal Moreno Rodilla

D. Vidal Moreno Rodilla, profesor del Departamento de Departamento de Informática y Automática de la Universidad de Salamanca,

Hace constar:

Que el trabajo titulado “Plataforma para recreación de estrategia basada en aprendizaje reforzado” ha sido realizado por D. Erick José Mercado Hernández, con DNI 43840208T y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a martes, 4 de julio de 2023

MORENO RODILLA VIDAL - 07870365H

Firmado digitalmente por MORENO RODILLA VIDAL - 07870365H
Fecha: 2023.07.04 10:24:08 +02'00'

Dr. Vidal Moreno Rodilla

Lista de cambios

Número	Fecha	Versión	Autor
0	02-02-2023	Versión 0.1 (Creación del documento)	Erick José Mercado Hernández
1	01-07-2023	Versión 1.0 (Desarrollo del documento)	Erick José Mercado Hernández

Resumen

Este proyecto busca revolucionar el campo de los juegos tácticos basados en turnos (TBRPG). Elaborado en Unity, este proyecto se centra en la aplicación de técnicas de aprendizaje reforzado y algoritmos de inteligencia artificial sofisticados para mejorar la jugabilidad y la experiencia del jugador.

En el núcleo se encuentran varios algoritmos y técnicas sofisticadas de IA, cada uno de los cuales contribuye a la creación de una experiencia de juego profundamente rica y variada. Al tratarse de juegos con adversario es necesario dotar al mismo, visto como agente inteligente de las mejores capacidades proactivas que permitan esta experiencia de juego.

En primer lugar, al tratarse de "enemigos/contrarios" con movilidad es necesario que éstos dispongan de capacidades navegacionales en el escenario, para lo que es necesaria la capacidad de planificar movimientos, para lo que, en este caso se van a proponer estrategias como el algoritmo A* que permiten tener esta capacidad con soluciones de calidad. Es una técnica de búsqueda informada eficiente y versátil, utilizada para encontrar el camino más corto en gráficos complejos, como los mapas de juegos. Adicionalmente, el cálculo de la distancia de Chevysev, una medida heurística de la distancia entre dos puntos en una cuadricula se utilizará para informar las decisiones de las unidades controladas por IA, permitiéndoles navegar de manera eficiente en el entorno del juego.

Por otra parte, las capacidades estratégicas requieren de disponer de capacidades de toma de decisión autónomas, a nivel de planes de acción (ataque-defensa en nuestro caso). Para ello, y con objeto de que el adversario tenga un comportamiento adecuado e inteligente, se van a explorar las técnicas de aprendizaje reforzado. Por tanto, una de las características más destacadas de la plataforma será su capacidad para recrear situaciones de estrategia en un entorno de juego. Esto permite a los usuarios, desde los entusiastas de los juegos hasta los académicos, estudiar y comprender el aprendizaje reforzado en un entorno interactivo y lúdico. A través de la experimentación directa y la observación, los

usuarios pueden explorar cómo las técnicas de aprendizaje reforzado y la IA pueden combinarse para resolver problemas de estrategia de manera efectiva y eficiente.

Finalmente, y, con objeto de tener capacidades de comportamiento en tiempo real, y teniendo en cuenta que los escenarios reales de juegos generan complejos árboles de búsqueda será necesario utilizar soluciones que permitan alcanzar soluciones óptimas.

Así, la plataforma incorporará el algoritmo de Montecarlo, una herramienta poderosa para la toma de decisiones probabilísticas y la modelación de incertidumbre en el contexto complejos como es en el de los juegos de estrategia.

Estas herramientas de IA no solo mejoran la jugabilidad al generar oponentes controlados por computadora más inteligentes y desafiantes, sino que también mejoran la experiencia general del jugador al permitir una interacción más fluida y rica con el entorno del juego.

Palabras clave: Unity, aprendizaje reforzado, algoritmo de búsqueda A*, algoritmo de Montecarlo, distancia de Chevyshev, inteligencia artificial.

Abstract

This project seeks to innovate the field of turn-based tactical games (TBRPG). Developed in Unity, this project focuses on the application of enhanced learning techniques and sophisticated artificial intelligence algorithms to improve gameplay and player experience.

At the core are several sophisticated AI algorithms and techniques, each contributing to the creation of a deeply rich and varied gameplay experience. As these are adversarial games, it is necessary to equip the adversary, seen as an intelligent agent, with the best proactive capabilities to enable this gaming experience.

First, when dealing with "enemies/counterparts" with mobility, it is necessary that they have navigational capabilities in the scenario, for which the ability to plan movements is necessary, for which, in this case, strategies such as the A* algorithm will be proposed that allow this ability with quality solutions. It is an efficient and versatile informed search technique, used to find the shortest path in complex graphs, such as game maps. Additionally, the calculation of the Chevysev distance, a heuristic measure of the distance between two points on a grid, will be used to inform the decisions of AI-controlled units, allowing them to navigate efficiently in the game environment.

On the other hand, strategic capabilities require autonomous decision-making capabilities, at the level of action plans (attack-defence in our case). To this end, and with the aim of ensuring that the adversary behaves appropriately and intelligently, reinforced learning techniques will be explored. Therefore, one of the most outstanding features of the platform will be its ability to recreate strategy situations in a game environment. This allows users, from gaming enthusiasts to academics, to study and understand reinforced learning in an interactive and playful environment. Through direct experimentation and observation, users can explore how reinforcement learning techniques and AI can be combined to solve strategy problems effectively and efficiently.

Finally, in order to have real-time behavioural capabilities, and taking into account that real game scenarios generate complex search trees, it will be necessary to use solutions that allow optimal solutions to be reached.

Thus, the platform will incorporate the Monte Carlo algorithm, a powerful tool for probabilistic decision making and uncertainty modelling in the complex context of strategy games.

These AI tools not only enhance gameplay by generating smarter and more challenging computer-controlled opponents, but also improve the overall player experience by enabling a smoother and richer interaction with the game environment.

Keywords: Unity, reinforced learning, A* search algorithm, Monte Carlo algorithm, Chevyshev distance, artificial intelligence.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi tutor Vidal, por aceptar mi propuesta, la cual me ha permitido realizar algo que llevaba fraguándose años antes de iniciar esta carrera y cuyo valioso asesoramiento y apoyo constante han sido cruciales en el desarrollo de este proyecto. Su experta orientación ha sido un faro en cada etapa de este desafío.

Mi gratitud también se extiende a los profesores del grado, cuyas enseñanzas me han proporcionado las herramientas y conocimientos necesarios para llevar a cabo este proyecto. Vuestro compromiso con la educación ha sido una fuente de inspiración.

A mi familia, gracias por vuestro inquebrantable apoyo y amor. Vuestra confianza en mí me ha impulsado a seguir adelante incluso en los momentos más difíciles.

Agradezco también a mis amigos, especialmente a aquellos que me introdujeron en el mundo de los juegos de rol. Vuestra pasión por el juego ha sido el aire que ha mantenido viva la llama para llevar a cabo este proyecto.

Gracias a todos por vuestra ayuda y por formar parte de este viaje.

Índice de contenido

Índice de Tablas

Índice de Ilustraciones

1.- Introducción

La industria de los videojuegos ha experimentado una transformación dramática en las últimas décadas, avanzando de simples gráficos en 2D a mundos inmersivos en 3D con personajes autónomos y dinámicas de juego complejas. En esta evolución, los juegos de rol tácticos basados en turnos (TBRPG) han surgido como un campo especialmente interesante y desafiante, combinando la planificación estratégica con la interacción en tiempo real. Los avances en inteligencia artificial (IA) y el aprendizaje reforzado están cambiando esta área, permitiendo experiencias de juego más profundas y enriquecidas. Presentamos aquí un proyecto que se sitúa en la vanguardia de estos emocionantes desarrollos.

Este innovador proyecto se ha desarrollado en Unity (Unity Technologies, s.f.), una de las plataformas más populares y versátiles para el desarrollo de videojuegos. Unity destaca por su capacidad para crear juegos 2D y 3D, y su flexibilidad la hace adecuada para una amplia gama de géneros, incluyendo TBRPG (Smith & Ferns, 2021). Aspiramos a aplicar técnicas de aprendizaje reforzado y algoritmos de IA sofisticados en este entorno para mejorar la jugabilidad y la experiencia del jugador.

El núcleo de este proyecto se basa en una serie de algoritmos avanzados, entre ellos el algoritmo de búsqueda A*, con el cálculo de la distancia de Manhattan o la distancia de Chevyshev. Estos se utilizan para optimizar la toma de decisiones de las unidades controladas por la IA en el juego, lo que resulta en un comportamiento más inteligente y desafiante de los oponentes controlados por el ordenador. La aplicación de estas tecnologías de IA en un entorno de juego realista no solo mejora la experiencia del juego, sino que también contribuye a la investigación en IA.

En lo que respecta al aprendizaje reforzado, este trabajo presenta una oportunidad sin precedentes. Utilizando un entorno de juego como plataforma para estudiar cómo los algoritmos pueden aprender y adaptarse a través de la experiencia, podemos ayudar a los investigadores a entender mejor y a mejorar los métodos de aprendizaje reforzado. Estos hallazgos tienen potencial para trascender el ámbito de los videojuegos, con aplicaciones

potenciales en una amplia gama de industrias, incluyendo robótica, medicina, finanzas y más.

Consideramos los videojuegos como una valiosa plataforma para la investigación científica. Mediante el uso de estos como un laboratorio virtual, podemos explorar complejas interacciones estratégicas y decisiones en un entorno controlado y repetible. Esto no sólo permite refinar nuestros algoritmos de IA (navegación, aprendizaje reforzado, búsqueda…), sino que también proporciona una visión más profunda de cómo los humanos toman decisiones y resuelven problemas.

A través de la unión de disciplinas aparentemente dispares, como la informática, la psicología cognitiva, la ingeniería de software y la teoría de juegos, podemos crear una metodología híbrida con resultados prometedores. Este enfoque, ejemplificado por la "Plataforma para recreación de estrategia basada en aprendizaje reforzado", abre nuevos horizontes en la intersección de estas disciplinas.

Para la educación y la formación, los resultados de este trabajo también tienen implicaciones significativas. La plataforma puede usarse como una herramienta educativa que facilita la comprensión de conceptos complejos de IA y aprendizaje reforzado de una manera más atractiva y accesible. Los estudiantes y profesionales pueden experimentar de primera mano cómo los algoritmos de aprendizaje reforzado interactúan y adaptan su comportamiento en respuesta a diferentes situaciones estratégicas, lo que conduce a un aprendizaje más profundo y eficaz.

La industria del videojuego también se beneficiará de este trabajo. Al demostrar la eficacia de los algoritmos de aprendizaje reforzado en la mejora de la jugabilidad y la experiencia del jugador, es probable que veamos una influencia en cómo se desarrollan y diseñan futuros TBRPG. Esto podría abrir la puerta a la creación de juegos más inteligentes y desafiantes, lo que a su vez podría elevar el nivel de toda la industria.

Más allá del ámbito del juego, la iniciativa ofrece una perspectiva única sobre la toma de decisiones humanas. Al analizar y comparar cómo los humanos y las IA gestionan diferentes situaciones estratégicas, se abre la posibilidad de descubrir nuevas facetas de nuestras propias fortalezas y debilidades como tomadores de decisiones. Este

conocimiento tiene un valor potencialmente significativo en una serie de disciplinas, entre ellas la psicología cognitiva y la economía conductual.

De este modo, a pesar de sus raíces en el mundo de los videojuegos, las implicaciones y beneficios de este proyecto se extienden a campos mucho más amplios. Su enfoque innovador subraya el potencial de los videojuegos como una valiosa herramienta para la investigación, la educación y la mejora de nuestra comprensión de la toma de decisiones humanas.

2.- Objetivos del proyecto

Este proyecto tiene como propósito diseñar e implementar una plataforma para la recreación de estrategias basada en aprendizaje supervisado en el ámbito de los juegos de estrategia. Los objetivos específicos del proyecto son los siguientes:

- Investigar y analizar los conceptos teóricos y técnicos fundamentales relacionados con el aprendizaje supervisado y su aplicación en el desarrollo de juegos de estrategia, identificando las técnicas y enfoques más relevantes en este campo.
- Estudiar y evaluar las plataformas y herramientas existentes para la recreación y análisis de estrategias en juegos de estrategia, identificando áreas de mejora y limitaciones en las soluciones actuales.
- Diseñar la arquitectura y los componentes de la plataforma propuesta, considerando aspectos clave como la escalabilidad, modularidad y facilidad de uso para los desarrolladores y aficionados a los juegos de estrategia.
- Desarrollar e implementar módulos y componentes que faciliten la integración de modelos de IA basados en aprendizaje supervisado, la recolección de datos y la implementación de estrategias en la plataforma propuesta.
- Realizar pruebas y evaluaciones de rendimiento y funcionalidad de la plataforma en diversos escenarios y juegos de estrategia previamente creados como casos de estudio, analizando el impacto de la integración del aprendizaje supervisado en la generación y adaptación de estrategias.
- Analizar la eficiencia y precisión de los modelos de IA implementados en la plataforma, evaluando su capacidad para mejorar y adaptar estrategias en tiempo real en función de las acciones del jugador.
- Documentar y difundir los resultados del proyecto, proporcionando información detallada sobre la plataforma desarrollada, sus características y funcionalidades, así como las conclusiones y futuras líneas de investigación derivadas del proyecto.

3.- Conceptos teóricos

En este apartado se va a explicar los conceptos teóricos de manera que se pueda tener una base sobre el campo en el que se va a desarrollar el proyecto y así poder entender en mayor medida como implementarlo.

3.1.- Algoritmo A*

El algoritmo A* es un popular algoritmo de búsqueda en grafos utilizado en una variedad de aplicaciones, desde la inteligencia artificial en videojuegos hasta la planificación de rutas en sistemas de GPS. A* es especialmente conocido por su capacidad para encontrar el camino más corto entre dos nodos o puntos en un mapa o un grafo.

Este algoritmo fue introducido por primera vez por Peter Hart, Nils Nilsson y Bertram Raphael en 1968. A* es una versión mejorada del algoritmo de Dijkstra y utiliza heurísticas para guiar su búsqueda, lo que lo hace más eficiente en muchos casos.

El funcionamiento de A* se basa en mantener dos listas: una lista abierta y una lista cerrada. La lista abierta contiene los nodos que están siendo considerados para la búsqueda, mientras que la lista cerrada contiene los nodos que ya han sido visitados.

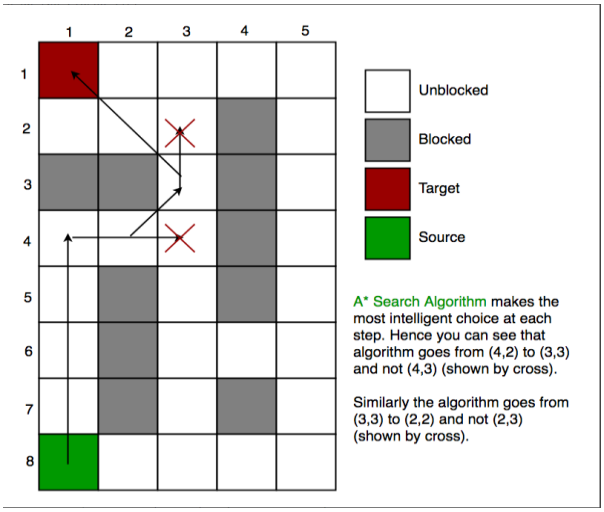


*Ilustración 1: Explicación algoritmo A**

Cada nodo en el grafo tiene tres valores importantes asociados a él:

- g(n): es el costo real para llegar a este nodo desde el nodo de inicio.
- h(n): es la estimación heurística del costo desde este nodo hasta el nodo objetivo.
- f(n): es la suma de g(n) y h(n).

El proceso básico del algoritmo A* es el siguiente:

1. Comienza con el nodo inicial en la lista abierta.
2. Selecciona el nodo de la lista abierta con el valor f(n) más bajo.
3. Si este nodo es el nodo objetivo, entonces el algoritmo ha encontrado el camino y se detiene.
4. Si no es el nodo objetivo, se mueve a la lista cerrada y se añaden todos sus nodos vecinos a la lista abierta.
5. Para cada nodo vecino, se calcula su valor g(n) y se actualiza su valor f(n).
6. Este proceso se repite volviendo al paso 2, hasta que se encuentre el nodo objetivo o no queden nodos en la lista abierta.

Un aspecto crucial del algoritmo A* es la elección de la función heurística h(n). Para garantizar que A* encuentre el camino más corto, la función heurística debe ser admisible, es decir, nunca debe sobrestimar el costo para llegar al objetivo. Un ejemplo común de una función heurística admisible es la distancia en línea recta desde un nodo hasta el nodo objetivo en un mapa 2D.

La distancia Manhattan (Krause, 1986) , también conocida como distancia de ciudad o distancia L1, es una medida de la distancia entre dos puntos en un sistema de coordenadas basado en una cuadrícula (como un tablero de ajedrez o un mapa de la ciudad de Manhattan, de donde toma su nombre).

En el contexto del algoritmo A*, la distancia Manhattan puede ser utilizada como la función heurística h(n) que mencioné anteriormente. Esto es particularmente útil cuando el movimiento está limitado a una cuadrícula y no se permiten movimientos diagonales.

La distancia Manhattan entre dos puntos es la suma de las diferencias absolutas de sus coordenadas. Por ejemplo, si tienes dos puntos, $P_1(x_1, y_1)$ y $P_2(x_2, y_2)$, la distancia Manhattan entre estos dos puntos se calcula como:

$$Distancia_{Manhattan} = |x_1 - x_2| + |y_1 - y_2|$$

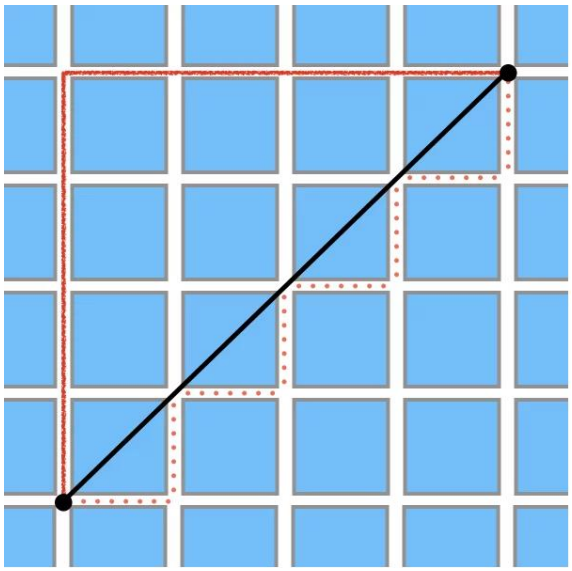


Ilustración 2: Representación distancia de Manhattan

La distancia de Chebyshev (Cantrell, 2000) se usa a menudo como una de estas heurísticas. Es un sistema de medición que permite los movimientos diagonales, a diferencia de la distancia de Manhattan que solo permite movimientos horizontales y verticales. En un juego de cuadrícula, por ejemplo, si se permite el movimiento en diagonal, podría ser una elección más adecuada para la heurística.

La distancia de entre dos puntos (X_1, Y_1) y (X_2, Y_2) se calcula como:

$$Distancia_{Chevyshev} = max(|x1 - x2|, |y1 - y2|)$$

Esto significa que la distancia de Chebyshev es el mayor valor de la diferencia absoluta entre las coordenadas x o y de los dos puntos. Esto permite movimientos diagonales, ya que un movimiento diagonal implica cambiar tanto las coordenadas x como y al mismo tiempo.

Entonces, en términos de la implementación del algoritmo A*,se usaría para calcular la función heurística h(n), que estima el costo del camino más corto desde el nodo n hasta el objetivo.

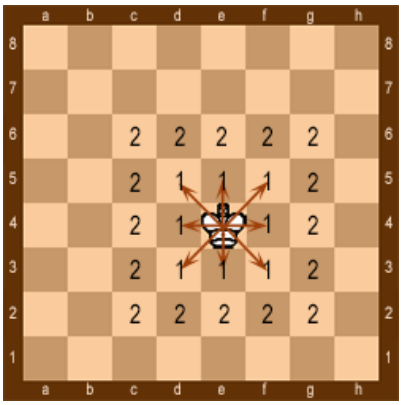


Ilustración 3: Representación de la distancia de Chevyshev

3.2.- Aprendizaje reforzado

El aprendizaje por refuerzo es una rama de la inteligencia artificial y del aprendizaje automático que se centra en cómo un agente puede aprender a tomar decisiones óptimas en un entorno, dado un sistema de recompensas y castigos. Este tipo de aprendizaje se inspira en cómo los seres vivos aprenden de su entorno y sus interacciones.

En el aprendizaje por refuerzo, un agente toma acciones en un entorno, y por cada acción recibe una recompensa o un castigo (una señal de "refuerzo"). El objetivo del agente es aprender una política, que es una estrategia de toma de decisiones que maximiza la recompensa acumulada a lo largo del tiempo.

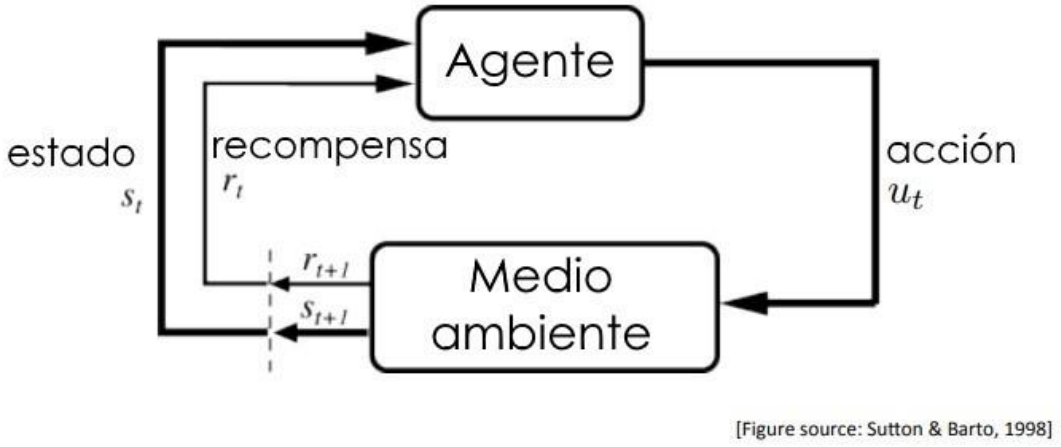


Ilustración 4: Representación del aprendizaje reforzado

El aprendizaje por refuerzo se puede describir con el siguiente ciclo:

1. El agente observa el estado actual del entorno.
2. Basado en esta observación, el agente selecciona y ejecuta una acción.
3. El entorno cambia de estado, y el agente recibe una recompensa o un castigo.
4. El agente utiliza esta retroalimentación para actualizar su política.

Este ciclo se repite, con el agente mejorando continuamente su política a medida que adquiere más experiencia en el entorno.

Un aspecto crucial del aprendizaje por refuerzo es el equilibrio entre la exploración y la explotación. La exploración se refiere a probar acciones que el agente no ha intentado o ha intentado raramente, para ver si conducen a una mayor recompensa. La explotación se refiere a usar la política actual del agente para tomar la acción que, según su experiencia, maximizará la recompensa. Ambas son necesarias para que un agente aprenda una política óptima.

3.3.- Algoritmo de Montecarlo

El método de Montecarlo es una técnica computacional que se basa en el uso de números aleatorios y probabilidades para resolver problemas complejos. El nombre "Montecarlo" se debe al famoso casino en Mónaco, reflejando el elemento aleatorio intrínseco del método.

El método de Montecarlo es útil en una variedad de contextos, especialmente cuando el problema es demasiado complejo para resolverlo con métodos analíticos. Se utiliza en física, matemáticas, economía, inteligencia artificial, juegos, y más.

Este método se basa en la generación de un gran número de resultados aleatorios (o "muestras") y luego se analiza el conjunto de resultados para hacer estimaciones sobre la solución del problema.

El método de Montecarlo también se utiliza para resolver problemas más complejos, como la integración en múltiples dimensiones, la simulación de sistemas físicos, y la planificación y toma de decisiones en IA y juegos. En estos contextos, el método de Montecarlo ofrece una forma de explorar y hacer estimaciones sobre un espacio de posibilidades muy grande y complejo.

En inteligencia artificial y juegos, el método de Montecarlo se utiliza de manera extensiva en la toma de decisiones y planificación. En el juego de Go, por ejemplo, la famosa IA AlphaGo de DeepMind utiliza una variante de este método, llamada Monte Carlo Tree Search (MCTS), para explorar posibles secuencias de movimientos y determinar la mejor acción a seguir. Este enfoque ha demostrado ser muy efectivo, permitiendo a AlphaGo vencer a campeones humanos del juego.

El método de Montecarlo también se utiliza en el campo de la optimización, donde se busca encontrar la solución óptima a un problema en un espacio de soluciones posibles. Por ejemplo, puede ser usado para optimizar los parámetros de un modelo de aprendizaje automático, o para encontrar la mejor ruta en un problema de viajante de comercio.

```
Método MonteCarloDecisionMaking

    Inicializa una lista vacía de resultados

    Para i desde 1 hasta Número de Simulaciones hacer lo siguiente:

        Define un estado inicial

        Mientras que el estado no sea terminal hacer lo siguiente:

            Selecciona una acción basada en alguna política (por ejemplo,
            aleatoria, basada en valores históricos, etc.)

            Ejecuta la acción y observa la recompensa y el próximo estado

            Actualiza el estado al próximo estado

        Fin del bucle Mientras

        Añade la recompensa final a la lista de resultados

    Fin del bucle Para

    Calcula y devuelve la media (y posiblemente otros
    estadísticos como la mediana, el percentil 95, etc.) de los resultados

Fin del método MonteCarloDecisionMaking
```

Ilustración 5: Pseudocódigo de ejemplo del algoritmo de Montecarlo

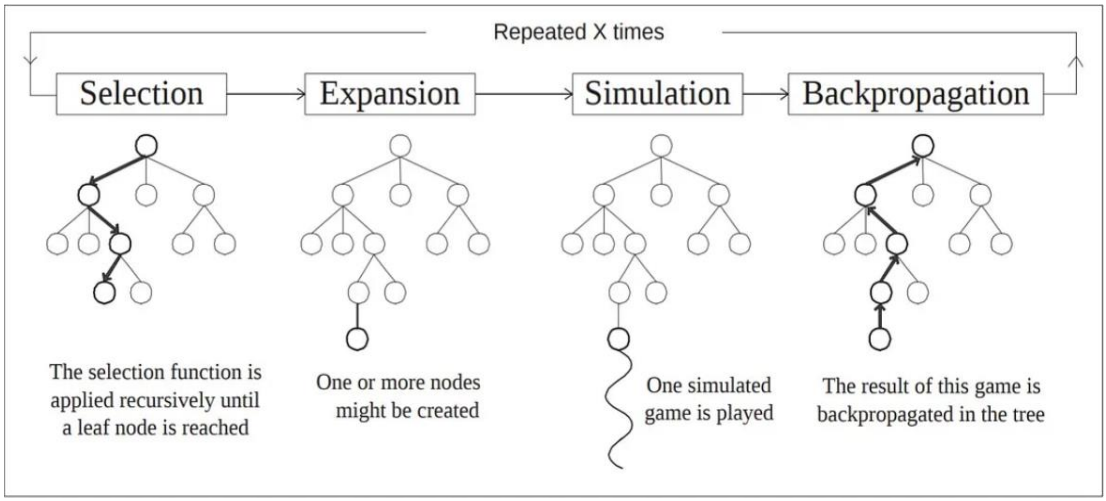


Ilustración 6: Esquema de la simulación

3.4.- Inteligencia Artificial en TBRPG

Los juegos de rol tácticos basados en turnos (TBRPG) ofrecen un entorno complejo y desafiante para la inteligencia artificial (IA). En estos juegos, los jugadores y la IA toman turnos para mover personajes por un mapa de cuadrícula y realizar acciones, como atacar o usar objetos. Para jugar de manera eficaz, la IA debe tomar decisiones estratégicas y tácticas basadas en la información del estado del juego y las acciones de los jugadores.

1. **Toma de decisiones:** En un TBRPG, la IA debe tomar decisiones en cada turno sobre qué personajes mover y qué acciones realizar. Esto puede implicar la evaluación de diferentes opciones y la elección de la que maximice un objetivo, como minimizar el daño a los personajes controlados por la IA o maximizar el daño a los personajes del jugador. La IA puede utilizar algoritmos como MiniMax, Alpha-Beta Pruning, o el algoritmo de Montecarlo para explorar el árbol de decisiones y seleccionar la mejor acción.
2. **Evaluación del estado del juego:** La IA debe evaluar el estado actual del juego para tomar decisiones informadas. Esto puede implicar el uso de funciones de evaluación que asignen un valor a cada estado del juego, basado en factores como la salud de los personajes, la posición estratégica y el número de acciones disponibles.
3. **Planificación de la ruta:** La IA debe calcular las rutas eficientes para mover a sus personajes por el mapa de cuadrícula. El algoritmo A* es comúnmente utilizado para esto, a menudo con la distancia de Manhattan o la distancia de Chebyshev como heurística cuando los movimientos diagonales están permitidos.
4. **Aprendizaje y adaptación:** La IA puede utilizar técnicas de aprendizaje automático para adaptarse a las tácticas del jugador y mejorar su rendimiento a lo largo del tiempo. Por ejemplo, podría usar el aprendizaje por refuerzo para aprender una política que maximice la recompensa a largo plazo, basándose en una función de recompensa que favorezca los estados de juego en los que la IA está en una posición ventajosa.

5. **Generación de contenido procedimental:** En algunos TBRPGs, la IA también puede estar involucrada en la generación de contenido procedimental, como la creación de niveles o la selección de encuentros de enemigos, para proporcionar una experiencia de juego variada y desafiante.

4.- Metodología, técnicas y herramientas

En este apartado se van a presentar las metodologías, las técnicas y las herramientas que se han utilizado durante el desarrollo del proyecto.

Para comenzar, se va a explicar la metodología utilizada siguiendo el proceso unificado, se continuará explicando las técnicas y patrones utilizados entre los que se encuentran el patrón Singleton o la metodología para la elicitación de requisitos de Duran y Bernárdez y para finalizar se expondrán las herramientas relevantes usadas, en las que consta el motor grafico de UNITY, el lenguaje de programación y herramientas auxiliares.

4.1.- Metodología

El desarrollo del proyecto va a realizarse siguiendo el Proceso Unificado del Desarrollo Software. El proceso unificado del Desarrollo Software es un enfoque para el desarrollo de software que se basa en la iteración y la adaptación. Este se centra en la entrega continua de software funcional y tiene como principal marco un ciclo de vida de desarrollo de software.

En este proceso es un marco de trabajo o extensible que puede ser adaptado a organizaciones o proyectos. En lo respectivo al ciclo de vida, este se puede enmarcar en un ciclo iterativo e incremental, lo cual se refiere a que hay que realizar varias iteraciones las cuales están basadas en las etapas clásicas del Proceso Unificado del Desarrollo de Software. Estas etapas, las cuales podemos definir como modelado del negocio, requisitos, análisis, diseño, implementación, pruebas y despliegue, se van a ir refinando y formando poco a poco para llegar a obtener un producto software completo.

Este tipo de ciclo de vida se ha escogido debido a que el tiempo estimado para el proyecto, que son 8 meses es un poco ajustado, el desarrollo únicamente lo va a implementar una persona y el proyecto contiene diversos módulos con tecnologías muy diferentes que hay que investigar, diseñar e implementar.

Para comenzar, se ha elaborado una planificación temporal que se adapta al ciclo de vida y a las tecnologías que se van a utilizar en cada momento obteniendo las siguientes iteraciones:

- **Iteración inicial**: En esta iteración se va a realizar una investigación del estado del arte de cada una de las funcionalidades del proyecto, observar que tecnologías y arquitecturas proponen soluciones similares y así poder elaborar una primera propuesta simplificada de los requisitos del análisis y del diseño del sistema para así poder establecer mejor que componentes forman parte del proyecto y que van a realizar cada uno.
- **Iteración controles de cámara e iteración con el entorno:** En esta iteración se lleva a cabo el control de la cámara del juego, así como crear todos los componentes que se utilizara para que también se pueda interactuar con el escenario.
- **Iteración construcción del juego:** En esta iteración se procederá con la construcción e implementación propiamente dicha de los algoritmos y componentes que darán vida a nuestro juego, entre los que se encuentran aquellos importantes como el algoritmo de búsqueda del camino más corto A*, se define su heurística con la distancia de Chevyshev así como el algoritmo de aprendizaje reforzado para la toma de las decisiones de las diferentes unidades conocido como algoritmo o método de Montecarlo.
- **Iteración final:** En esta iteración se refinan los aspectos finales del proyecto tales como la documentación, pulimento de posibles bugs que se han producido durante la fase anterior e implementación de los sistemas faltantes.

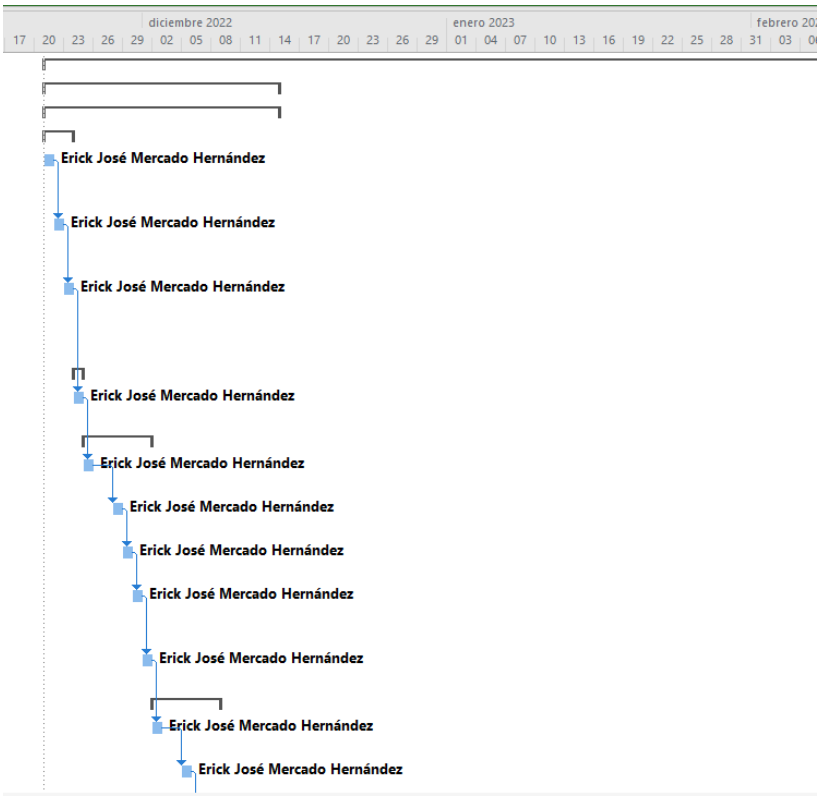


Ilustración 7:Diagrama de Gantt (I)

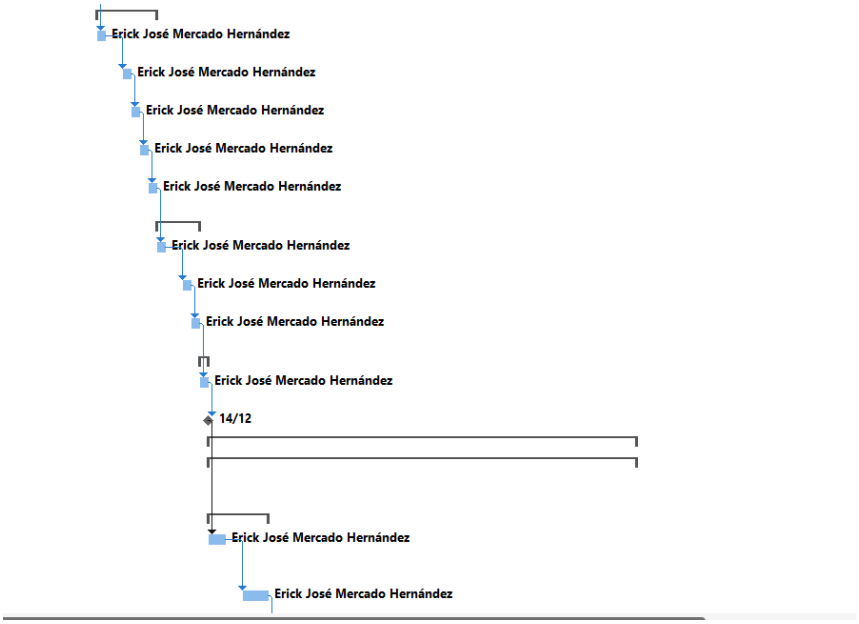


Ilustración 8: Diagrama de Gantt (II)

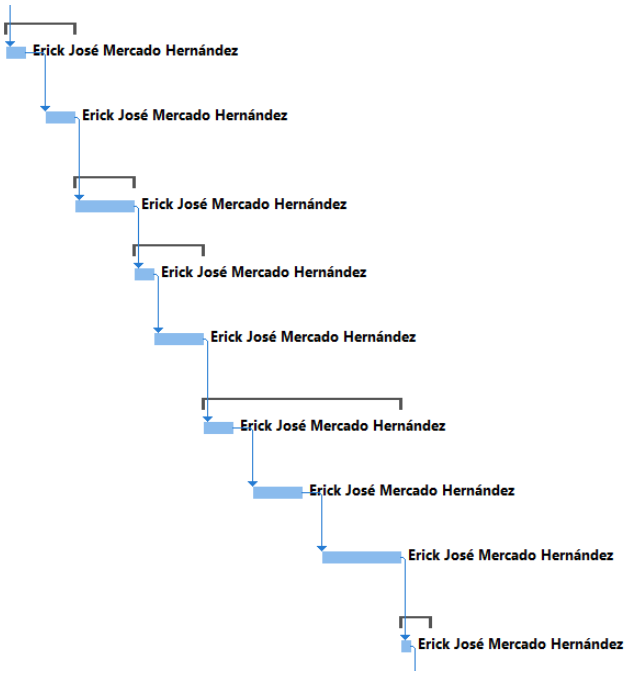


Ilustración 9: Diagrama de Gantt (III)

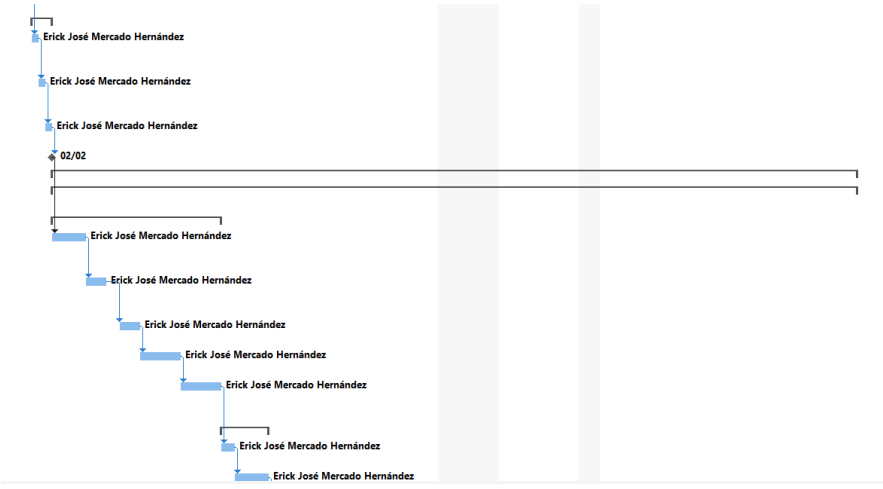


Ilustración 10: Diagrama de Gantt (IV)

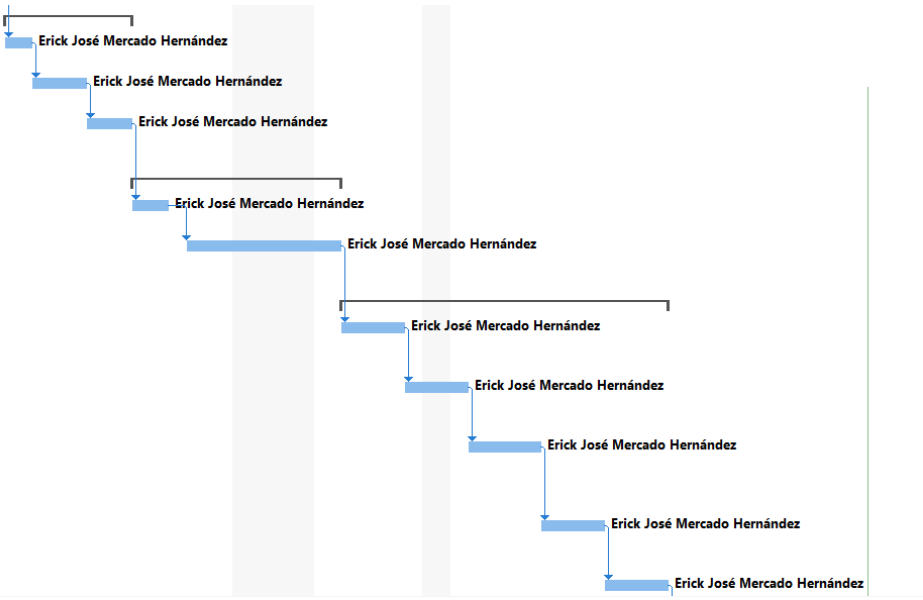


Ilustración 11: Diagrama de Gantt (V)

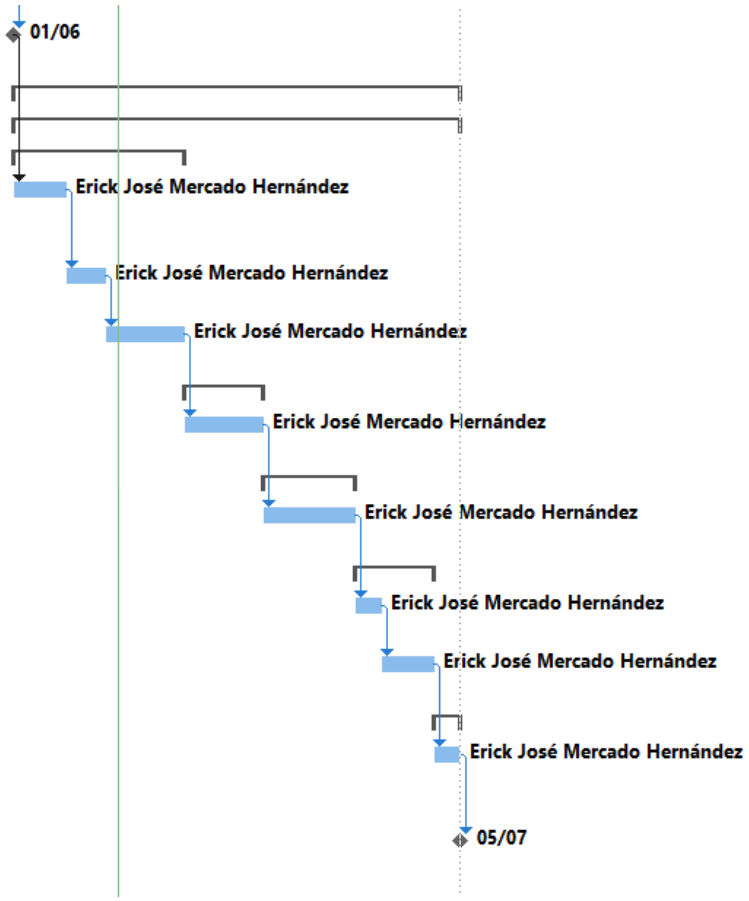


Ilustración 12: Diagrama de Gantt (VI)

4.2.- Técnicas

En este apartado se van a exponer las diferentes técnicas que se han utilizado a lo largo del desarrollo del proyecto.

4.2.1.- Patrón Singleton

Singleton es un patrón de diseño de software que garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella.

La clase es la encargada de crear su propia instancia y de mantener una referencia a ella y esta proporciona un método estático que permite a las demás clases acceder a la instancia sin necesidad de crear una nueva.

El patrón Singleton permite limitar el acceso a los recursos compartidos y garantizar que sólo exista una instancia e esos recursos. También permite una mayor flexibilidad al permitir que la instancia de Singleton sea creada y destruida dinámicamente.

4.2.2.- Patrón Abstrac Factory

El patrón de diseño Abstract Factory, o Fábrica Abstracta, es un patrón creacional que proporciona una interfaz para crear familias de objetos relacionados sin especificar sus clases concretas. En otras palabras, este patrón permite a los clientes utilizar los métodos proporcionados para crear diferentes objetos de diferentes familias sin tener que preocuparse por los detalles de implementación de estas clases.

4.2.2.- Metodología para la Elicitación de Requisitos de Sistemas de Software de Durán y Bernárdez.

La metodología para la Elicitación de Requisitos de Sistemas de Software de Durán y Bernárdez es una técnica para recopilar y documentar los requisitos funcionales de un sistema de software en el ámbito de la investigación. Esta fue propuesta por Amador Durán Toro y por Beatriz Bernárdez Jiménez, de la Universidad de Sevilla, a principios

del año 2000 y desde su publicación hasta el día de hoy se ha convertido en un estándar de la industria del software.

En esta metodología se exponen las siguientes tareas para poder realizar la elicitación y la documentación de los requisitos:

- Definir los objetivos y el alcance del proyecto y establecer el plan de elicitación de requisitos.
- Recopilar los requisitos del sistema utilizando diferentes técnicas.
- Verificar que los requisitos están completos y son consistentes.
- Establecer prioridades.
- Documentar los requisitos de manera clara y precisa para su posterior implementación.

4.3.- Herramientas

Para la realización del proyecto se han utilizado las siguientes herramientas que se van a exponer a continuación, de forma que, basándonos en la investigación inicial, eran algunas de las mejores para poder llevar a cabo el desarrollo del sistema.

4.3.1.- Motor gráfico

A lo largo de este punto se van a describir el motor grafico que se ha utilizado en el desarrollo del proyecto.

4.3.1.1 Unity

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies. Desde su lanzamiento en 2005, Unity se ha convertido en una de las plataformas de desarrollo de videojuegos más populares y se utiliza tanto en la industria de los videojuegos como en industrias como la arquitectura, la ingeniería y la construcción.

Algunos conceptos teóricos principales de Unity incluyen:

1. **Motor de juego:** Unity es un motor de juego, lo que significa que proporciona un marco para desarrollar juegos de computadora. Esto incluye renderizado de gráficos, física de simulación, sonido, scripting, animación y mucho más.
2. **Multiplataforma:** Unity es conocido por su capacidad para exportar juegos a múltiples plataformas. Estas incluyen Windows, Mac, Linux, Android, iOS, consolas de juegos y más.
3. **Entorno de programación**: Unity utiliza principalmente C# para la programación de comportamientos de juego y características. Esto incluye el control de los personajes, la gestión de la interfaz de usuario, la interacción con bases de datos, la manipulación de gráficos y sonidos, y muchas otras funcionalidades.
4. **Interfaz de Usuario (UI):** La interfaz de usuario de Unity se divide en varias partes, incluyendo la Vista de Escena, donde los desarrolladores pueden manipular objetos en 3D en tiempo real; el Inspector, que permite ver y editar las propiedades de los objetos; y la Vista de Juego, que permite a los desarrolladores ver su juego tal como lo haría un jugador.
5. **Assets y Prefabs**: Los "assets" o "recursos" son cualquier elemento que se utiliza en el juego, como modelos 3D, texturas, sonidos, scripts, etc. Los prefabs, o prefabricados, son instancias de objetos que se pueden reutilizar en varias escenas, permitiendo una mayor eficiencia y consistencia.
6. **Componentes y sistemas de juego**: Unity permite a los desarrolladores construir sus juegos utilizando un enfoque basado en componentes. Esto significa que los objetos en un juego están formados por componentes individuales que definen su comportamiento y características.
7. **Scripting**: Unity utiliza el lenguaje de programación C# para scripting. Los scripts se utilizan para controlar el comportamiento de los objetos en el juego y para implementar la lógica del juego.
8. **Unity y Realidad Virtual/Aumentada**: Unity se ha convertido en una plataforma líder para el desarrollo de aplicaciones de realidad virtual (VR) y realidad aumentada (AR). Unity soporta una variedad de dispositivos de VR/AR y tiene

características integradas para ayudar a los desarrolladores a crear experiencias inmersivas.

Unity es una plataforma potente y versátil que permite a los desarrolladores crear desde simples juegos 2D hasta simulaciones 3D completas para una variedad de industrias y aplicaciones. Con un gran número de recursos disponibles en línea y una comunidad activa, Unity es una excelente opción para aquellos interesados en el desarrollo de videojuegos y aplicaciones interactivas.

4.3.2.- Lenguaje de programación

En este apartado se van a exponer el lenguaje de programación utilizados en el desarrollo y en el interior de cada lenguaje se van a mostrar los paquetes más relevantes utilizadas en el lenguaje utilizado.

4.3.2.1- C#

C# (pronunciado C Sharp) es un lenguaje de programación orientado a objetos y fuertemente tipado, diseñado por Microsoft en el año 2000 como parte de su plataforma .NET. Aunque se inspira en lenguajes anteriores como Java y C++, tiene sus propias características y convenciones únicas.

Algunos de los aspectos teóricos clave de C#:

1. **Fuertemente Tipado:** C# es un lenguaje fuertemente tipado, lo que significa que el tipo de datos de cada variable y objeto se conoce en tiempo de compilación. Esto ayuda a minimizar errores como operaciones inválidas entre tipos incompatibles.
2. **Orientado a Objetos:** C# es un lenguaje de programación orientado a objetos (OOP), lo que significa que se centra en la creación y manipulación de "objetos", que son instancias de "clases". Las clases son plantillas que definen las propiedades y comportamientos de un objeto.
3. **Interoperabilidad de Plataformas:** C# fue diseñado para la plataforma .NET de Microsoft, pero también se puede usar en una variedad de plataformas a través de

.NET Core y Xamarin. También es el principal lenguaje de programación utilizado en el motor de juego Unity.

4. **Gestión Automática de Memoria:** C# usa un recolector de basura automático, lo que significa que los objetos que ya no son necesarios son automáticamente eliminados de la memoria, liberando así recursos.
5. **Excepciones y manejo de errores:** C# incluye un sólido sistema de manejo de errores basado en excepciones, lo que permite a los programadores manejar errores en tiempo de ejecución y prevenir el bloqueo de aplicaciones.
6. **Seguridad de Tipos:** C# proporciona características de seguridad de tipos, lo que significa que previene operaciones que no son seguras en cuanto a tipos, como conversiones de tipos inseguras, operaciones aritméticas desbordadas, acceso a memoria no asignada, etc.
7. **Soporte para programación asíncrona**: C# proporciona soporte incorporado para la programación asíncrona, que es fundamental para el desarrollo de aplicaciones modernas que requieren tareas no bloqueantes y eficientes en la gestión de recursos.
8. **Lenguaje de alto nivel:** C# es un lenguaje de alto nivel, lo que significa que su sintaxis está diseñada para ser fácilmente entendible por los humanos en lugar de las máquinas. Esto hace que el lenguaje sea más fácil de leer y escribir.
9. **Sintaxis y estructura del lenguaje:** La sintaxis de C# es similar a otros lenguajes C-like, con funciones, variables, operadores, ciclos, declaraciones condicionales, clases y más. C# también soporta la sobrecarga de operadores, los genéricos, los delegados, los eventos, y tiene un soporte robusto para la manipulación de cadenas y expresiones regulares.

4.3.2.1.1.- UnityEngine.UI

Este es un sistema de IU (Interfaz de Usuario) para Unity, y viene con el motor de Unity. Se utiliza para crear interfaces de usuario en tus juegos.

Es un paquete de Unity que proporciona a los desarrolladores las herramientas necesarias para crear y manipular interfaces de usuario en sus juegos. Estas herramientas permiten a los desarrolladores crear menús de juego, pantallas de pausa, indicadores de vida, indicadores de puntuación, diálogos, inventarios, y prácticamente cualquier elemento de interfaz de usuario que se pueda ver en un videojuego.

Algunos componentes claves que `UnityEngine.UI` ofrece:

- **Canvas**: Este es el área donde se dibujan todos los elementos de la UI. Puedes tener varios Canvas en una escena y cada Canvas puede tener su propia cámara de renderización.
- **Panel**: Los paneles se utilizan para organizar elementos de la UI en un grupo, lo que facilita moverlos o manipularlos todos a la vez. Los paneles también pueden tener un fondo, lo que los hace útiles para crear ventanas de diálogo o menús de pausa.
- **Text**: Es un componente que se utiliza para mostrar texto en la UI. Puedes cambiar el tamaño, el color, la fuente y otros atributos del texto.
- **Button**: Los botones son una parte integral de cualquier UI. Puedes asignar acciones específicas a los botones, que se ejecutan cuando el jugador hace clic en el botón.
- **Slider**: Los deslizadores son útiles para cualquier tipo de control de la UI que requiera un rango de valores. Son comúnmente utilizados para cosas como controlar el volumen de la música o el brillo de la pantalla.
- **Image/Sprite**: Estos componentes permiten mostrar imágenes o sprites en la UI, útiles para iconos de elementos, avatares de personajes, indicadores de vida y mucho más.

Todos estos componentes de la UI se pueden personalizar ampliamente en cuanto a su aspecto y comportamiento, y pueden interactuar con scripts escritos en C# para crear UIs interactivas y dinámicas.

Además de estos componentes básicos, `UnityEngine.UI` también incluye sistemas para manejar la navegación de la UI (como el movimiento entre botones utilizando el teclado o el controlador), y para manejar animaciones de la UI.

4.3.2.1.2.- TextMeshPro

TextMeshPro es una solución avanzada para la visualización de texto en Unity. Se trata de un sistema de representación y disposición de texto que proporciona un mayor control sobre la estética y el formato del texto en comparación con la funcionalidad estándar de texto de Unity. TextMeshPro permite a los desarrolladores utilizar una amplia variedad de estilos y efectos en sus elementos de texto.

Algunas características y capacidades clave de TextMeshPro son:

- **Calidad de renderizado**: TextMeshPro proporciona un renderizado de texto de alta calidad y fidelidad. A diferencia del componente estándar de texto de Unity, TextMeshPro utiliza Signed Distance Field (SDF) para renderizar los glifos del texto, lo que permite un escalado y zoom suave sin pixelación o borrosidad.
- **Soporte para fuentes**: TextMeshPro permite el uso de fuentes TrueType y OpenType, y proporciona una herramienta para generar "fuentes de atlas" a partir de estas, que son más eficientes para su renderizado.
- **Estilos de texto y decoraciones**: Con TextMeshPro, puedes aplicar estilos de texto como negrita, cursiva, subrayado y tachado. Además, es posible ajustar el espaciado entre letras, palabras, líneas y párrafos.
- **Efectos de texto**: TextMeshPro ofrece una variedad de efectos para mejorar la apariencia del texto. Esto incluye sombras, contornos, degradados, resplandor, y otros.

- **Manejo de texto 3D**: A diferencia del componente de texto estándar de Unity, TextMeshPro puede trabajar con texto en un entorno 3D, lo que permite la creación de texto tridimensional con profundidad y perspectiva.
- **Interacción con texto**: TextMeshPro permite la interacción con el texto en formas más avanzadas, como el reconocimiento de enlaces y el resaltado de texto al pasar el cursor sobre él.

4.3.2.1.3.- Cinemachine:

Cinemachine es una suite de herramientas cinematográficas para Unity que mejora y simplifica la manera en que los desarrolladores pueden manipular y controlar cámaras en sus juegos. Cinemachine puede ayudar a los desarrolladores a crear cámaras dinámicas, de seguimiento, de tercera persona, de vista superior, y muchos otros tipos de cámaras, con una gran cantidad de opciones de personalización.

Algunas características clave de Cinemachine son:

- **Virtual Cameras**: Las cámaras virtuales son una característica central de Cinemachine. Son entidades que no renderizan por sí mismas, sino que describen cómo se debe mover y comportar una cámara. Puedes tener varias cámaras virtuales y cambiar entre ellas, lo que permite crear fácilmente cortes y transiciones de cámara.
- **Cinemachine Brain**: Este es el componente que se adjunta a una cámara de Unity. Gestiona todas las cámaras virtuales en la escena y decide cuál de ellas está controlando la cámara física en cualquier momento. También se encarga de las transiciones y mezclas entre diferentes cámaras virtuales.
- **Blend System**: El sistema de mezcla de Cinemachine te permite definir cómo se realiza la transición entre diferentes cámaras virtuales. Puedes controlar la duración y el estilo de la mezcla para crear transiciones de cámara suaves.
- **Noise and Shake**: Cinemachine proporciona sistemas de ruido y vibración que puedes utilizar para crear efectos de cámara como temblores o movimientos

aleatorios, que son útiles para escenas de acción o para simular el movimiento de una cámara handheld.

- **Camera Tracking**: Cinemachine puede seguir objetos, orientarse hacia ellos y reencuadrarlos automáticamente. Esto es útil para las cámaras de seguimiento, que se mueven y giran para seguir a un personaje o a otro objeto en movimiento.
- **Cinemachine Impulse**: Este sistema permite a los desarrolladores generar y propagar eventos de fuerza física que pueden influir en las cámaras. Por ejemplo, una explosión en el juego podría hacer que la cámara tiemble.

4.3.3.- Herramientas auxiliares

En este apartado se van a describir las principales herramientas que han ayudado en el desarrollo del proyecto, tanto para la creación del código como para la realización de la documentación.

4.3.3.1.- Git, GitHub

Git es un sistema de control de versiones de código abierto utilizado para llevar un seguimiento de los cambios en archivos y carpetas y para coordinar el trabajo en proyectos de software. El sistema de control de versiones permite a los desarrolladores trabajar de forma independiente y hacer commit de sus cambios en su propio repositorio local, lo que facilita la colaboración y el trabajo en equipo (Guervós, 2017).

GitHub es una plataforma en línea de código abierto que proporciona alojamiento y gestión de proyectos de software utilizando el sistema de control de versiones Git. La plataforma se utiliza para compartir y colaborar en proyectos de software. Ofrece la posibilidad de hacer seguimiento de errores y solicitudes de características, la integración con diferentes servicios de integración continua y el soporte para la revisión de código (Tsitoara, 2020).

4.3.3.2.- EZ Estimate

EZ Estimate es una herramienta CASE (Kendall, 2005) utilizada para generar estimaciones de costos de proyectos. Esta estima dicho coste a partir del modelo de

requisitos y la complejidad asociada a ellos. Nos da un resultado del coste del tiempo que va a tomar en horas de persona.

4.3.3.3.- Microsoft Project

Microsoft Project es una herramienta de software de gestión de proyectos utilizada para planificar, seguir y analizar proyectos. La herramienta se utiliza para organizar y llevar a cabo tareas y actividades necesarias para completar un proyecto dentro de un plazo determinado. Además, se pueden crear y asignar tareas a diferentes miembros del equipo, establecer dependencias entre tareas y seguir el progreso de un proyecto. (Biafore, 2013)

Ofrece una amplia variedad de herramientas de análisis y visualización, como la posibilidad de generar informes y gráficos que muestren el progreso del proyecto y el uso de recursos, entre los que se encuentra el diagrama de Gantt el cual ayuda a visualizar las tareas y principales hitos de una mejor forma.

4.3.3.4.- Visual Paradim

Visual Paradigm es una herramienta de software de modelado y diseño utilizada para crear y gestionar modelos de sistemas (Henderi, 2021). La herramienta se utiliza para representar y documentar de manera visual diferentes aspectos de un sistema, como su estructura, funciones, flujo de trabajo y relaciones entre diferentes elementos.

Con Visual Paradigm, los usuarios pueden crear y gestionar modelos utilizando diferentes lenguajes de modelado, como UML (Fowler, UML gota a gota, 1999). La herramienta también permite a los usuarios crear diferentes tipos de modelos, como diagramas de clases, diagramas de actividad y diagramas de secuencia, etc.

4.3.3.5.- Doxygen

Doxygen es una herramienta de documentación que se utiliza para generar documentación a partir de comentarios en código fuente. Aunque no es específica de Unity (y de hecho se utiliza en una amplia gama de proyectos de software), puede ser útil en proyectos de Unity para ayudar a documentar y entender los scripts de C#.

Algunas características clave de Doxygen:

- **Compatibilidad con múltiples lenguajes:** Doxygen puede procesar código fuente escrito en varios lenguajes de programación, incluyendo C#, el lenguaje utilizado para escribir scripts en Unity. También es compatible con C++, C, Objective-C, Python, Java, PHP, entre otros.
- **Generación de documentación automática:** Doxygen genera documentación a partir de comentarios en el código fuente. Puedes estructurar estos comentarios de una manera específica para indicar qué partes del comentario deben utilizarse para qué partes de la documentación.
- **Varias salidas de documentación:** Doxygen puede generar documentación en varios formatos, incluyendo HTML (que puede ser vista en un navegador web) y LaTeX (que puede ser convertido a PDF para una documentación impresa).
- **Diagramas de clases:** Si usas ciertas etiquetas en tus comentarios, Doxygen puede generar diagramas de clases a partir de tu código, lo que te ayuda a visualizar las relaciones entre diferentes clases y métodos.
- **Integración con IDEs:** Muchos entornos de desarrollo integrados (IDEs) tienen algún tipo de soporte para Doxygen, lo que te permite ver la documentación generada directamente en tu IDE.

5.- Aspectos relevantes del desarrollo

En este apartado se van a exponer los aspectos relevantes del desarrollo. Se va a comenzar explicando el proceso de desarrollo desde el punto de vista de la ingeniería del software y se van a poder observar aspectos como el modelo de requisitos, el modelo de análisis y el modelo del diseño.

Posteriormente, se va a mostrar una visión del sistema a alto nivel, explicando la arquitectura del sistema y todos sus componentes.

Se continuará con cada una de las partes del sistema por separado, explicando la investigación realizada en lo relevante a ellas, el funcionamiento, tecnologías y también mostrando el diseño que se ha elegido para cada apartado.

Por último, se va a explicar los aspectos que tienen que ver con el despliegue.

5.1.- Proceso de desarrollo

Para el proceso de desarrollo se ha seguido la metodología del proceso unificado, el cual ya se ha presentado en el apartado de metodología. El proceso unificado presenta el ciclo de vida de un proyecto el cual es iterativo e incremental.

El proceso de desarrollo del sistema software se ha divido en 4 iteraciones. Para cumplimentar dichas iteraciones se van a seguir los siguientes pasos que definen también el ciclo de vida de un proyecto software, los cuales son investigación, requisitos, análisis, diseño, implementación y pruebas.

5.1.1.- Investigación

El proceso de investigación, el cual se describe a lo largo del Anexo II, se ha realizado durante todo el transcurso de las diferentes iteraciones que conforma el proyecto. Esto se debe a que así se focaliza toda la atención de esa iteración en específico o en la generación de algo designado para dicha iteración.

En base a lo expuesto anteriormente, se va a presentar resumidamente el objetivo de la investigación tratado en cada iteración del proyecto y se van a exponer los diferentes resultados obtenidos.

- **Iteración inicial**:
    - Para comenzar, se lleva a cabo una investigación de los casos y problemáticas más importantes que se den a lo largo del desarrollo de un proyecto de este tipo. De las que se ha concluido que debe desarrollarse de tal manera que se pueda usar todo el potencial del motor gráfico, así como la comunicación de los diferentes componentes que conforman las diferentes escenas del proyecto.
    - Una vez conocida la problemática, se va a llevar a cabo una investigación de otros proyectos similares para investigar sus mecánicas y como se han llevado a cabo las diferentes soluciones. En cuanto a este tipo de soluciones se ha encontrado bastantes ejemplos dentro del mismo género de videojuego en el que me apoyo para realizar este proyecto se encuentran juegos como “XCOM” o “Fire Emblem”, que, si bien tienen enfoques y mecánicas diferentes entre sí, tienen en común el uso de “celdas o baldosas” para mover a la unidad lo que se podría obtener de hay el algoritmo de búsqueda del camino más corto como lo es el A*.
    - Por último, se ha llevado a cabo una investigación sobre que técnicas, lenguaje de programación, software, etc. utilizar para realizar el desarrollo del proyecto. Llegando a la conclusión de que se usara el motor grafico de UNITY y el lenguaje que este utiliza C# para crear los diferentes scripts que nos permitirán ejecutar los diferentes componentes del proyecto.

      Para poder desarrollar una IA se ha optado por usar un algoritmo de aprendizaje reforzado conocido como “Algoritmo de Montecarlo” que permite desarrollar una IA contrincante lo suficientemente capaz de ser desafiante para el jugador y elegir las acciones mediante castigo o recompensa.

- **Iteración controles de cámara e interacción con el entorno:**
    - Primero en esta parte de la iteración se va a investigar sobre cómo se utilizarán los controles de cámara ya que esta es importante para el resto del juego. Todo este desarrollo se realizará mediante script de C# y usando un paquete de Unity llamado Cinemachine que permite el uso de una cámara virtual que puede ser controlada por el jugador. Una vez implementado el GameObject llamado CameraController, añadimos a nuestra VirtualCamera el "transform" del GameObject para que este pueda seguirlo y poder interaccionar con ella. Ahora ya tenemos una cámara funcional que puede rotar sobre el eje Y ademas se puede desplazar sobre los ejes X y Z.
    - Posteriormente, se va a investigar cómo interaccionar con el entorno, para ello será necesario crear varios GameObject y scripts de C# para controlar la dinámica. El control de interacción con el mundo de nuestro juego usaremos el ratón, que podrá interactuar con la interfaz.
- **Iteración construcción del juego:**
    - Primero que se realizara en este apartado es investigar toda la parte de los algoritmos necesarios tanto de encaminamiento como de parte de la IA.

      Para el encaminamiento se ha usado el algoritmo de búsqueda del camino más corto A*. Para la heurística de este apartado se tomará el algoritmo de Chevyshev que permite el desplazamiento diagonal dentro de los ejes X y Z. Este algoritmo se implementará dentro del GameObject Pathfinding que utilizará otro GameObject llamado Grid que será el encargado de generar las casillas por la que se moverán las unidades.

      Para que una unidad tenga movimiento se usara implementara la acción la cual es una clase heredada de la clase abstracta llamada BaseAction. Dicha acción llamada MoveAction es la que permite mover la posición de una unidad dentro de las casillas que se han creado anteriormente en el Grid.

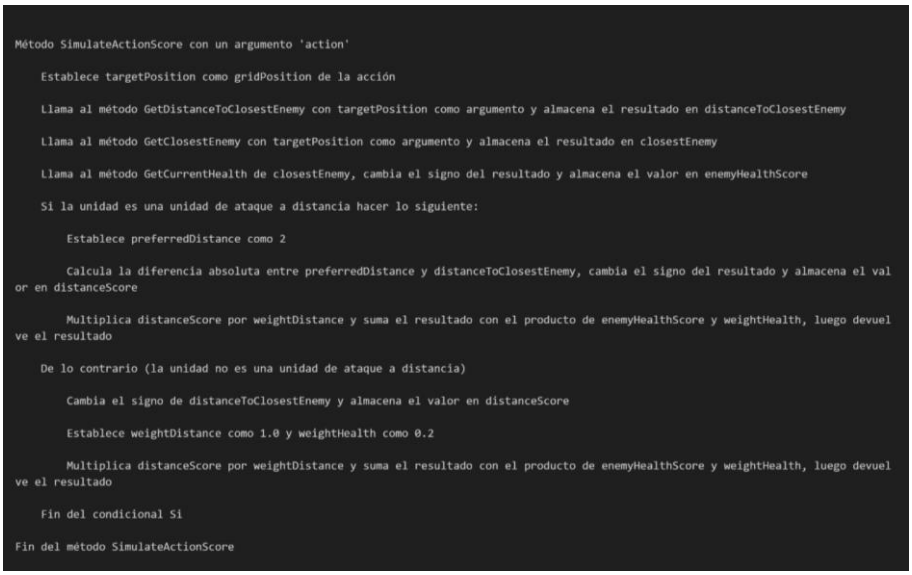
- Se continua con la investigando de los modelos 3D de nuestras unidades. Para ello se ha usado diversos paquetes de assets que hemos importado al proyecto entre los que se encuentran los más importantes “PolygonDungeonRealm”, “Medieval Village Building Pack” y “PoligonFantasyHeroCharacters” entre otros. Para realizar las animaciones para las unidades se ha utilizado la utilidad de Unity llamada “animator” y se han importado modelos de animación de la página web perteneciente a “Adobe” llamada “Mixamo”.
- Una vez creados los modelos se continúa conformando la unidad o personajes con los que jugaremos dentro del juego y que serán nuestros contrincantes, para ello implementamos todas las acciones de ataque, implementamos un sistema de vida, y el script más importante llamado Unit.

  Dentro de este apartado incluimos también todo lo relativo a la inteligencia artificial. Existen dos scripts llamados EnemyAI y EnemyAIAction que permiten implementar un cerebro a cada unidad. Dentro de EnemyAI se encuentra el método que llamara al algoritmo de Montecarlo dentro de la clase BaseAction, el cual elegirá la acción a ejecutar por la unidad contrincante en base a una serie de simulaciones donde se obtendrá que acción tendrá un peso mayor para que la unidad la realice.

```
Método GetBestEnemyAIAction

    Establece numSimulationsPerAction como 1000 // Número de simulaciones de Monte Carlo para cada acción

    Inicializa una nueva lista vacía, enemyAIActionList

    Llama al método GetValidActionGridPositionList y almacena el resultado en validActionGridPositionList

    Para cada objeto gridPosition en validActionGridPositionList hacer lo siguiente:

        Llama al método GetEnemyAIAction con gridPosition como argumento y almacena el resultado en enemyAIAction

        Establece totalScore a 0

        Para i desde 0 hasta numSimulationsPerAction hacer lo siguiente: // Comienza simulaciones de Monte Carlo

            Llama al método SimulateActionScore con enemyAIAction como argumento y almacena el resultado en simulationScore
// Simulación de Monte Carlo

            Incrementa totalScore por simulationScore

        Fin del bucle Para // Fin de las simulaciones de Monte Carlo

        Establece actionValue de enemyAIAction como (totalScore / numSimulationsPerAction) // Establece el valor de la acció
n como el promedio de las simulaciones de Monte Carlo

        Agrega enemyAIAction a enemyAIActionList

    Fin del bucle Para

    Si el tamaño de enemyAIActionList es mayor que 0 hacer lo siguiente:

        Ordena enemyAIActionList de mayor a menor basado en actionValue de cada EnemyAIAction

        Devuelve el primer elemento de enemyAIActionList // Devuelve la acción con el mayor valor promedio de las simulacion
es de Monte Carlo

    De lo contrario

        Devuelve null, indicando que no hay posibles acciones de IA enemiga

    Fin del condicional Si

Fin del método GetBestEnemyAIAction
```

Ilustración 13: Pseudocódigo algoritmo de Montecarlo

Para elegir que acción ejecutar se simularan diferentes escenarios por cada acción a ejecutar, en cada simulación se plantea una recompensa o castigo. Un ejemplo es el caso de mover a la unidad, ya que esta decidirá moverse a la unidad enemiga más cercana, si la unidad es de rango, su máxima distancia que puede acercarse al objetivo es de dos celdas para poder realizar un ataque.



```
Método SimulateActionScore con un argumento 'action'

    Establece targetPosition como gridPosition de la acción

    Llama al método GetDistanceToClosestEnemy con targetPosition como argumento y almacena el resultado en distanceToClosestEnemy

    Llama al método GetClosestEnemy con targetPosition como argumento y almacena el resultado en closestEnemy

    Llama al método GetCurrentHealth de closestEnemy, cambia el signo del resultado y almacena el valor en enemyHealthScore

    Si la unidad es una unidad de ataque a distancia hacer lo siguiente:

        Establece preferredDistance como 2

        Calcula la diferencia absoluta entre preferredDistance y distanceToClosestEnemy, cambia el signo del resultado y almacena el val
or en distanceScore

        Multiplica distanceScore por weightDistance y suma el resultado con el producto de enemyHealthScore y weightHealth, luego devuel
ve el resultado

    De lo contrario (la unidad no es una unidad de ataque a distancia)

        Cambia el signo de distanceToClosestEnemy y almacena el valor en distanceScore

        Establece weightDistance como 1.0 y weightHealth como 0.2

        Multiplica distanceScore por weightDistance y suma el resultado con el producto de enemyHealthScore y weightHealth, luego devuel
ve el resultado

    Fin del condicional Si

Fin del método SimulateActionScore
```

Ilustración 14: Ejemplo de simulación del algoritmo

- Iteración final:
    - En esta iteración se refinarán la documentación generada a lo largo del proyecto, se corregirán los diferentes bugs que se pueden encontrar dentro del juego, así como se obtendrán datos de la experiencia de usuario mediante una demo entregada a un número reducido de personas.
    - Primero se corrigen los bugs visuales que quedan en el proyecto, sobre todo aquellos referentes a los diferentes al motor de colisiones y algunos efectos especiales.
    - Se continua con el refinamiento del diseño del juego, añadiendo todas las funciones no vitales para dar una experiencia más cercana a un rpg.

      Se implementa todo lo referente al sistema de audio y los últimos requisitos que faltan, como lo son el poder modificar las preferencias del usuario, sea nivel de sonido, resolución de pantalla y si se lanzara en formato de ventana o pantalla completa.

- Se recogen los datos de la experiencia de usuario en la cual se hace hincapié en la necesidad de mostrar más información en el combate, en el coste de cada acción que una unidad puede realizar. El resto de experiencia de usuario es satisfactoria, los botones son bastante visibles, de estilo minimalista lo cual concuerda con el resto del juego.

5.1.2.- Especificación de requisitos

Basándome en la propuesta realizada para este Trabajo de Fin de Grado, más la investigación documentada previamente y la planificación temporal en la que se había realizado la funcionalidad que va a ofrecer la plataforma, se van a definir los objetivos principales de la aplicación.

Después de esto, se ha realizado un primer borrador de los requisitos mediante la metodología para la Elicitación de Requisitos de Durán y Bernárdez en la que se ha establecido las principales funcionalidades del sistema que se corresponden con las primeras funcionalidades.

Posteriormente durante el transcurso de iteraciones el modelo de requisitos se fue refinando hasta el resultado definitivo, el cual se va a poder observar a continuación.

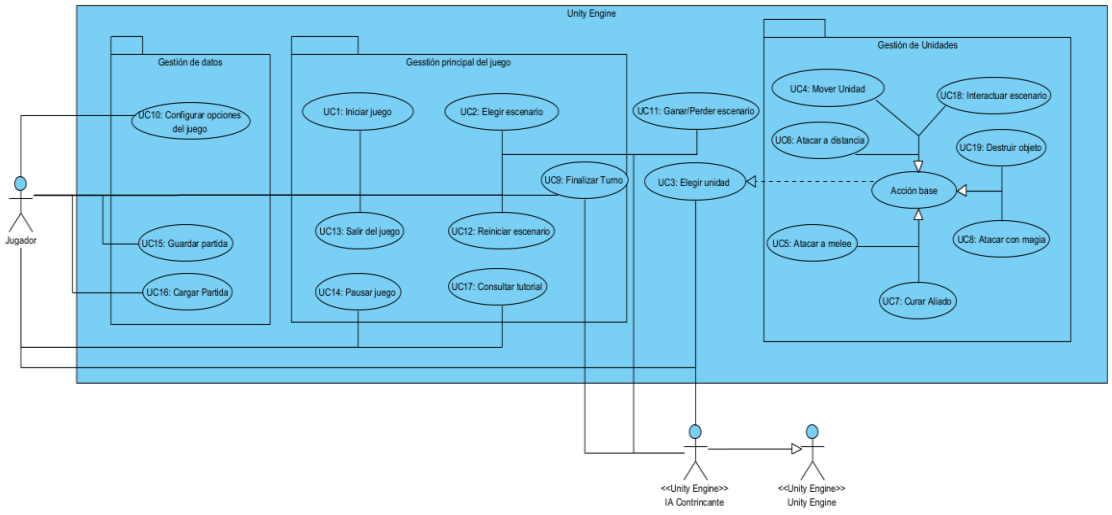


Ilustración 15: Diagrama de casos de uso

En el modelo de requisitos final se han identificado, un total de diecinueve requisitos funcionales, ocho requisitos de información y nueve requisitos no funcionales que van a ayudar a cumplir con el objetivo inicial el cual era la creación de una plataforma de desarrollo de estrategia basado en aprendizaje reforzado. Todos estos requisitos se pueden resumir en las siguientes tablas y si se desea conocer este apartado más en detalle se recomienda consultar el Anexo II.

Tabla 1: Resumen de los requisitos funcionales

Referencia	Nombre	Descripción
CU-01	Menú Principal	El jugador puede elegir entre iniciar el juego, acceder a las configuraciones o salir del juego.
CU-02	Elegir escenario	El jugador selecciona un escenario o nivel para comenzar a jugar.
CU-03	Seleccionar unidad	El jugador selecciona un personaje en pantalla para realizar acciones
CU-04	Mover unidad	El jugador selecciona una unidad y la mueve a una ubicación del mapa
CU-05	Ataque cuerpo a cuerpo	El jugador selecciona una unidad con arma cuerpo a cuerpo para atacar a una unidad enemiga
CU-06	Ataque a distancia	El jugador selecciona una unidad con arma a distancia para atacar a una unidad enemiga
CU-07	Curar aliado	El jugador selecciona una unidad con la capacidad de curar a sus aliados
CU-08	Ataque con magia	El jugador selecciona una unidad con acciones mágicas para dañar a las unidades enemigas
CU-09	Finalizar turno	El jugador finaliza su turno, permitiendo que las unidades controladas por la IA tomen sus acciones.
CU-10	Configurar opciones del juego	El jugador ajusta las opciones del juego, como gráficos, sonidos y controles.
CU-11	Ganar o perder escenario	El jugador gana o pierde un escenario basado en el estado del equipo.
CU-12	Reiniciar escenario	El jugador puede comenzar de nuevo el escenario.
CU-13	Salir del juego	El jugador puede cerrar el juego
CU-14	Pausar el juego	El jugador puede acceder al menú de pausa donde aparecerán diferentes opciones

CU-15	Guardar partida	El jugador guarda su progreso en el juego, incluyendo el estado actual de las unidades y la posición en el escenario o nivel.
CU-16	Cargar partida	El jugador carga una partida guardada previamente, retomando el progreso desde el punto en que se guardó.
CU-17	Consultar tutorial/ayuda	El jugador accede a una sección de ayuda o tutorial que proporciona información sobre cómo jugar y consejos para el juego.
CU-18	Interactuar escenario	Este caso de uso representa la capacidad del jugador o personaje para interactuar con el escenario o entorno en un juego. Puede incluir la interacción con objetos, elementos del entorno, manipulación de objetos, activación de mecanismos y otras interacciones similares que tienen lugar dentro del mundo del juego.
CU-19	Destruir objeto	Este caso de uso representa la capacidad del jugador o personaje para destruir objetos en el entorno del juego. Esto puede incluir romper objetos, derribar estructuras o eliminar elementos del escenario.

Tabla 2: Resumen de requisitos de información

Referencia	Nombre	Descripción
IRQ-01	Mecánicas de juego	Establecer las reglas del juego, como los sistemas de combate por turnos, la progresión del personaje, las habilidades y talentos, y la gestión de recursos e inventario.
IRQ-02	Unidades	Establece las unidades jugables y enemigas en el juego, incluyendo sus atributos, habilidades, animaciones y comportamientos.
IRQ-03	Interfaz de usuario	Diseña y desarrolla la interfaz de usuario del juego, incluyendo menús, pantallas de información, indicadores en el juego y elementos de control.
IRQ-04	Sonido y música	Implementa efectos de sonido, música y ambientes sonoros en el juego para mejorar la experiencia del jugador y la inmersión en el mundo del juego.

IRQ-05	Inteligencia artificial	Diseña y desarrolla la inteligencia artificial de las unidades enemigas y no jugables, incluyendo sus tácticas de combate, comportamientos y toma de decisiones.
IRQ-06	Pruebas y depuración	Realiza pruebas en el juego para identificar y solucionar errores, problemas de rendimiento y otros problemas potenciales.
IRQ-07	Optimización y rendimiento	Trabaja en la optimización del juego para garantizar que funcione de manera eficiente en diferentes dispositivos y plataformas, y cumpla con los Requisitos No Funcionales de rendimiento.
IRQ-08	Gráficos y efectos visuales	Crea y optimiza los gráficos y efectos visuales del juego, incluyendo personajes, entornos, animaciones y efectos especiales.

Tabla 3: Resumen de requisitos no funcionales

Referencia	Nombre	Descripción
NFR-01	Rendimiento	La velocidad y eficiencia con la que el juego se ejecuta, incluyendo el tiempo de carga, la fluidez de las animaciones y la velocidad de respuesta del juego a las acciones del jugador.
NFR-02	Escalabilidad	La capacidad del juego para manejar un aumento en la cantidad de niveles, unidades, objetos y otros elementos sin comprometer su rendimiento o funcionalidad.
NFR-03	Portabilidad	La facilidad con la que el juego puede ser adaptado y ejecutado en diferentes plataformas (por ejemplo, PC, consolas, dispositivos móviles) y sistemas operativos.
NFR-04	Usabilidad	La facilidad de uso del juego, incluyendo la interfaz de usuario, el diseño de los menús, la claridad de las instrucciones y la capacidad de los jugadores para aprender y dominar el juego rápidamente.
NFR-05	Accesibilidad	La capacidad del juego para ser disfrutado por personas con discapacidades, incluyendo opciones de subtítulos, ajustes de contraste de colores y soporte para dispositivos de entrada alternativos.
NFR-06	Estabilidad	La solidez y fiabilidad del juego, evitando bloqueos, errores y problemas de rendimiento que puedan afectar negativamente la experiencia del jugador.
NFR-07	Personalización	La capacidad de los jugadores para personalizar aspectos del juego, como la apariencia de los personajes, la configuración de las teclas de acceso rápido y las opciones de dificultad.
NFR-08	Estética y diseño	La calidad visual y artística del juego, incluyendo gráficos, animaciones, diseño de niveles y estilo general.
NFR-09	Mantenibilidad	La facilidad con la que el juego puede ser actualizado, corregido y mejorado a lo largo del tiempo, así como la capacidad de los desarrolladores para agregar nuevos contenidos y características.

5.1.3.- Análisis y diseño

Al igual que en el caso del apartado anterior, debido a que el modelo de ciclo de vida de este proyecto es iterativo e incremental, al principio se han realizado unos modelos iniciales y a medida que se han ido pasando por las diferentes iteraciones se han ido refinando para obtener unos requisitos acordes a la funcionalidad de este proyecto. Se va a presentar los aspectos relevantes al análisis y el diseño y para consultar más detalles de este apartado se debe consultar el anexo III.

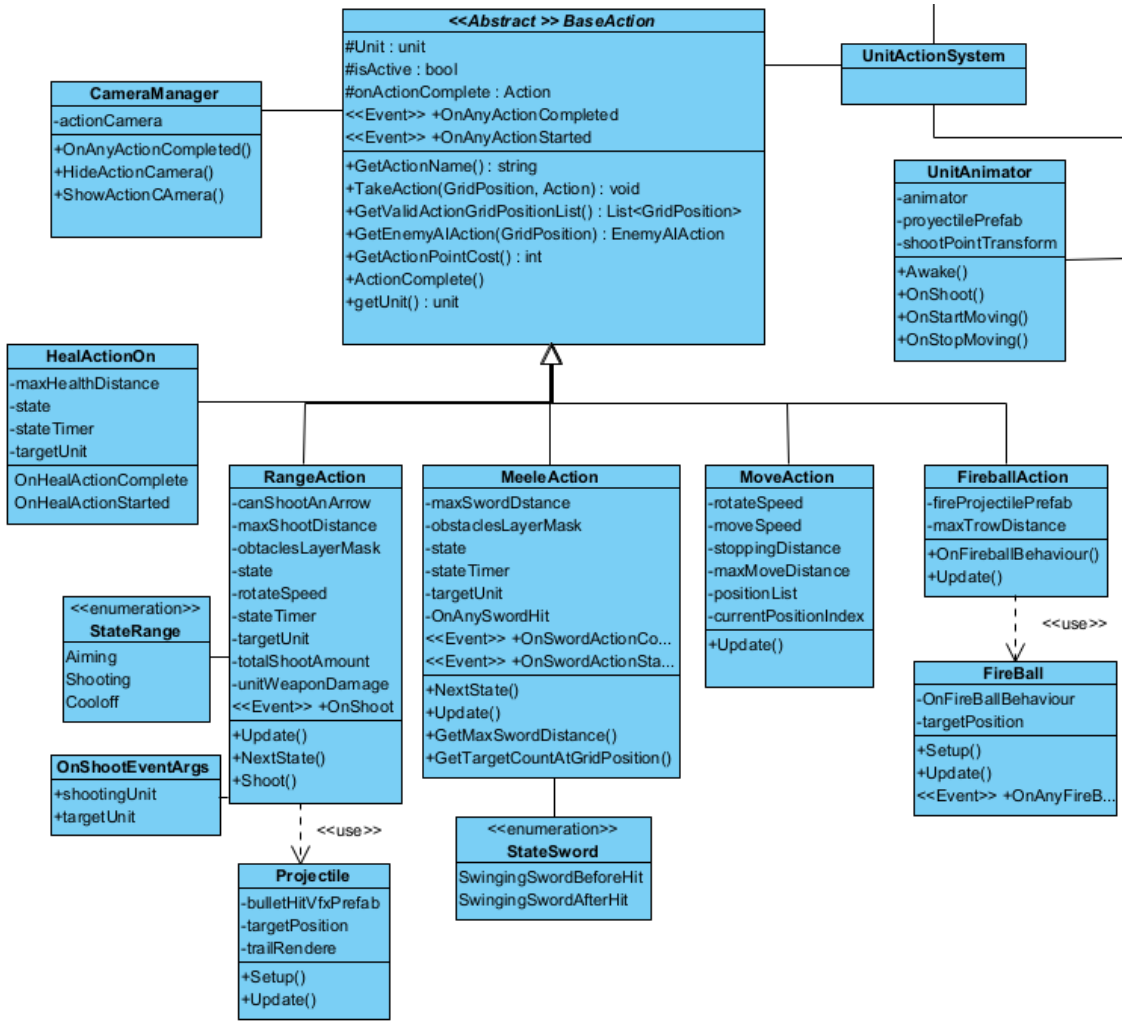


Ilustración 16: Diagrama de clases I

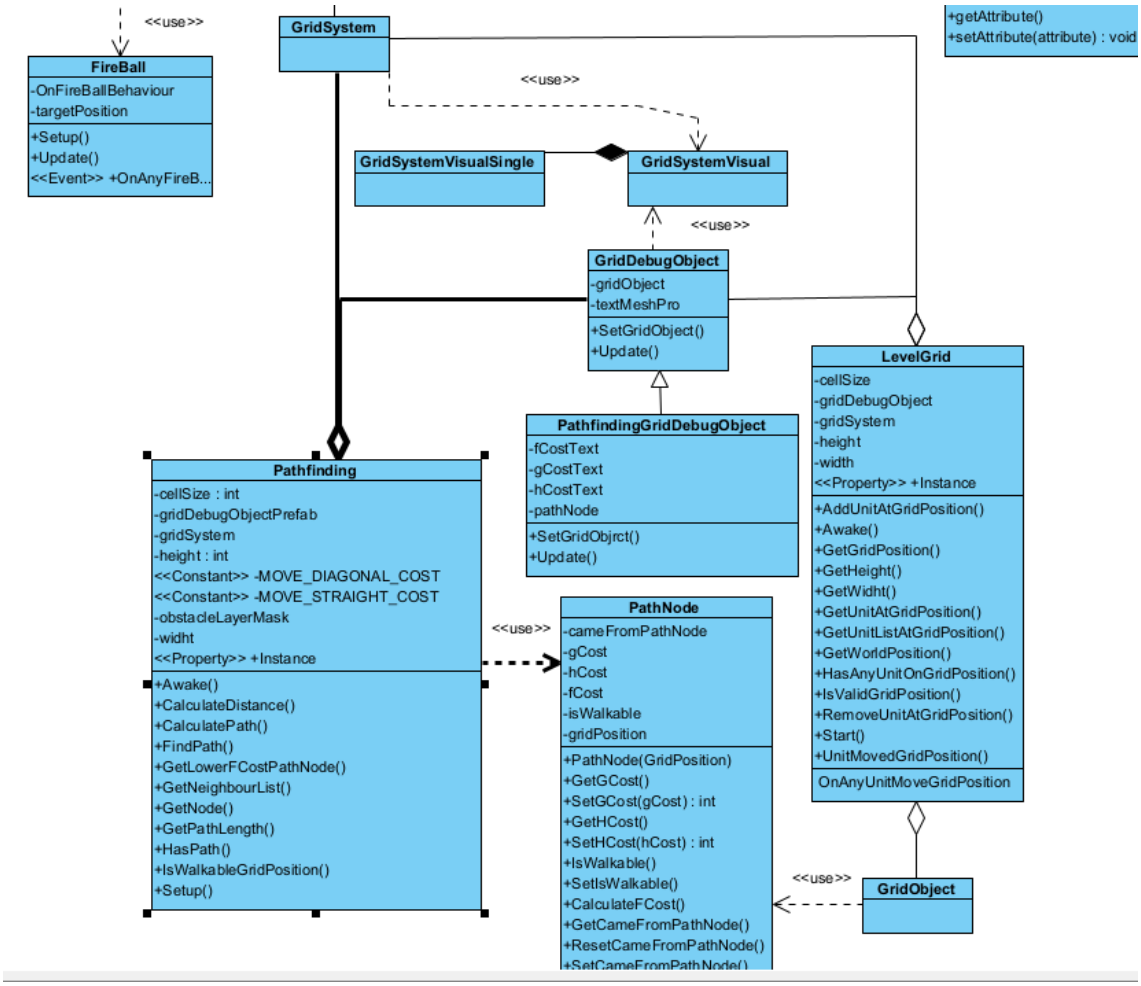


Ilustración 17: Diagrama de clases II

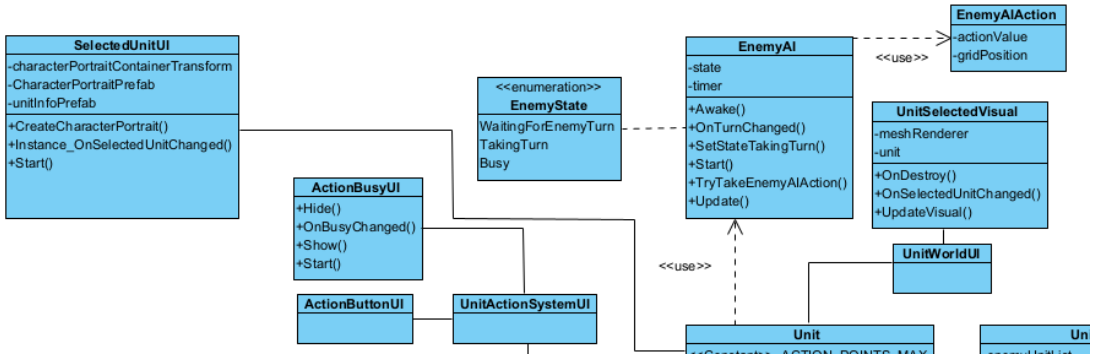


Ilustración 18: Diagrama de clases III

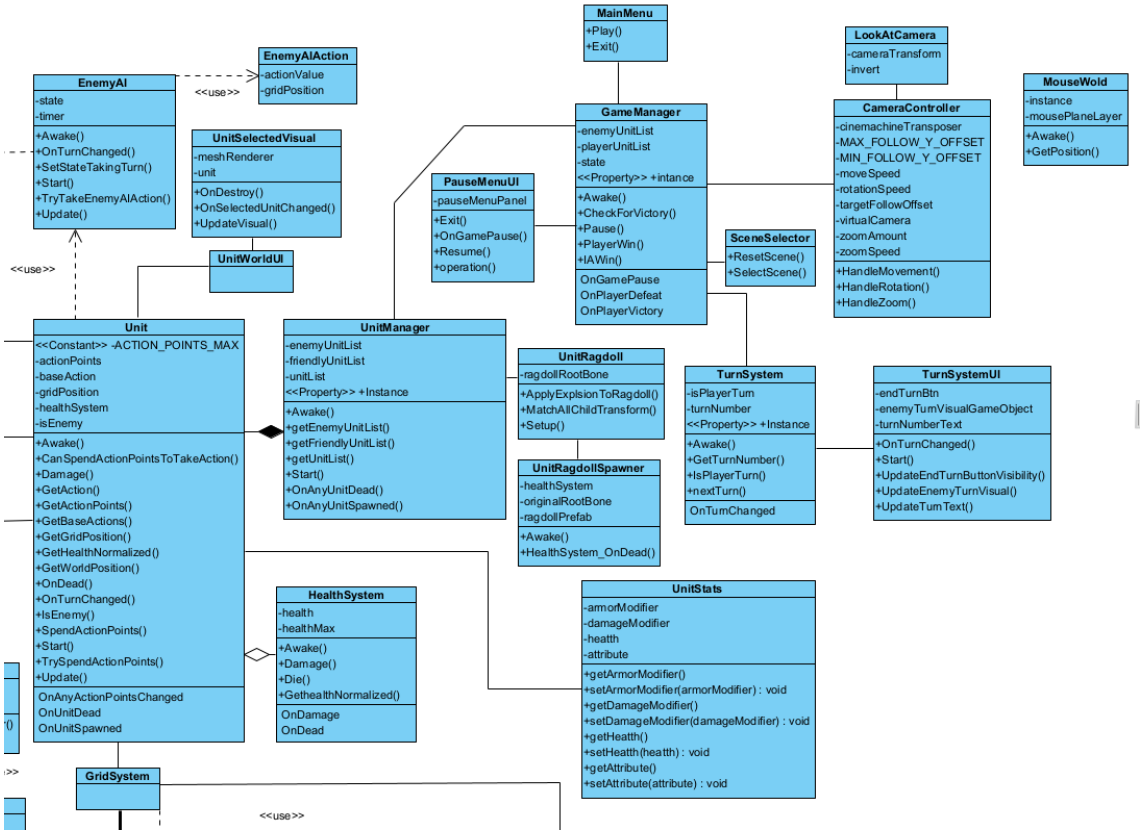


Ilustración 19: Diagrama de clases IV

Las clases que se muestran en las ilustraciones anteriores se explicaran en una tabla cada una de las más importantes en profundidad. Estas clases se van a ver ordenadas siguiendo el orden mostrado en las ilustraciones.

Tabla 4: Detalle de clases importantes

Clase	Descripción
BaseAction	Es la clase de la que heredan el resto de las acciones, se trata de una clase abstracta que representa cada una de las acciones que tiene disponible una unidad, se encarga también de implementar el algoritmo de Montecarlo que simula los escenarios posibles para seleccionar la acción a ejecutar por la IA contrincante.
HealAction	Esta clase se encarga de ejecutar la acción de curar a un aliado, hereda diferentes métodos de su clase principal.
RangeAction	Esta clase se encarga de ejecutar la acción de realizar un ataque a distancia, hereda varios métodos de su clase principal
MeleeAction	Esta clase se encarga de ejecutar los ataques cuerpo a cuerpo de la unidad, hereda varios métodos de su clase principal.
MoveAction	Esta clase se encarga de mover a la unidad por el mapa, siguiendo un camino de celdas.
FireballAction	Esta clase instancia otro objeto llamado Fireball que realizará daño dentro de un área de explosión.
UnitAnimator	En esta clase se implementa todo lo referente a las animaciones que puede ejecutar la unidad
Unit	Esta clase es la principal que designa un GameObject como una unidad, permite la creación de las estadísticas del personaje, cargar el sistema de vida de la unidad, etc.
GridSystem	Esta clase contiene el sistema de celdas por las que se moverán las unidades, así mismo se encarga de comprobar si esta esta ocupada por un obstáculo o no.
Pathfinding	Es la clase encargada de implementar el algoritmo A*, en esta clase se calcula el camino más corto que puede recorrer una unidad, comprobar si existen obstáculos o si la casilla ya está ocupada.
LevelGrid	Esta clase se encarga de generar las casillas por las que se moverán las unidades, así mismo se encarga de comprobar si esta está ocupada por un obstáculo o no.

EnemyAI	Esta clase implementa el cerebro de la IA contrincante, se encarga de llamar al algoritmo de aprendizaje reforzado para poder emplear la mejor acción en el estado que se encuentra la unidad.
UnitManager	Se encarga de controlar cuantas unidades existe en la escena, diferenciando también entre enemigas y las del jugador.
TurnSystem	Se encarga de controlar los turnos entre el jugador y la IA contrincante.
SceneSelector	Se encarga de cambiar entre escenas.
MouseWorld	Se encarga de controlar todo lo referente a la posición del ratón dentro de las escenas que permiten interactuar con el entorno.
CameraController	Se encarga de controlar todo lo referente a la cámara, tanto el desplazamiento como la rotación el zoom.
UnitRagdoll	Se encarga de controlar el "Ragdoll" de la unidad que se crea cuando la vida de esta llega a cero.
HealthSystem	Se encarga de controlar la vida que dispone la unidad en cada momento.
GameManager	Se encarga de controlar el estado del juego y de guiar los diferentes componentes que lo conforman.

En cuanto al diseño procedimental, fue definido en la iteración inicial para que posteriormente a medida que iban pasando las demás iteraciones se fueran refinando como se ha ido explicando a lo largo de esta sección.

En cuanto al diseño arquitectónico, se ha definido mediante el diagrama de clases del diseño que, aunque se va a mostrar a continuación, para consultarlo en mayor profundidad se puede consultar el Anexo III.

En cuanto a la arquitectura a alto nivel se puede plantear en una etapa ya que todo nuestro proyecto se realiza usando el motor Unity Engine que se encarga de englobar todo lo necesario para la realización del proyecto.

5.1.4.- Implementación

Para la implementación de este proyecto se va a seguir las siguientes directrices:

- **Configuración del proyecto en Unity:** Se crea un nuevo proyecto en Unity y se configura para que se ajuste a las necesidades de nuestro proyecto.
- **Creación de assets:** Los assets son los elementos más importantes de todo proyecto de Unity, en este caso se han creado propios como todos los scripts, se han creado algunos sprites como las interfaces. También se han importado assets descargados tanto de la "Assets Store de Unity", como de otros propietarios y se detallaran a continuación organizados entre objetos para construir el mundo, efectos especiales, personajes y música:
    - **Polygon Dungeon Realms** (Synty Studios, 2020)
    - **Free RPG Weapons** (GAPH, 2017)
    - **Fantasy Cursors Pack** (HAIKU, 2022)
    - **Medieval Village Building Pack** (DHSFX, 2020)
    - **Polygon Fantasy Characters** (Synty Studios, 2020)
    - **Polygon Fantasy Hero Characters** (Synty Studios, 2020)
    - **Viverna assets** (Viverna, s.f.)
    - **Polygon Prototipe** (Synty Studios, 2020)
    - **52 Special Effect Pack** (GAPH, 2017)
    - **Fantasy Videogame Music** (HAIKU, 2022)
    - **Gami Fantasy Music** (ToxicDelta, 2017)
    - **Animaciones** (Mixamo, s.f.)
- **Construcción de escenas y niveles:** Usando los assets anteriores se crean los escenarios que se mostrara en las siguientes ilustraciones:

- **Programación:** Unity utiliza principalmente C# para la programación. Por lo que toda la lógica y scripts estarán escritos en este lenguaje, el cual es necesario para que algunos de los assets anteriores puedan moverse o interactuar con otras partes del entorno o mostrar parte de la Interfaz de Usuario.

5.1.5.- Pruebas

Para comprobar el funcionamiento del proyecto se han llevado a cabo varios tipos de pruebas que se van a exponer a continuación.

Tratándose de un juego TBRPG se han seguido las siguientes pruebas unitarias para el desarrollo de videojuegos las cuales se van a dividir en las siguientes pruebas (Schultz, Bryant, & Langdell, 2005):

Funcionalidad de turnos: Prueba para asegurar que los turnos cambien correctamente entre jugadores y enemigos. Cada unidad en el juego debe recibir su turno en el orden correcto.

1. Prueba de rotación de turnos: Una vez que un jugador ha movido todas sus unidades y ha terminado su turno, la transición al siguiente jugador o a la IA debe ser fluida y clara. Realiza pruebas para asegurarte de que esta transición de turno funcione correctamente.
2. Prueba de acciones en turnos: Asegurarse de que cada unidad pueda realizar todas sus acciones posibles durante su turno. Esto puede incluir moverse, atacar, usar habilidades, etc. Una vez que una unidad ha realizado todas sus acciones, no debe ser capaz de actuar de nuevo hasta que llegue el próximo turno del jugador.
3. Prueba de fin de ronda: Verificar que el inicio de una nueva ronda ocurra correctamente después de que todos los jugadores y la IA hayan tenido su turno. Además, asegúrate de que todas las unidades del jugador recuperen la capacidad de actuar al comienzo de una nueva ronda.

Pruebas de interacción de unidades: Como se trata de un TBRPG cada unidad posee una variedad de acciones únicas que pueden emplearse en diferentes contextos.

1. Pruebas de interacciones de ataque y daño: La correcta asignación y cálculo de daño es algo crítico. Se realizar pruebas para verificar que efectivamente los ataques resulten en el daño esperado o pueden fallar por las estadísticas de la unidad a la que está atacando.
2. Pruebas de efectos de estado: En este caso en concreto solo se tiene en cuenta las habilidades de curación de algunas unidades, para ello se realizan pruebas sobre unidades dañadas para comprobar su restaura el daño.

Pruebas de movimiento de la unidad: Las pruebas que verifiquen que cada unidad pueda moverse la cantidad correcta de espacios y solo a través de terrenos permitidos.

1. Prueba de iteraciones de unidades y obstáculos: Se verifica que las unidades no atraviesen los obstáculos y otras unidades dentro del escenario de juego.
2. Prueba de limitaciones de movimiento: Cada unidad tiene un número limitado de espacios que puede moverse, estas pruebas sirven para que una unidad se mueva una cantidad de espacio limitada ni más ni menos.
3. Pruebas de restricciones de terreno: Esta prueba consiste en que las unidades no salgan del escenario de juego limitado por las celdas.

Pruebas de AI (Inteligencia Artificial): La mayoría de los TBRPG tienen AI para controlar a los enemigos. Las pruebas unitarias pueden verificar que la AI tome decisiones estratégicas razonables y siga las reglas del juego (Rabin, 2017).

1. Prueba de toma de decisiones estratégicas de la IA: Esta prueba se desarrolló para evaluar la capacidad del sistema de inteligencia artificial para tomar decisiones complejas dentro del entorno del juego. Esta prueba implicó poner a la IA en una serie de situaciones estratégicas, cada una con sus propios desafíos y consideraciones únicas.

   El propósito de esta prueba era no sólo evaluar el rendimiento de la IA en la toma de decisiones estratégicas, sino también mejorar y afinar los algoritmos de

aprendizaje reforzado utilizados en el proyecto. Los resultados de estas pruebas contribuyeron significativamente a la iteración y mejora de la IA, lo que finalmente resultó en una IA más robusta y efectiva en la toma de decisiones estratégicas en el juego.

2. Prueba de seguimiento de las reglas del juego por parte de la IA: se llevó a cabo para verificar si el sistema de inteligencia artificial era capaz de adherirse a las reglas y restricciones del juego sin violarlas..

   Cada escenario se configuró de tal manera que presentaba un desafío único para la IA, en el sentido de que se requería una estrategia y toma de decisiones cuidadosas para ganar mientras se adhería a las reglas del juego. En ciertos casos, las situaciones eran tales que violar una regla podría parecer el camino más fácil hacia el éxito.

   El desempeño de la IA se evaluó en términos de su capacidad para navegar por estas situaciones sin infringir ninguna de las reglas y restricciones del juego.

Pruebas de interfaz de usuario (UI): En los TBRPG, los jugadores a menudo necesitan mucha información para tomar decisiones. Las pruebas unitarias pueden verificar que la UI muestre la información correcta en el momento adecuado.

1. Prueba de visualización de información: Se asegura de que la UI muestra toda la información correctamente y en el formato claro y comprensible posible.
2. Prueba de actualización en tiempo real de la UI: Asegura de que la UI se actualice en tiempo real para reflejar los cambios en el estado del juego. Esto puede incluir el cambio de las estadísticas de las unidades debido al daño recibido, el uso de habilidades, efectos de estado, etc. También debe mostrar correctamente el cambio de turnos y cualquier otra información dinámica.
3. Prueba de interactividad de la UI: La UI debe permitir a los jugadores interactuar con el juego de la manera esperada. Esto puede incluir seleccionar unidades, moverlas, seleccionar y usar habilidades, usar objetos, etc. Debes realizar pruebas para verificar que todos estos elementos de interactividad funcionen correctamente.

4. Prueba de accesibilidad de la UI: Esta prueba segura de que la UI sea accesible para todos los jugadores. Esto puede incluir ajustar el tamaño del texto, proporcionar opciones de contraste de color, incluir subtítulos para cualquier audio, entre otros. Las pruebas de accesibilidad son vitales para garantizar que tu juego pueda ser disfrutado por la mayor cantidad de jugadores posible.

Pruebas de estado de guardado/carga: Dado que los TBRPG suelen ser juegos largos que los jugadores no pueden terminar en una sola sesión, es importante probar la funcionalidad de guardar y cargar juegos.

1. Prueba de funcionalidad de guardado: Se realizan pruebas para asegurarse de que la función de guardado capture todo el estado del juego, incluyendo la posición y estadísticas de las unidades, etc.
2. Prueba de funcionalidad de carga: Al cargar un juego guardado, el estado del juego debe restablecerse exactamente a cómo estaba cuando se guardó. Realiza pruebas para verificar que todos los aspectos del estado del juego se carguen correctamente.
3. Prueba de integridad de datos de guardado/carga: Asegura que los datos de guardado no se corrompan fácilmente y que el juego pueda manejar correctamente la carga de datos de guardados corruptos, ya sea recuperándolos parcialmente o informando al jugador del problema.

5.2.- Secuencia de los casos de uso

Se va a presentar el diseño de la programación que se corresponde con el análisis de la aplicación exponiendo el funcionamiento del sistema a un alto nivel.

En la siguiente ilustración

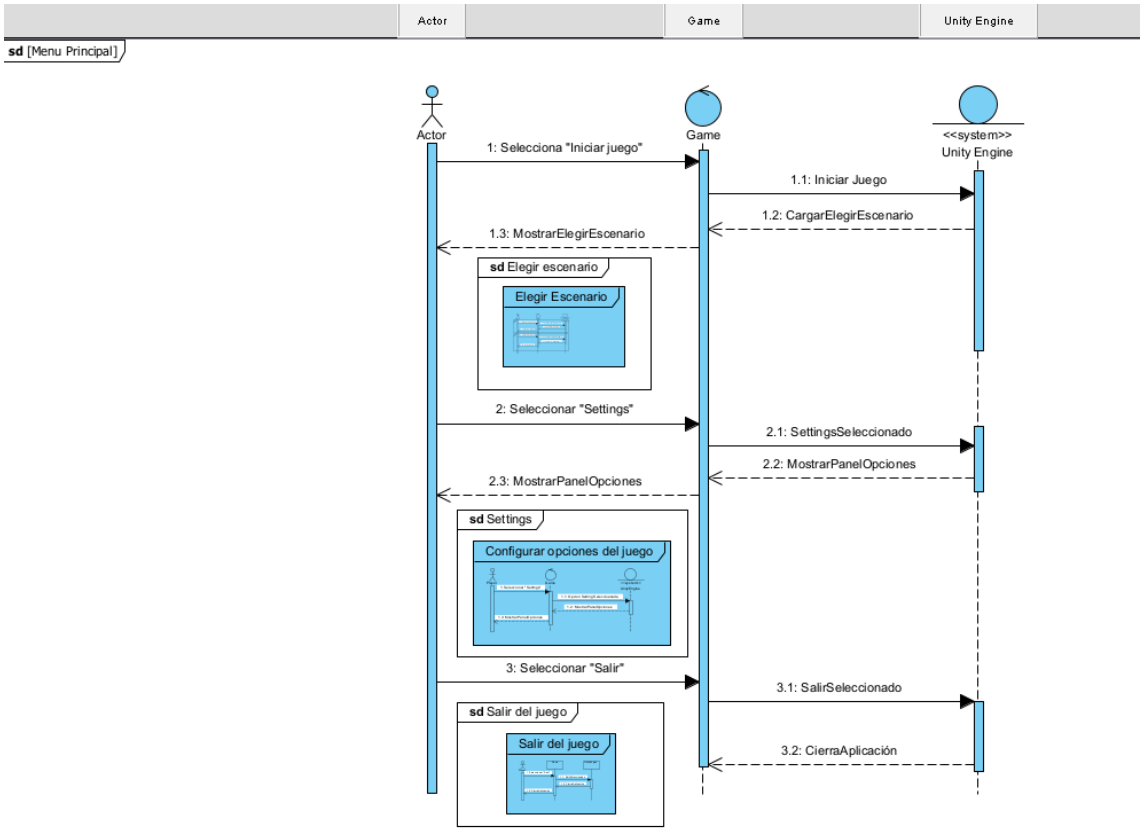


Ilustración 20: Diagrama de análisis del caso de uso (I)

En la ilustración 20 se muestra el primer diagrama de análisis donde se ve como el jugador puede iniciar seleccionar diferentes acciones, entre ellas se encuentran iniciar juego, ver el menú de opciones o salir del juego.

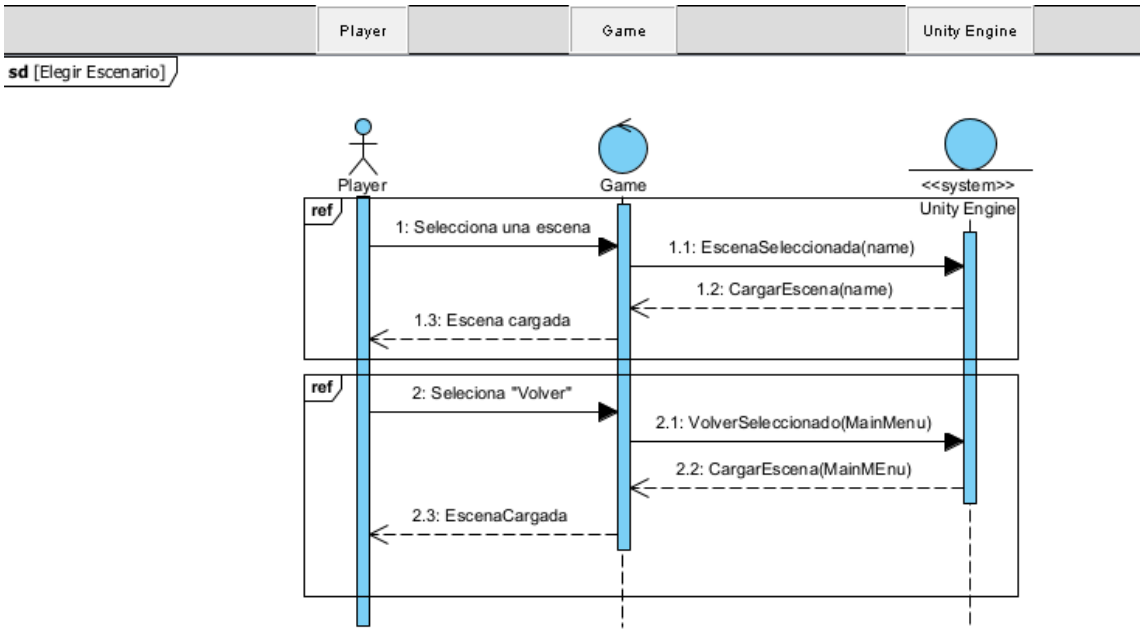


Ilustración 21: Diagrama de análisis del caso de uso (II)

Una vez seleccionado en el diagrama del caso anterior "iniciar juego" se nos muestra la escena "elegir escenario" en este caso podemos elegir los escenarios jugables a través de un botón, también se puede volver atrás pulsando el botón "volver".

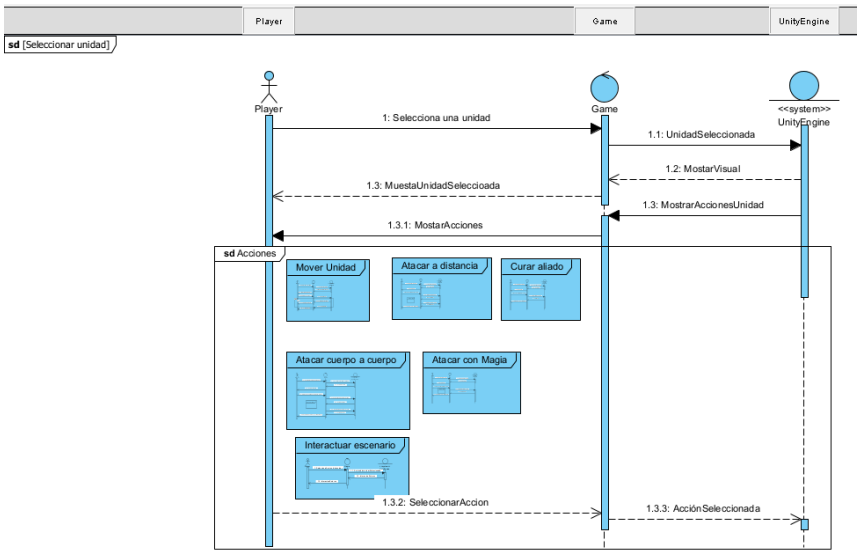


Ilustración 22: Diagrama de análisis del caso de uso (III)

Para elegir una unidad solo tenemos que seleccionar con el ratón, pasamos por encima del collider de la unidad y una vez seleccionada se nos mostraran las acciones que esta unidad posee.

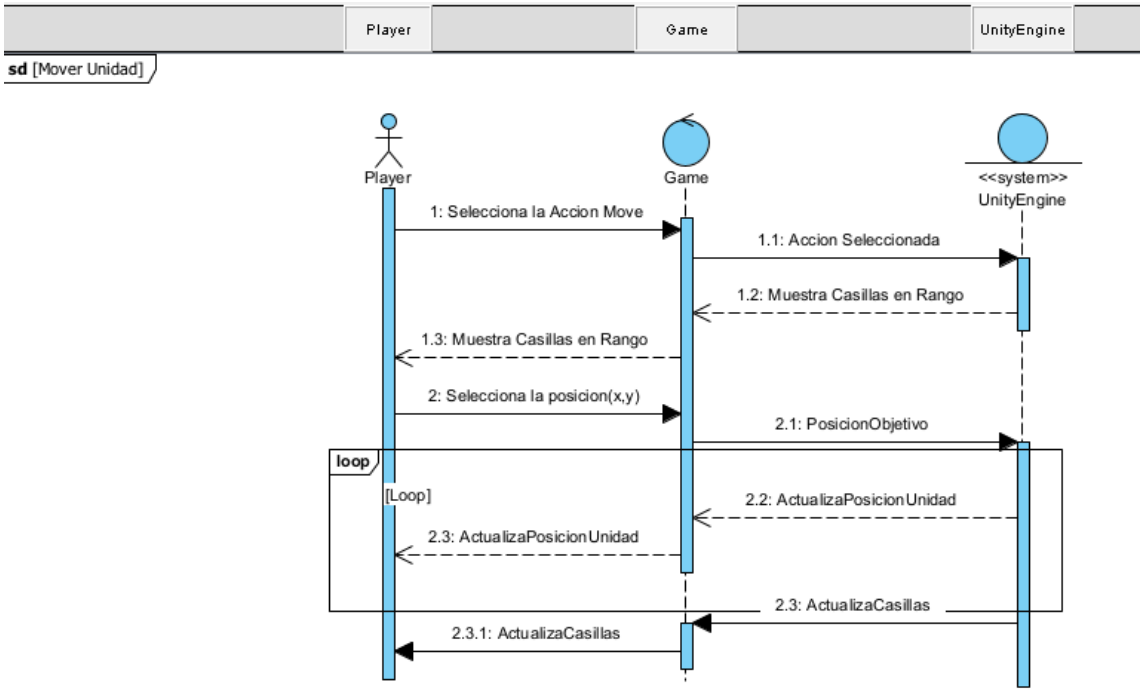


Ilustración 23: Diagrama de análisis del caso de uso (IV)

Para mover a la unidad el jugador tiene que seleccionar la acción mover, con la cual nos mostrará un área en color verde donde la unidad se puede mover, luego selecciona una casilla dentro de esa área y la unidad ira actualizando la posición en cada casilla que pise hasta llegar al objetivo marcado.

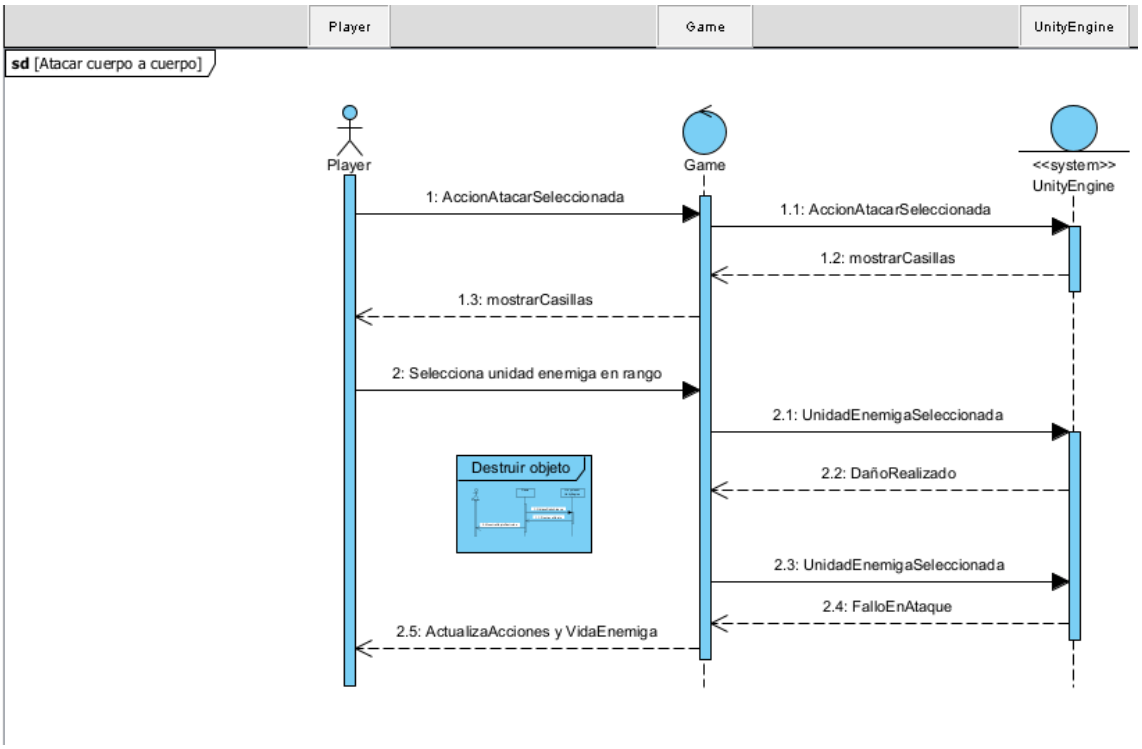


Ilustración 24: Diagrama de análisis del caso de uso (V)

Para realizar un ataque cuerpo a cuerpo la unidad debe tener una unidad enemiga en su rango para poder realizar la acción. Si tenemos una unidad o objeto destruible dentro de nuestro rango de acción, podemos realizar el ataque dañando al objetivo.

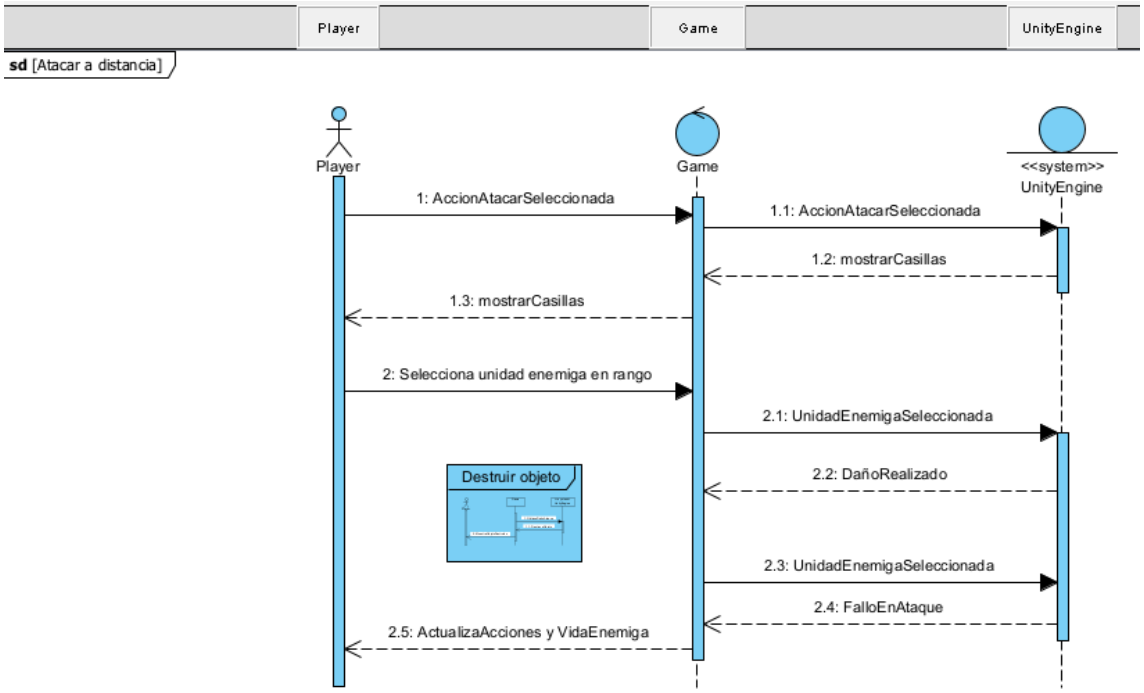


Ilustración 25: Diagrama de análisis del caso de uso (VI)

En este caso de uso sigue el mismo patrón que el caso de uso anterior, solo podemos realizar un ataque a distancia si el enemigo o un objeto destruible se encuentra dentro de nuestro rango de acción.

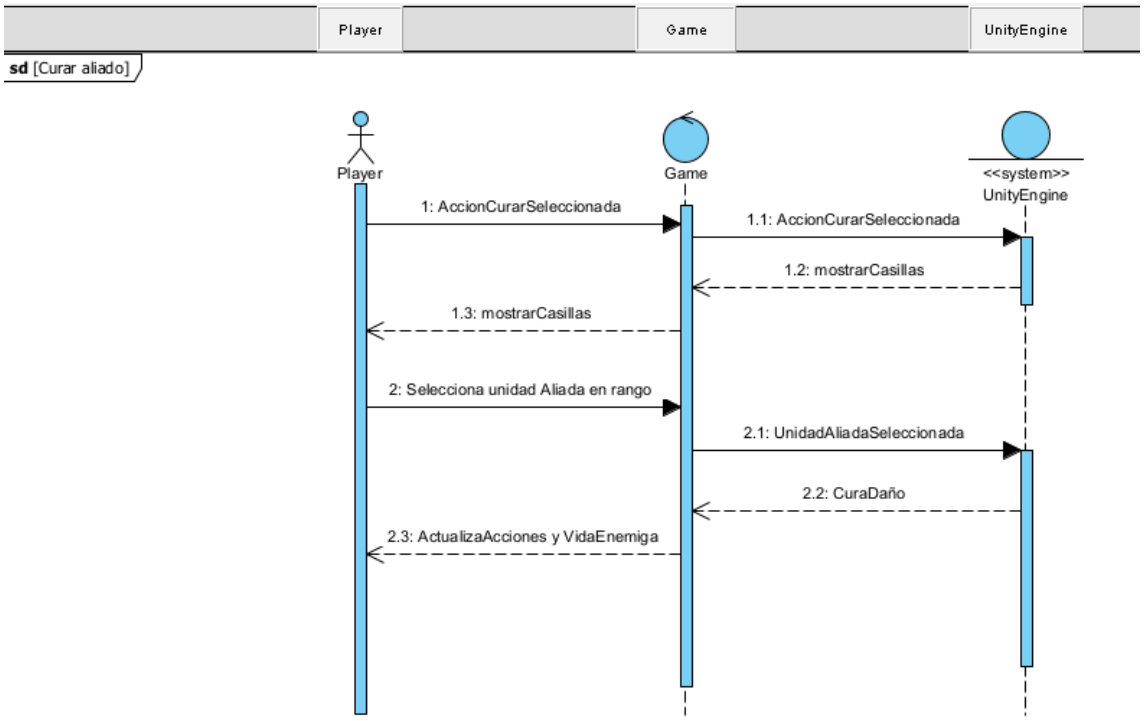


Ilustración 26: Diagrama de análisis del caso de uso (VII)

Este caso de uso de la ilustración 26 nos muestra la secuencia de como curar a un aliado, sigue el mismo patrón que el caso de uso anterior de atacar solo que cambiando el objetivo disponible por los aliados.

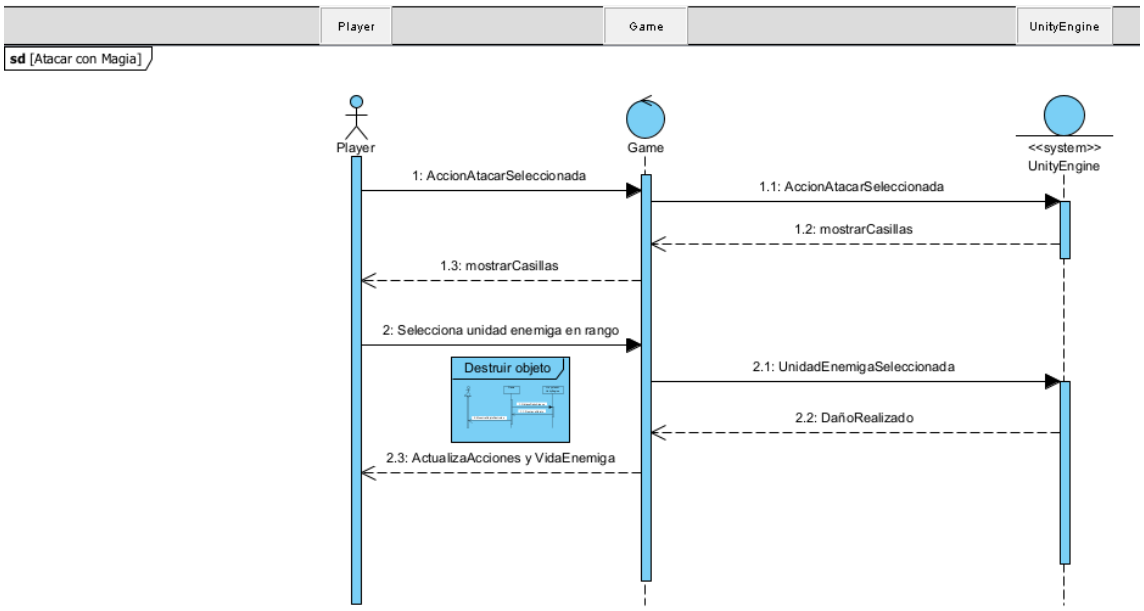


Ilustración 27: Diagrama de análisis del caso de uso (VIII)

En el caso de uso "Atacar con magia" que se muestra en la ilustración 27 se puede ver que sigue la misma secuencia de los casos de uso de ataque anteriores, además este caso tiene como efecto generar daño en un área, la cual puede dañar a las unidades aliadas.

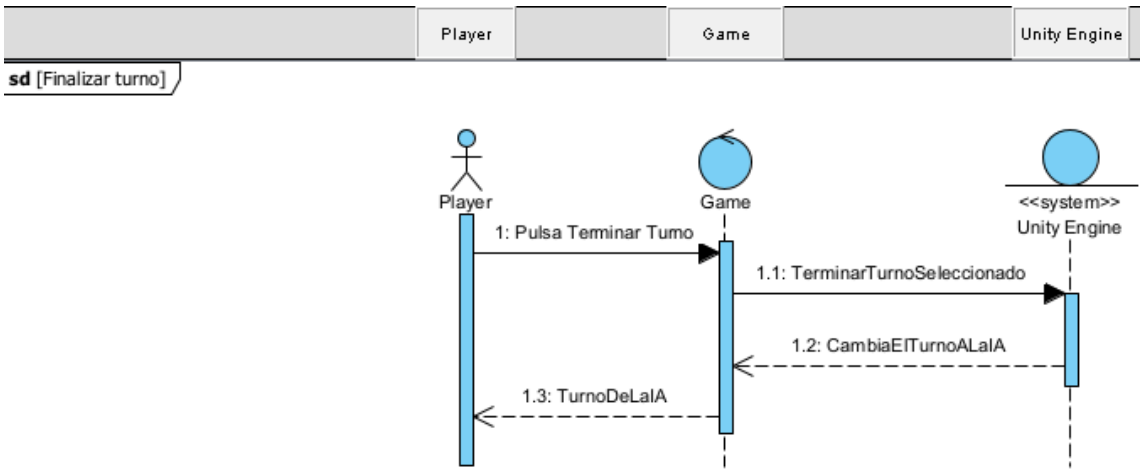


Ilustración 28: Diagrama de análisis del caso de uso (IX))

En la ilustración 28 se muestra como pasar turno, existen diferentes maneras de pasar turno, una es pulsando sobre el botón terminar turno, otra es pulsando sobre la tecla espacio y la última y la que usara la IA es que todas las unidades terminen sus puntos de acción.

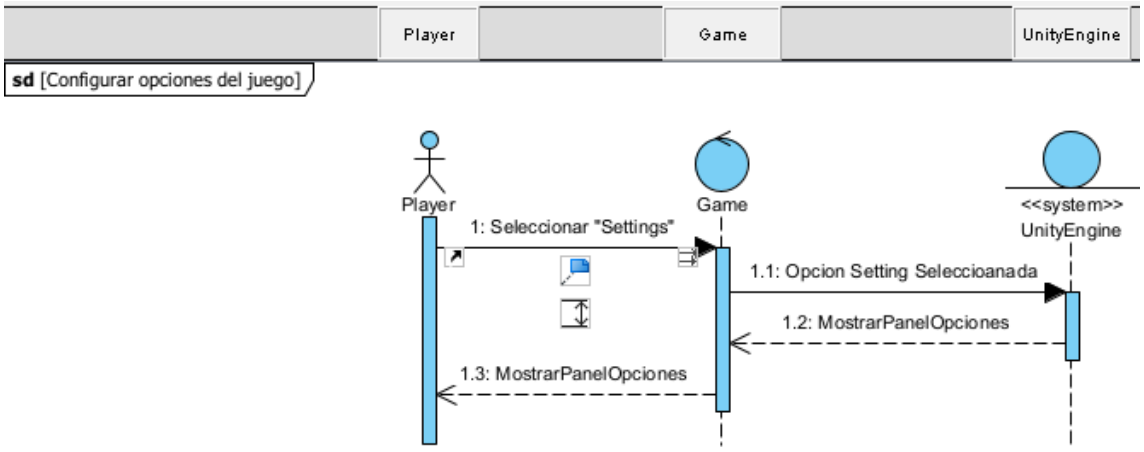


Ilustración 29: Diagrama de análisis del caso de uso (X)

En la ilustración 20 se muestra cómo podemos acceder desde el menú de inicio a las opciones que puede modificar el jugador, la secuencia seria seleccionar el botón “Settings” dentro del menú de inicio, aunque también podemos encontrarlo dentro del menú de pausa durante el juego.

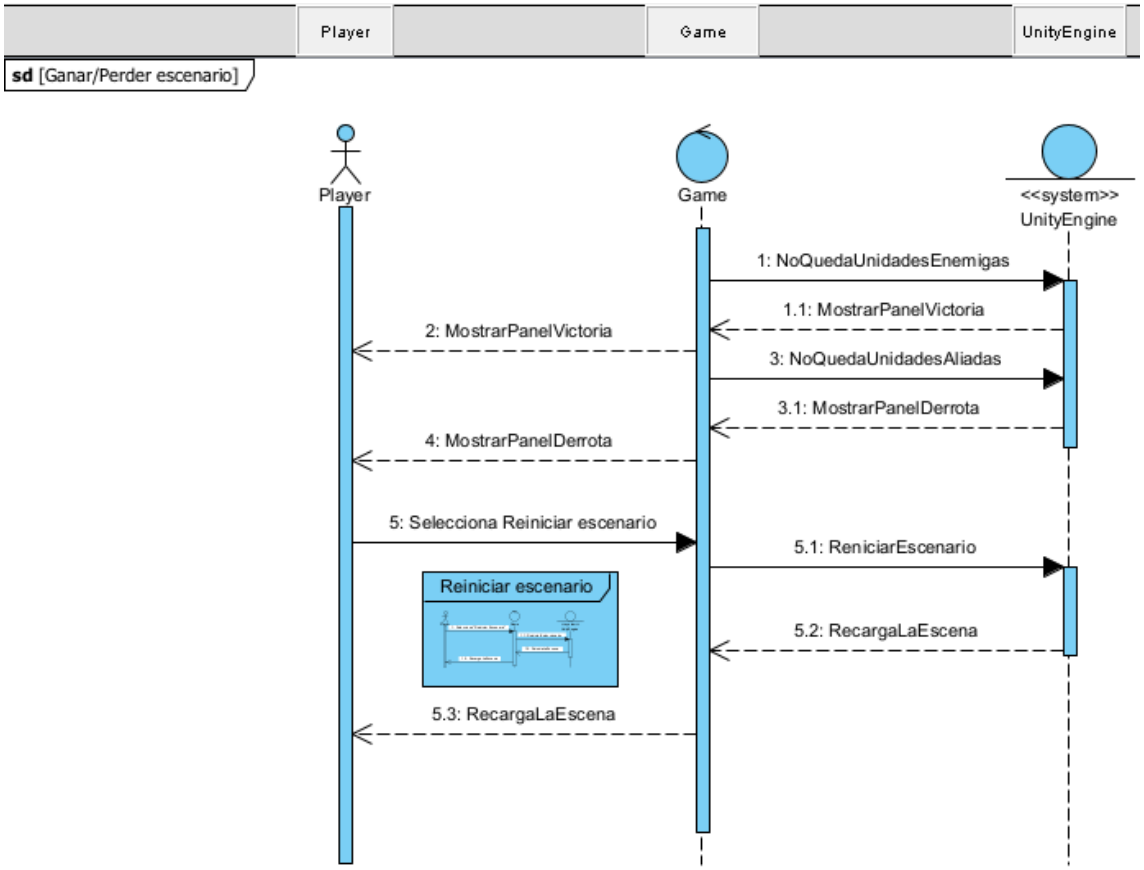


Ilustración 30: Diagrama de análisis del caso de uso (XI)

La secuencia que sigue este caso de uso se ejecuta cuando dentro de la lista de unidades enemigas o aliadas no queda ningún elemento, mostrando este por pantalla el mensaje de victoria o derrota, además de las opciones de recargar el escenario o de volver al menú de inicio.

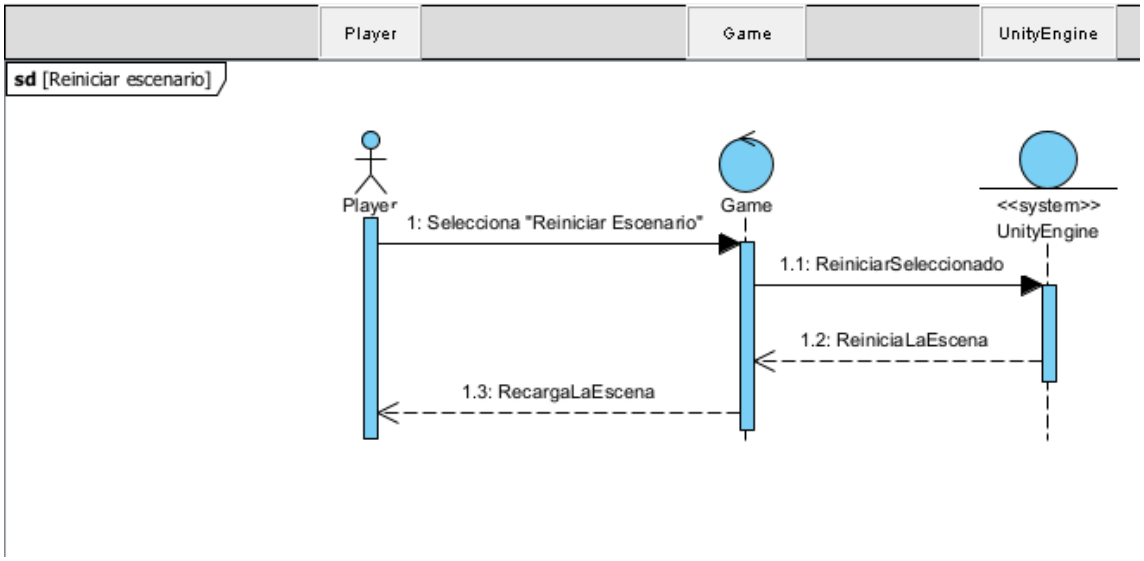


Ilustración 31: Diagrama de análisis del caso de uso (XII)

Esta opción se nos muestra cuando ganamos o perdemos una partida, si pulsamos el botón podemos se recargará el escenario a su etapa inicial.

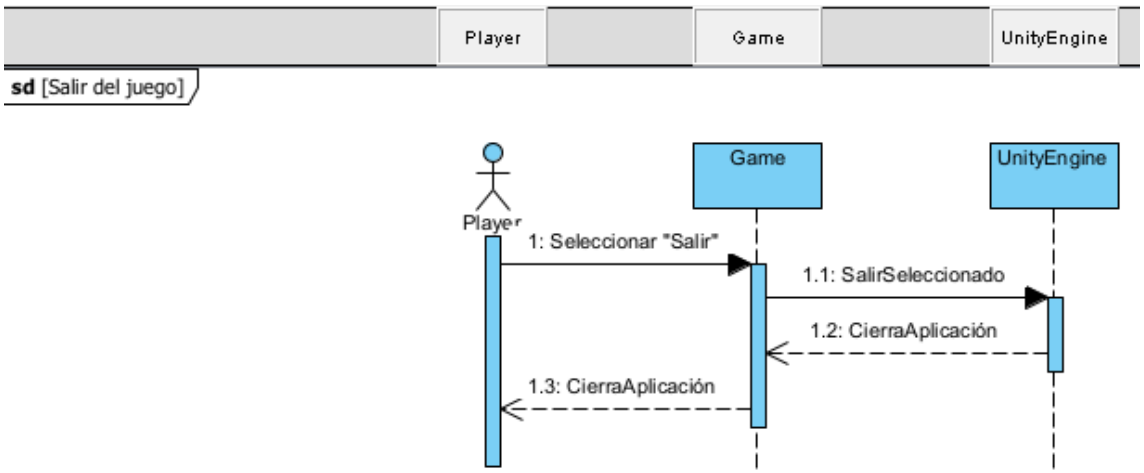


Ilustración 32: Diagrama de análisis del caso de uso (XIII)

Cuando seleccionamos el botón de salir, el juego se cerrará automáticamente, podremos acceder a este caso de uso tanto desde el menú de inicio como desde el menú de pausa.

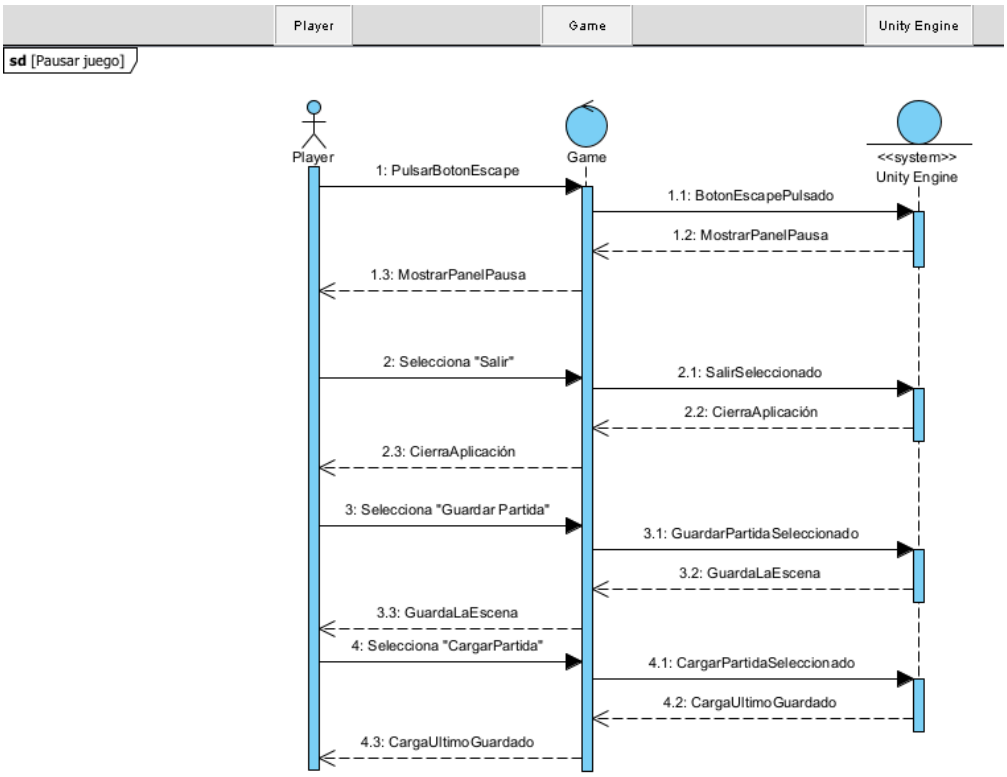


Ilustración 33: Diagrama de análisis del caso de uso (XIV)

Para ejecutar el caso de uso de la ilustración 33, durante nuestra partida solo tendremos que pulsar sobre la tecla “escape” se nos desplegara los casos de uso como salir, guardar, cargar o seleccionar las opciones.

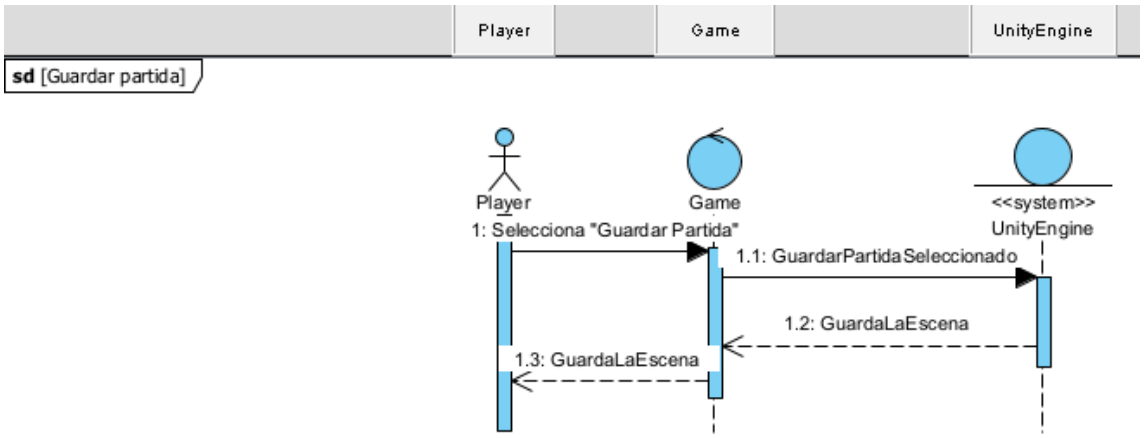


Ilustración 34: Diagrama de análisis del caso de uso (XV)

En este caso de uso si hemos activado el menú de pausa dentro del juego podemos seleccionar la opción guardar, que guardara el estado actual de la partida en un archivo binario.

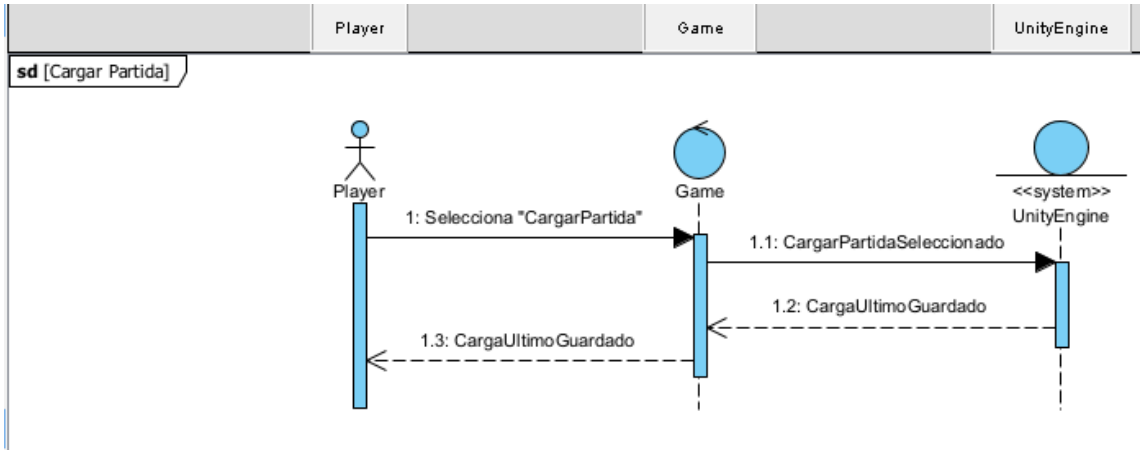


Ilustración 35: Diagrama de análisis del caso de uso (XVI)

En este caso de uso podemos acceder desde la pantalla principal, cojera el último archivo guardado y cargara la última escena en el estado que fue guardada.

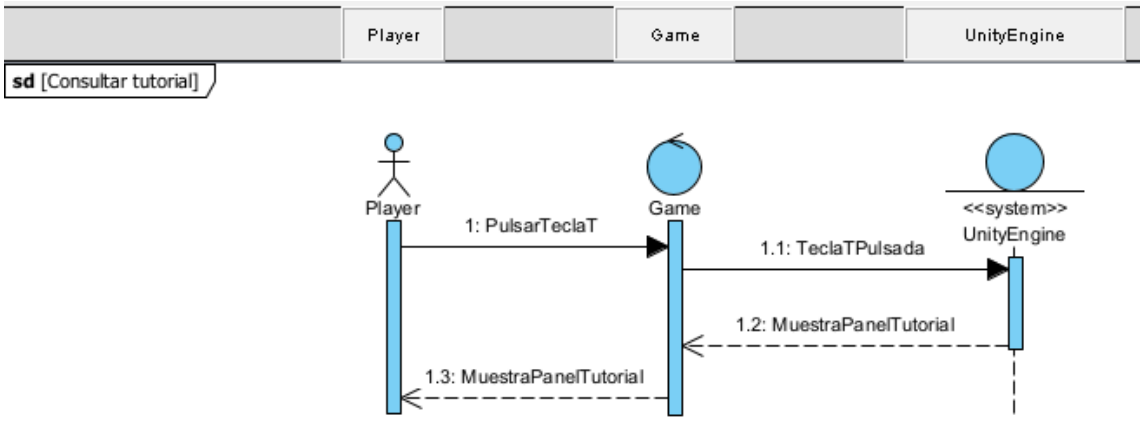


Ilustración 36: Diagrama de análisis del caso de uso (XVII)

Durante el juego, si pulsamos T, nos mostrara el tutorial del juego.

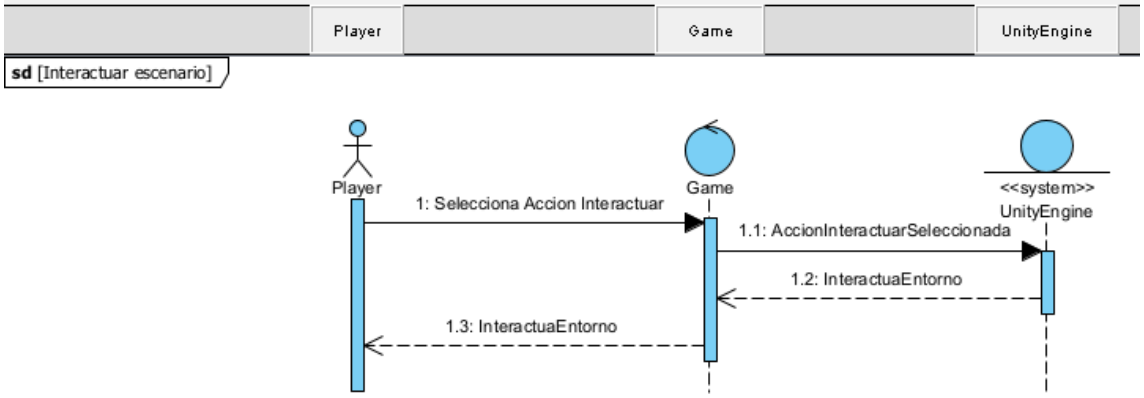


Ilustración 37: Diagrama de análisis del caso de uso (XVIII)

Cuando hemos seleccionado una unidad, podemos elegir la acción interactuar, si un objeto dentro del radio de acción de esta unidad se encuentra un objeto interactúale, gastara un punto de acción e interactuara con el objeto.

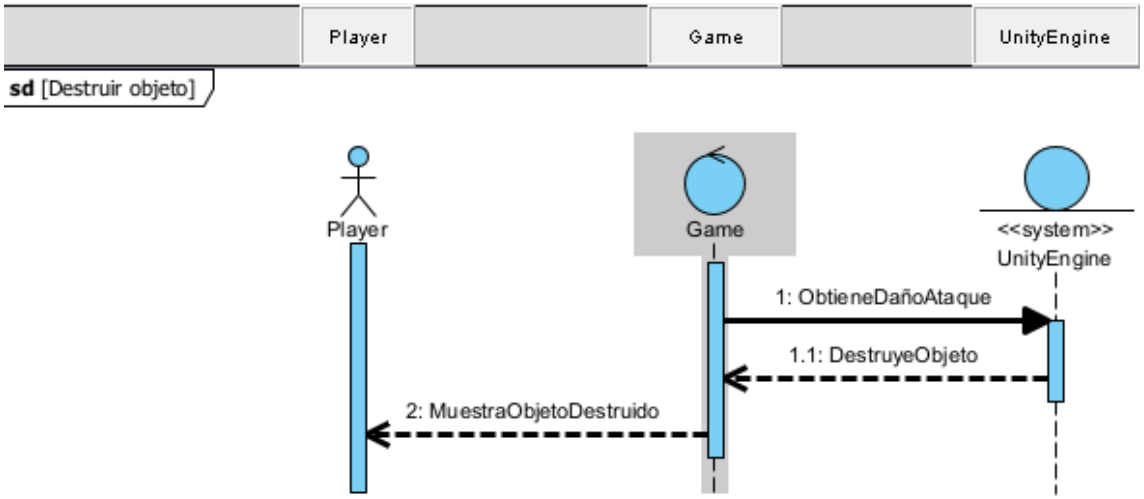


Ilustración 38: Diagrama de análisis del caso de uso (XIX)

Por último, tenemos este caso de uso “destruir objeto” en el que si con las acciones que permiten realizar daño, provocan el suficiente en un objeto destruible, este quedara eliminado y desaparecerá de la pantalla.

5.3.- Aspectos relativos al despliegue

A continuación, se va a poder apreciar la complejidad del despliegue del sistema. A parte de mostrar el diagrama, se va a describir brevemente el despliegue del proyecto:

- Usando el motor grafico de Unity para ejecutar el juego y compilarlo.
- Dependiendo del dispositivo donde se despliegue usara un entorno grafico diferente siendo en Windows donde existen dos entornos de ejecución dependiendo de la versión que tengamos instalada, aunque también se puede admitir la de Vulkan
- En MacOS se usa el entorno de ejecución propio llamado Metal.

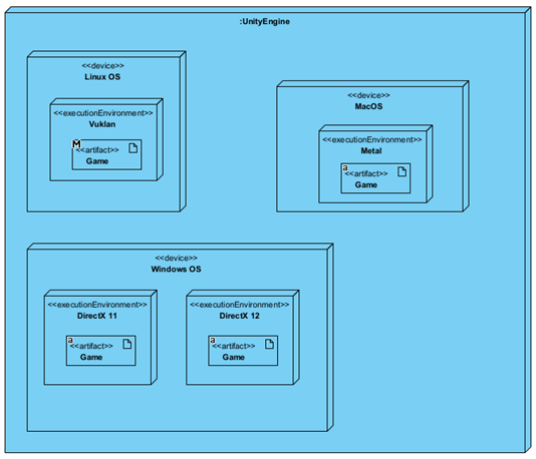


Ilustración 39: Diagrama de despliegue

5.4.- Diseño de escenarios

Ilustración 40: Menú principal

La ilustración 40, se nos muestra el menú principal con el nombre en clave del proyecto. Este nombre es un identificador único que distingue este proyecto de otros, reflejando su identidad y singularidad.

En la ilustración, también vemos distintos botones interactivos que facilitan la navegación y proporcionan un control completo a los usuarios. Cada uno de estos botones tiene una función específica y esencial para la interacción con la interfaz:

1. Botón 'Play': Este es posiblemente el botón más atractivo para los usuarios. Al hacer clic en él, los usuarios comienzan a interactuar con el contenido principal del proyecto. Actúa como la puerta de entrada a la experiencia que ofrece el proyecto. Una vez que se pulsa, el usuario se sumerge en el mundo que el proyecto ha creado para él.
2. Botón 'Configuraciones': Este botón es crucial para personalizar la experiencia del usuario. Al hacer clic en él, se abre una variedad de opciones y ajustes que el usuario puede modificar según sus preferencias. Estos ajustes pueden abarcar

desde la calidad gráfica hasta los controles de sonido, pasando por las opciones de control y muchas otras características que hacen que cada experiencia sea única para el usuario.

3. Botón 'Salir': Aunque puede parecer menos emocionante, el botón 'Salir' es igualmente importante. Proporciona a los usuarios el control de cerrar el proyecto cuando lo deseen. Asegura que la interacción con el proyecto es a voluntad del usuario, dándole la libertad de entrar y salir a su conveniencia.

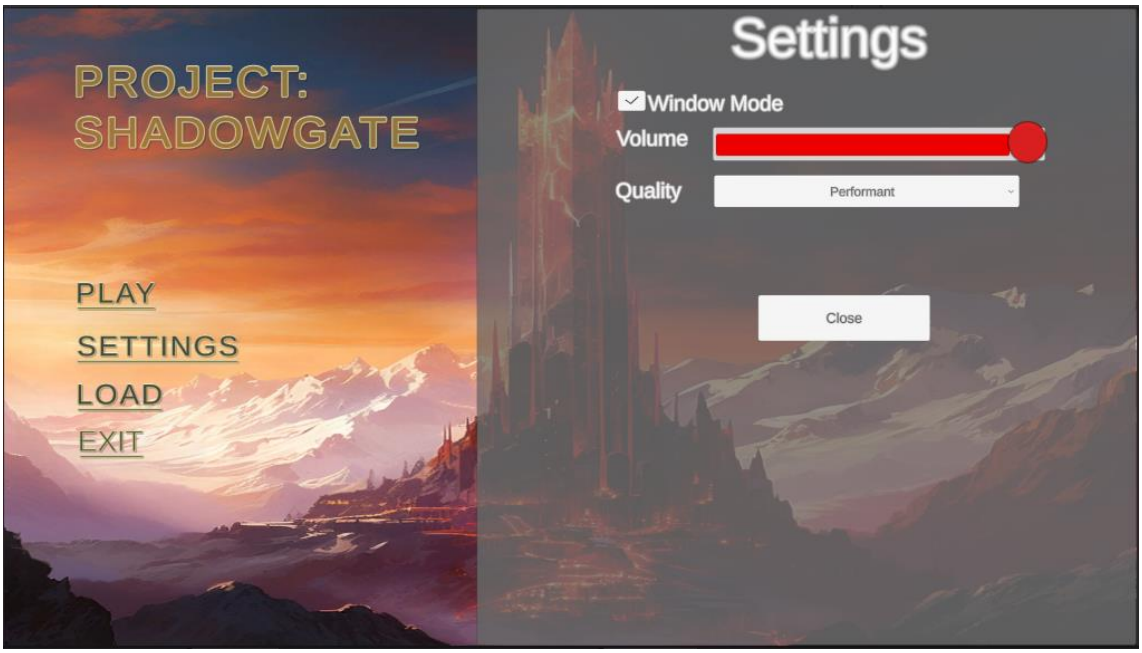


Ilustración 41: Panel de opciones

En la ilustración 41 se muestra el panel de opciones en el cual se muestran las diferentes opciones. Entre las opciones que el jugador puede modificar se encuentran la opción de jugar a pantalla completa o en modo ventana, el volumen de la música o la calidad de imagen del juego, que puede ser baja, media o alta dependiendo de cómo se quiera apreciar.

Ilustración 42: Mapa de escenarios

La Ilustración 41 presenta un fascinante "Mapa de Escenarios", un elemento crucial de la interfaz de usuario que permite a los jugadores una navegación intuitiva y una elección informada sobre su próximo paso en el juego.

El Mapa de Escenarios está diseñado inteligentemente con distintos escenarios, cada uno representado como un botón interactivo.

En la parte superior de este mapa, se encuentra un botón denominado "Volver al menú principal". Este botón es esencial en términos de la experiencia de navegación del usuario. Proporciona a los jugadores la capacidad de retroceder al menú principal, en lugar de sentirse atrapados en una selección que tal vez quisieran reconsiderar. Este botón de añade un nivel de flexibilidad y comodidad para el jugador, lo que resulta en una experiencia de usuario más fluida y amigable.

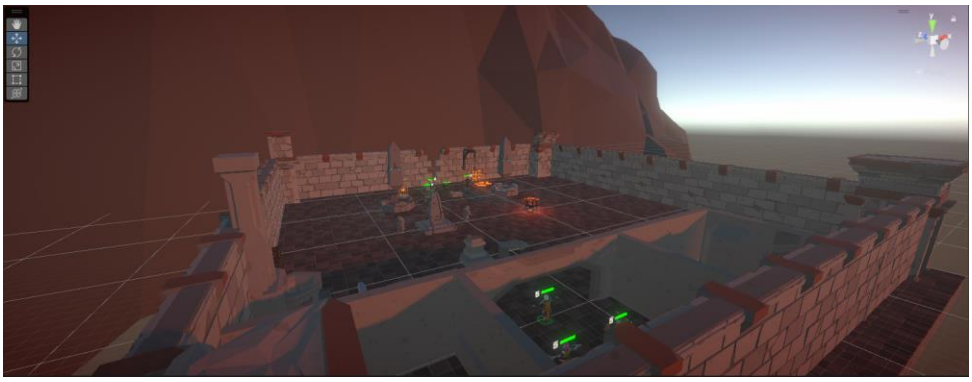


Ilustración 43: Escenario(I) The Walls

La Ilustración 42 introduce un escenario particularmente intrigante denominado "The Walls" (Las murallas). Este escenario se muestra de manera detallada, ofreciendo un vistazo de la ambientación y las características únicas que posee.

"The Walls" es un escenario diseñado con una combinación de elementos tanto naturales como artificiales, lo que proporciona un desafío fascinante y equilibrado para los jugadores.

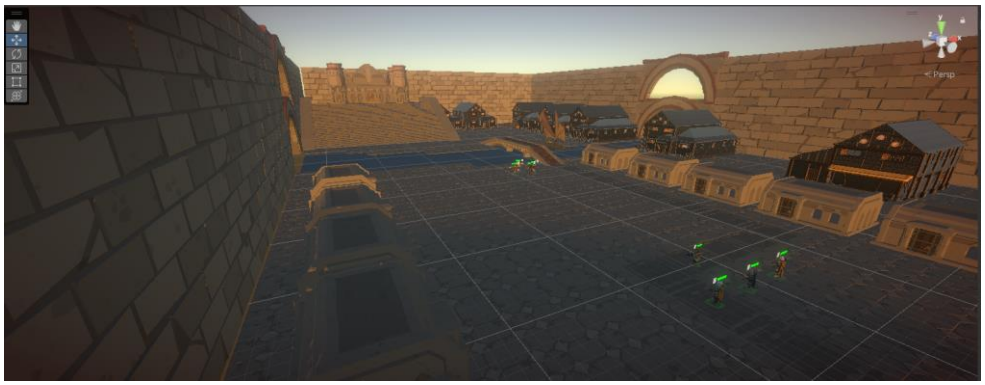


Ilustración 44: Escenario (II) The Town

La Ilustración 43 nos presenta un nuevo escenario titulado "The Town" (La Ciudad). Este dibujo describe un entorno urbano, proporcionando una visión distinta a los jugadores y una experiencia de juego diferente a la del escenario "The Walls".

"The Town" es un escenario que simula un ambiente de ciudad, repleto de edificios, y otros elementos arquitectónicos.

Ilustración 45: Menú de Pausa

En el menú de pausa que se muestra en la ilustración 45, podremos acceder a las diferentes opciones dentro del juego, como son continuar con la partida, guardar la partida, acceder al menú de opciones, volver al menú principal o salir del juego..

Ilustración 46: Menú Victoria/Derrota

Si todas nuestras unidades caen derrotadas o por el contrario dejamos al contrincante sin unidades se mostrará la ilustración 46 donde lo único que cambiara es el mensaje que nos mostrara siendo victoria o derrota.

6.- Resultados

En este apartado se van a exponer los diferentes resultados que se han obtenido de la realización del proyecto y las posibles líneas de trabajo futuro que se pueden vislumbrar tras la finalización de este.

6.1.- Resultados

Tras una investigación preliminar en la que se exploraron varios sistemas existentes que realizan modelos de dominio similares al proyecto que se planea desarrollar, hemos identificado varios objetivos clave que debemos abordar: la gestión de jugadores, la gestión de la progresión, la gestión de la IA, la simulación de escenarios, el almacenamiento de datos y, por último, el objetivo principal del proyecto, la implementación de la IA y el aprendizaje reforzado en un juego de rol táctico basado en turnos.

A partir de la investigación y de los resultados obtenidos de la misma, se ha concebido el proyecto "Plataforma para recreación de estrategia basada en aprendizaje reforzado". Este proyecto consiste en un sistema cuyo objetivo principal es la incorporación de técnicas de aprendizaje reforzado y algoritmos de IA sofisticados para mejorar la jugabilidad y la experiencia del jugador en los TBRPG. Una vez planteado el objetivo principal, lo hemos desglosado en un conjunto de objetivos secundarios que añaden una mayor funcionalidad al sistema de juego.

Para llevar a cabo el proyecto, ha sido necesario utilizar diversas tecnologías entre las que encontramos Unity para el desarrollo del juego, lenguajes de programación como C#, y algoritmos de IA avanzados como la búsqueda A*, el algoritmo de Montecarlo y el cálculo de la distancia de Manhattan o la distancia de Chevyshev.

Como resultado, hemos desarrollado un sistema de juego que permite monitorizar y ajustar la jugabilidad a través de algoritmos de IA avanzados. Estos algoritmos permiten a las unidades controladas por la IA tomar decisiones estratégicas en tiempo real, lo que resulta en una experiencia de juego más rica y desafiante para los jugadores.

Con todas estas características, consideramos que hemos cumplido el objetivo principal, ya que, con el sistema desarrollado, hemos conseguido mejorar la jugabilidad y la experiencia del jugador en los TBRPG a través de la aplicación de técnicas de aprendizaje reforzado y algoritmos de IA sofisticados.

Adicionalmente, consideramos que hemos cumplido los objetivos secundarios, ya que hemos creado una serie de funcionalidades adicionales que enriquecen la experiencia general del juego.

En cuanto a los requisitos no funcionales como rendimiento, escalabilidad, portabilidad, usabilidad, accesibilidad, estabilidad, personalización, estética y diseño, y mantenibilidad, gracias a la utilización de tecnologías y metodologías adecuadas, y al hecho de que el proyecto está bien modularizado y construido en Unity, consideramos que estos criterios también se han cumplido satisfactoriamente.

El rendimiento se manifiesta en la respuesta rápida y eficiente del juego ante las acciones del jugador, minimizando el tiempo de carga y maximizando la fluidez del juego. La escalabilidad se evidencia en la capacidad del juego de adaptarse y expandirse para incluir más niveles o características en futuras versiones.

La portabilidad se demuestra en el hecho de que el juego, al estar construido en Unity, puede ser fácilmente desplegado en múltiples plataformas. La usabilidad y la accesibilidad se han garantizado al proporcionar una interfaz de usuario intuitiva y amigable, con opciones para personalizar la experiencia del juego.

La estabilidad ha sido una prioridad durante todo el proceso de desarrollo, con pruebas exhaustivas para asegurar que el juego funcione correctamente en diversas condiciones y plataformas. La personalización, la estética y el diseño se han tenido en cuenta para ofrecer una experiencia de juego atractiva y agradable, con la posibilidad de ajustar la experiencia según las preferencias del jugador.

Finalmente, la mantenibilidad se ha logrado a través de un código bien estructurado y comentado, permitiendo futuras actualizaciones y correcciones de manera eficiente. En conjunto, estos elementos contribuyen a un proyecto de alta calidad que cumple con todos los criterios no funcionales establecidos.

6.2.- Líneas de trabajo futuras

Las siguientes líneas de trabajo se presentan como posibles expansiones futuras y mejoras del proyecto, todas las cuales ofrecen oportunidades emocionantes para enriquecer aún más el juego y su experiencia general:

1. **Implementación de más niveles:** Introducir más niveles podría añadir profundidad y variedad al juego. Cada nivel puede plantear desafíos distintos, permitiendo a los jugadores y a la IA enfrentar nuevas situaciones y estrategias. Se podría considerar la adición de terrenos variados, desafíos de nivel únicos y objetivos estratégicos cambiantes.
2. **Implementación de un multijugador:** Introducir un componente multijugador podría añadir una capa adicional de competencia y cooperación al juego. Esto podría permitir a los jugadores enfrentarse entre sí o trabajar juntos para completar niveles, lo que añade una dimensión social al juego. Implementar esto podría ser un reto significativo, pero también podría aumentar considerablemente la longevidad y el atractivo del juego.
3. **Implementación de historia y desarrollo de personajes:** Añadir una narrativa y un desarrollo de personajes más profundos podría aumentar la inmersión y el compromiso del jugador. Esto podría incluir la creación de escenas de historia, misiones narrativas, y diálogos de personajes para aportar más contexto y profundidad al mundo del juego.
4. **Modificación del sistema de turnos para hacerlo más del estilo RPG:** Modificar el sistema de turnos para asemejarse más a un RPG tradicional podría proporcionar una experiencia de juego más profunda y estratégica. Esto podría implicar una revisión significativa de la mecánica del juego y la IA, pero también podría ofrecer una experiencia de juego más rica y gratificante.

5. **Personalización de unidades:** Permitir a los jugadores personalizar sus unidades podría añadir una capa adicional de estrategia al juego. Los jugadores podrían ajustar las habilidades, equipamiento y aspecto de sus unidades, lo que podría resultar en un mayor compromiso y rejugabilidad.
6. **Implementación de un sistema de logros:** Un sistema de logros podría proporcionar metas adicionales para los jugadores y ofrecer un sentido de progresión más allá de simplemente ganar partidas.
7. **Optimización de la IA:** Aunque la IA ya utiliza algoritmos avanzados, siempre hay espacio para mejoras. Con el tiempo, podrías recoger datos sobre las estrategias más exitosas y utilizar estos datos para optimizar aún más la IA.
8. **Ampliación de plataformas:** Considerar la posibilidad de llevar el juego a otras plataformas. Aunque el desarrollo inicial se haya realizado en Unity, lo cual facilita la transposición a otras plataformas, cada una tiene sus propias particularidades y desafíos que deben tenerse en cuenta.

6.3.- Conclusiones

En conclusión, el desarrollo de este proyecto ha representado una experiencia profundamente enriquecedora y educativa. Además de reforzar mis habilidades existentes y conocimientos en inteligencia artificial y programación en C#, también ha servido para incursionar en los desafíos únicos que presenta el desarrollo de videojuegos.

Usando Unity que ha sido la columna vertebral de este proyecto, he obtenido un conocimiento práctico valioso sobre cómo se crean los juegos desde el inicio hasta su finalización, abarcando desde el diseño de la lógica del juego hasta la implementación de interfaces de usuario, pasando por la creación de inteligencia artificial para las entidades del juego.

Pero lo más importante, este proyecto ha permitido la consolidación de una amplia gama de habilidades y conocimientos adquiridos durante mi carrera en Ingeniería Informática. La aplicación de principios de ingeniería de software, diseño de interfaces de usuario, desarrollo de algoritmos, resolución de problemas y gestión de proyectos, entre otros, ha reforzado mi comprensión de estas disciplinas. Más allá de eso, ha destacado la importancia de integrar y aplicar estos conocimientos de manera coherente y efectiva en un proyecto complejo y multifacético.

Este ha demostrado ser más que la suma de sus partes. La unificación de diversos campos de conocimiento en un solo proyecto ha proporcionado una valiosa lección sobre cómo estos elementos interconectados trabajan juntos en el mundo real. Me ha permitido ver de primera mano cómo los principios teóricos que he aprendido en las aulas se traducen en la práctica y cómo pueden ser aplicados para crear soluciones innovadoras y efectivas a problemas reales.

La experiencia adquirida en este proyecto es, por tanto, invaluable. No solo me ha proporcionado un conjunto de habilidades técnicas muy buscadas, sino que también ha mejorado mi capacidad para gestionar y llevar a cabo proyectos complejos y desafiantes. Estoy convencido de que estas habilidades y experiencias serán un activo invaluable en mi futura carrera y contribuirán significativamente a mi crecimiento y éxito continuo en el campo de la ingeniería de software.

Bibliografía

Arkin, R. C. (1992). *Homeostatic Control for a Mobile Robot: Dynamic Replanning in Hazardous Environments.* Atlanta: Georgia Institute of Technology.

Cantrell, C. D. (2000). *Modern mathematical methods for physicists and engineers.* Cambrige: Cambridge University Press.

DHSFX. (2020, Julio 22). *Unity Assets Store*. Retrieved from Fantasy Videogame Music - RPG Soundtracks: https://assetstore.unity.com/packages/audio/ambient/fantasy/fantasy-videogame-music-rpg-soundtracks-174830

GAPH. (2017, Diciembre 12). *Unity Assets Store*. Retrieved from 52 Special Effects Pack: https://assetstore.unity.com/packages/vfx/particles/spells/52-special-effects-pack-10419

Guru, R. (2023). *Refactoring Guru*. Retrieved from Refactoring Guru: https://refactoring.guru/es/design-patterns/abstract-factory

Guru, R. (2023). *Refactoring Guru*. Retrieved from Refactoring Guru: https://refactoring.guru/es/design-patterns/singleton

HAIKU. (2022, Agosto 18). *Unity Assets Store*. Retrieved from Garni - Ambient Fantasy Music: https://assetstore.unity.com/packages/audio/ambient/fantasy/garni-ambient-fantasy-music-228355

Juliani, Berges, Teng, Cohen, Harper, Elion, . . . Lange. (2020). *Unity: A General Platform for Intelligent Agents.* San Francisco: Unity Technologies.

Krause, E. F. (1986). *Taxicab Geometry*. Nueva York: Addison-Wesley Publishing Company.

Microsoft. (2023, 02 15). *learn.microsoft.com*. Retrieved from https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/

Mixamo. (n.d.). *Mixamo*. Retrieved from https://www.mixamo.com/#/

Rabin, S. (2017). *Game AI PRO 3.* Boca Raton: Taylor & Francis Group, LLC.

Rogers, S. (2010). *Level Up! The Guide to Great Video Game Design.* West Sussex: John Wiley & Sons, Ltd.

Schultz, C. P., Bryant, R., & Langdell, T. (2005). *Game Testing All in One.* Boston: Thomson Course Technology PTR.

Smith, M., & Ferns, S. (2021). *Unity 2021 Cookbook, Fourth Edition.* Birmingham: Packt Publishing.

Sutton, R. S., & Barto, A. G. (2020). *Reinforcement Learnig.* Cambridge, Massachusetts: Library of Congress Cataloging-in-Publication Data.

Synty Studios. (2020). *Synty Store*. Retrieved from POLYGON - Dungeon Realms: https://syntystore.com/products/polygon-dungeon-realms?_pos=1&_psq=dungeon+re&_ss=e&_v=1.0

Synty Studios. (2020). *Synty Store*. Retrieved from POLYGON - Modular Fantasy Hero Characters: https://syntystore.com/products/polygon-modular-fantasy-hero-characters

Synty Studios. (2020). *Synty Store*. Retrieved from POLYGON - Fantasy Characters Pack: https://syntystore.com/products/polygon-fantasy-characters-pack

ToxicDelta. (2017, Abril 19). *Unity Assets Store* . Retrieved from Fantasy RPG Cursor Pack: https://assetstore.unity.com/packages/2d/gui/icons/fantasy-rpg-cursor-pack-87154

Unity Technologies. (n.d.). *Wikipedia*. Retrieved from Unity: https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)

Viverna. (n.d.). *https://www.artstation.com/viverna_3d*. Retrieved from https://www.artstation.com/search?sort_by=relevance&query=flintlock

Yannakakis, G. N., & Togelius, J. (2018). *Artificial Intelligence and Games.* Springer.