



MÁSTER EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DISEÑO Y DESARROLLO DE INFRAESTRUCTURA CLIENTE-SERVIDOR PARA COMPETICIONES DE IA EN PAC-MAN

Autor: Lucas Martínez Luna

Director: Javier Matanza Domingo

Co-Director: Adam Gaweda

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

DISEÑO Y DESARROLLO DE INFRAESTRUCTURA CLIENTE-SERVIDOR

PARA COMPETICIONES DE IA EN PAC-MAN

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2023/24 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

MARTINEZ
LUNA LUCAS
ALBERTO -
05941865E

Digitally signed
by MARTINEZ
LUNA LUCAS
ALBERTO -
05941865E
Date: 2024.06.17
00:58:05 +02'00'

Fdo.: Lucas Martínez Luna

Fecha: 15/ 06/ 2024

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Fecha: 15/ 06/ 2024

Adam
Matthew
Gaweda

Digitally signed
by Adam
Matthew Gaweda
Date: 2024.06.18
13:08:12 -04'00'

Fdo.: Javier Matanza Domingo

Fdo.: Adam Gaweda



GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DISEÑO Y DESARROLLO DE INFRAESTRUCTURA CLIENTE-SERVIDOR PARA COMPETICIONES DE IA EN PAC-MAN

Autor: Lucas Martínez Luna

Director: Javier Matanza Domingo

Co-Director: Adam Gaweda

Madrid

Agradecimientos

Agradecer a mi familia por su apoyo constante y por todo el esfuerzo dedicado a mi formación personal y académica.

Agradecer también a todos los profesores y compañeros que han contribuido significativamente a mi formación.

Por último, agradecimiento especial a mis directores de TFM, Adam Gaweda y Javier Matanza por su paciencia y apoyo durante el desarrollo del proyecto

DISEÑO Y DESARROLLO DE INFRAESTRUCTURA CLIENTE-SERVIDOR PARA COMPETICIONES DE IA EN PAC-MAN

Autor: Martínez Luna, Lucas Alberto.

Director: Matanza Domingo, Javier.

Co-Director: Gaweda, Adam.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto se centra en el diseño y desarrollo de una infraestructura cliente-servidor para competiciones de Inteligencias Artificiales en el juego de Pac-Man. El principal resultado de este trabajo es la creación de una plataforma robusta que permite a los estudiantes y entusiastas de la IA, desarrollar y poner a prueba sus algoritmos y estrategias en un entorno controlado.

Palabras clave: Inteligencia Artificial, Arquitectura Cliente-Servidor, Pac-Man, Java, competición.

1. Introducción

La Inteligencia Artificial, comúnmente conocida como IA, ha recorrido un largo camino desde sus inicios a mediados del siglo XX, llegándose a convertir en la tecnología más transformadora de la actualidad.

En cuanto a videojuegos se refiere, la inteligencia artificial ha estado presente prácticamente desde sus inicios. Cuando se juega contra “la máquina”, en verdad estas jugando contra algoritmos que intentan emular una inteligencia en sus movimientos.

Estos algoritmos de IA han ido evolucionando rápidamente desde los primeros juegos que hacían uso de esta tecnología, como Space Invaders o Pacman, hasta juegos como FIFA donde las experiencias de juego son cada vez más realistas y desafiantes. [1]

Por otro lado, el desarrollo de competiciones de inteligencia artificial en entornos controlados, especialmente en juegos, ha experimentado un notable crecimiento en los últimos años. Esta tendencia se ha visto impulsada por avances significativos en algoritmos de aprendizaje automático y en la capacidad de procesamiento de las computadoras modernas.

En este contexto, este proyecto Fin de Máster tiene como objetivo principal la creación de una infraestructura basada en el modelo-cliente-servidor para la realización y gestión de competiciones de inteligencias artificiales en el juego de Pac-Man. A diferencia del clásico juego del Pac-Man, la versión desarrollada en este proyecto se distingue por ser un entorno multijugador 1 contra 1, donde los jugadores compiten directamente entre sí buscando obtener una puntuación superior a la de su oponente.

2. Definición del proyecto

La infraestructura desarrollada incluye varias innovaciones. En primer lugar, se ha implementado una arquitectura modular que separa claramente las responsabilidades del

cliente y del servidor, permitiendo una gestión eficiente de los recursos y una comunicación fluida entre ambos. La comunicación entre el cliente y el servidor se realiza mediante sockets en Java, utilizando un protocolo personalizado basado en JSON para la transferencia de datos. Esta configuración asegura una transmisión de datos rápida y segura, permitiendo la interacción en tiempo real entre los jugadores y el sistema.

Además, se ha contenerizado el servidor a través de Docker, lo que garantiza la portabilidad y facilidad de despliegue del sistema en diversos entornos y sistemas operativos.

Por otro lado, se han realizado una serie de modificaciones al juego de Pac-Man que añade una capa más de complejidad y de competitividad en las estrategias de los jugadores. En primer lugar, se ha llevado a cabo una reestructuración del laberinto, modificando las estructuras y sus tamaños. En segundo lugar, se han diseñado e implementado diferentes 'power-ups' que dan diferentes ventajas a los jugadores que los recolectan. Además, para facilitar la creación y modificación de estos laberintos, se ha adoptado un enfoque basado en mini celdas, permitiendo una construcción modular y flexible de los mapas.

Por último, se han implementado diferentes algoritmos de inteligencia artificial, como A*, para mejorar la toma de decisiones y la navegación de los agentes en el juego.

3. Descripción del sistema

Como se ha explicado anteriormente, la arquitectura del sistema se basa en el modelo cliente-servidor. El servidor se encarga de gestionar los entornos de juego de partidas concurrentes y de coordinar la comunicación con múltiples clientes. Esta arquitectura modular facilita la gestión eficiente de los recursos y asegura una comunicación fluida.

En la siguiente imagen, Ilustración 1, se muestra un esquema del sistema. En esta figura se ha usado una línea roja discontinua para separar claramente los dos principales componentes del sistema, cliente y servidor.

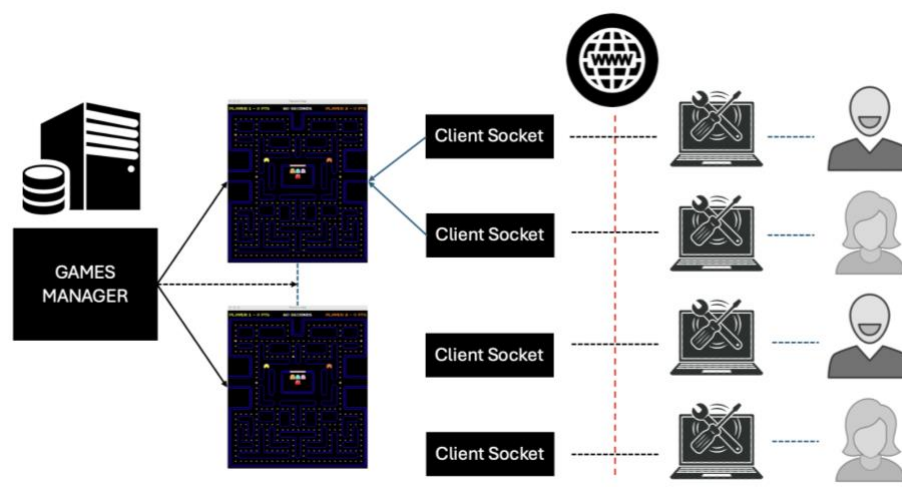


Ilustración 1 - Esquema del sistema desarrollado. Elaboración propia.

En la parte izquierda de este esquema se encuentran los componentes del servidor. Dentro de estos componentes destaca "Games Manager", quien se encarga de la creación,

gestión y control de las partidas. Este servidor mantiene el estado de cada juego y coordina la comunicación entre los diferentes clientes conectados.

Cada partida de Pac-Man se ejecuta como una instancia independiente en el servidor, permitiendo así la gestión simultánea de múltiples juegos. Los clientes, representados por los jugadores, se conectan a estas instancias de juego a través de sockets, utilizando el protocolo TCP/IP para asegurar una comunicación fiable y eficiente. Cada cliente establece una conexión con el servidor mediante un "Client Socket", a través del cual se envían y reciben datos en formato JSON.

Gracias a la estructura de la red que soporta conexiones distribuidas, los jugadores pueden interactuar con el juego desde diferentes ubicaciones.

4. Resultados

Uno de los resultados más importantes logrados con este proyecto es la creación de una infraestructura cliente-servidor eficiente. El sistema que se ha desarrollado permite una comunicación en tiempo real fluida, sincronizando correctamente las acciones de dos jugadores en una misma partida. Además, la infraestructura desarrollada es capaz de gestionar varias partidas de forma simultánea.

A continuación, se van a mostrar algunas de las principales pantallas desarrolladas. La primera de ellas, Ilustración 2, muestra la pantalla de inicio una vez el jugador se ha conectado al servidor. En esta pantalla, el jugador tiene la opción crear una sesión de partida o de unirse a una sesión existente.

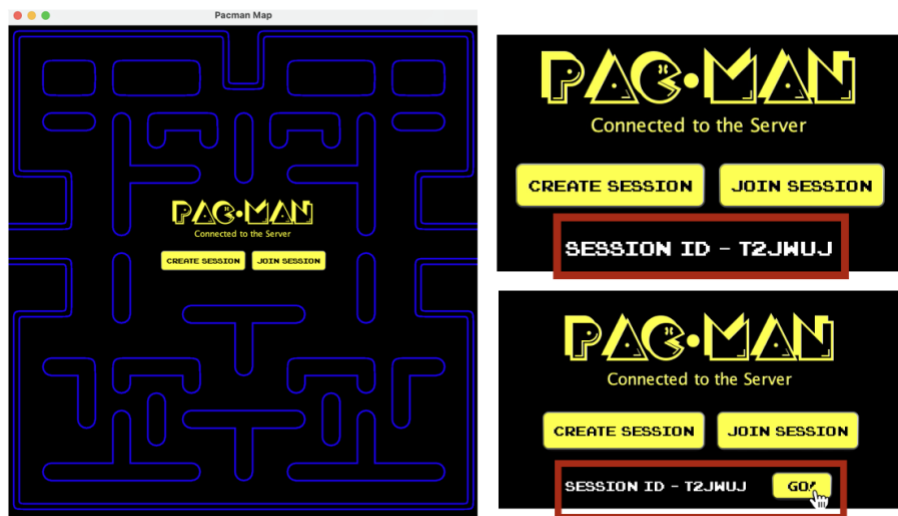


Ilustración 2 - Pantalla de inicio del juego. Elaboración propia.

Por otro lado, la siguiente imagen, Ilustración 3, muestra la pantalla de juego del Pac-Man una vez se han unido dos jugadores. Además, en esta pantalla se puede observar el nuevo diseño del laberinto, junto con los nuevos posicionamientos de los fantasmas y de los jugadores.

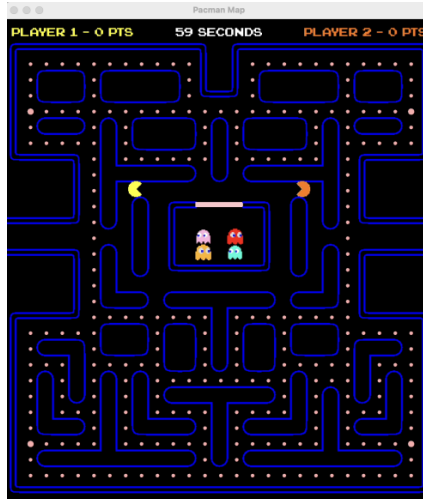


Ilustración 3 – Pantalla de juego de Pac-Man. Elaboración propia.

Por último, debido a las restricciones de tiempo, no se ha podido llevar a cabo una competición de IAs este curso académico. Sin embargo, para demostrar el espíritu competitivo y las capacidades de las implementaciones desarrolladas, se ha organizado un combate especial entre una implementación de elaboración propia y la implementación del director de este proyecto en Estados Unidos, el Doctor Adam Gaweda.

Para conmemorar este combate, se ha creado un cartel que, además de anunciar y destacar la batalla, también servirá como presentación del torneo que se va a realizar el próximo curso académico. En dicho torneo, se invitará a los nuevos participantes a intentar superar al ganador de esta batalla, incentivando así la innovación y desarrollo de nuevas estrategias.



Ilustración 4. Cartel de presentación. Elaboración de Adam Gaweda.

El combate llevado a cabo el 5 de junio entre ambas implementaciones dejó resultados muy interesantes. Entre estos destaca la existencia de un sesgo en uno de los jugadores,

dándole ventaja sobre el otro, debido a su posición inicial. Este sesgo fue corregido y se llevó a cabo una batalla final obteniendo los siguientes resultados:

	Player 1	Player 2	Ganador	Resultado Final
1	Lucas	Adam	Lucas	141-106
2	Lucas	Adam	Lucas	147-96
3	Lucas	Adam	Adam	113-132
4	Lucas	Adam	Lucas	135-102
5	Lucas	Adam	Adam	120-137
6	Adam	Lucas	Adam	140-125
7	Adam	Lucas	Lucas	114-138
8	Adam	Lucas	Lucas	102-145
9	Adam	Lucas	Lucas	113-135
10	Adam	Lucas	Adam	135-123

La principal diferencia entre las dos implementaciones que han participado en el combate, y que ha podido ayudar a mi agente a ganar un número mayor de partidas (6/10), es la forma en que se elige el objetivo. Mientras que la estrategia de Adam únicamente tiene en cuenta los puntos del laberinto, mi implementación evalúa las amenazas que suponen los fantasmas y además prioriza los power-ups cuando estos aparecen y están cerca.

5. Conclusiones

El proyecto ha logrado cumplir con todos los objetivos establecidos y expectativas generadas. Obteniendo como resultado final una plataforma robusta y escalable que permite la participación de múltiples estudiantes en el desarrollo de algoritmos de IA en un entorno controlado.

El mayor logro de este proyecto ha sido la exitosa implementación de la comunicación entre el cliente y servidor para la gestión de juegos multijugador en tiempo real.

En este proyecto se ha conseguido implementar con éxito el diseño del juego de Pac-Man, incluyendo las modificaciones necesarias para aumentar la competitividad y variabilidad de las estrategias. Además, se ha desarrollado y probado minuciosamente cada una de las diferentes fases del juego desarrolladas. Por otro lado, hay que destacar la creación de un archivo de configuración flexible que permite ajustar con facilidad los parámetros del juego.

Durante el desarrollo de este proyecto han surgido distintos desafíos, como la sincronización de partidas en tiempo real. Sin embargo, todos estos desafíos han sido superados con éxito, implementando soluciones como el uso de Threads para la gestión de múltiples clientes y de partidas concurrentes. Además, se ha probado el sistema para comprobar su correcto funcionamiento y asegurar que todos los jugadores tengan igualdad de condiciones, eliminando cualquier sesgo existente.

Los resultados obtenidos del combate final llevado a cabo entre mi implementación y la del coordinador de este proyecto, el doctor Adam Gaweda, han demostrado el correcto funcionamiento del sistema y la efectividad de los algoritmos desarrollados.

6. Referencias

- [1] «The Good Gamer,» [En línea]. Available: <https://thegoodgamer.es/el-papel-de-la-inteligencia-artificial-en-los-videojuegos/>.

DESIGN AND DEVELOPMENT OF CLIENT-SERVER INFRASTRUCTURE FOR AI COMPETITIONS IN PAC-MAN

Author: Martínez Luna, Lucas.

Supervisor: Matanza Domingo, Javier.

Co-Supervisor: Gaweda, Adam.

Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project focuses on the design and development of a client-server infrastructure for Artificial Intelligence (AI) competitions in the Pac-Man game. The main outcome of this work is the creation of a robust platform that allows students and AI enthusiasts to develop and test their algorithms and strategies in a controlled environment.

Keywords: Inteligencia Artificial, Arquitectura Cliente-Servidor, Pac-Man, Java, competición.

1. Introduction

Artificial Intelligence, commonly known as AI, has come a long way since its beginnings in the mid-20th century, becoming one of the most transformative technologies of our time. In the world of video games, AI has been around almost from the start. When you play against "the machine," you're up against algorithms designed to simulate intelligent behavior.

These AI algorithms have evolved rapidly, from the early games like Space Invaders and Pac-Man, to modern titles like FIFA, where the gameplay experiences are increasingly realistic and challenging. [1]

Additionally, the growth of AI competitions in controlled environments, especially in gaming, has surged in recent years. This rise is fueled by significant advancements in machine learning algorithms and the improved processing power of modern computers.

In this context, the main goal of this Master's thesis project is to create a client-server based infrastructure for organizing and managing AI competitions in the game of Pac-Man. Unlike the classic Pac-Man, this version is a 1v1 multiplayer environment where players compete directly against each other to achieve a higher score than their opponent.

2. Project definition

The developed infrastructure includes several innovations. Firstly, a modular architecture has been implemented, clearly separating the responsibilities of the client and the server. This allows for efficient resource management and smooth communication between the two. Communication between the client and the server is carried out using Java sockets, with a custom protocol based on JSON for data transfer. This setup ensures fast and secure data transmission, enabling real-time interaction between players and the system.

Additionally, the server has been containerized using Docker, ensuring the system's portability and ease of deployment across various environments and operating systems.

Furthermore, several modifications have been made to the Pac-Man game to add an extra layer of complexity and competitiveness to player strategies. Firstly, the maze has been restructured, altering its structures and sizes. Secondly, different 'power-ups' have been designed and implemented, providing various advantages to players who collect them. To facilitate the creation and modification of these mazes, a mini-cell approach has been adopted, allowing for a modular and flexible construction of the maps.

Finally, various artificial intelligence algorithms, such as A*, have been implemented to improve decision-making and navigation of the agents in the game.

3. System description

As previously explained, the system architecture is based on the client-server model. The server is responsible for managing the game environments of concurrent matches and coordinating communication with multiple clients. This modular architecture facilitates efficient resource management and ensures smooth communication.

The following image, Ilustración 5, shows a diagram of the system. In this figure, a dashed red line has been used to clearly separate the two main components of the system: the client and the server.

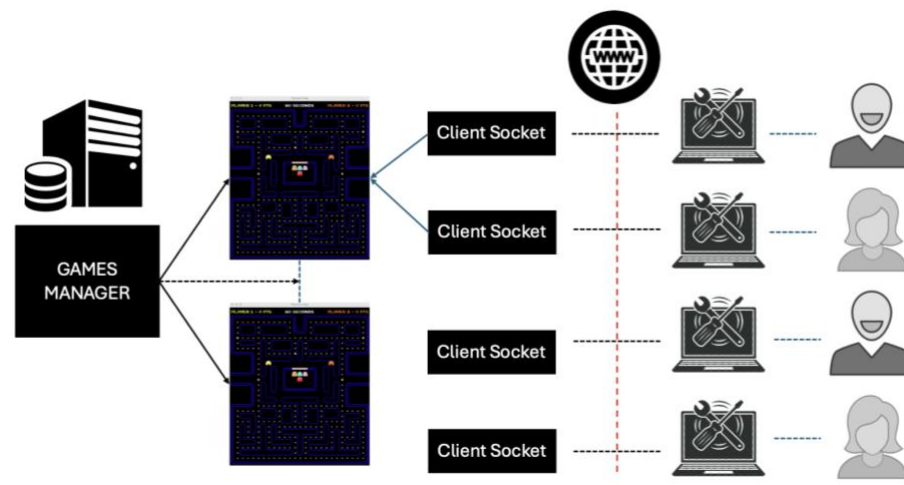


Ilustración 5 - Diagram of the Developed System. Self-produced.

On the left side of this diagram are the server components. Among these components, the "Games Manager" stands out, responsible for creating, managing, and controlling the matches. This server maintains the state of each game and coordinates communication between the different connected clients.

Each Pac-Man match runs as an independent instance on the server, allowing for the simultaneous management of multiple games. The clients, represented by the players, connect to these game instances via sockets, using the TCP/IP protocol to ensure reliable and efficient communication. Each client establishes a connection with the server through a "Client Socket," through which data is sent and received in JSON format.

Thanks to the network structure that supports distributed connections, players can interact with the game from different locations.

4. Results

One of the most important outcomes achieved with this project is the creation of an efficient client-server infrastructure. The system that has been developed allows smooth real-time communication, correctly synchronizing the actions of two players in the same game. Additionally, the developed infrastructure is capable of managing multiple games simultaneously.

Next, some of the main screens developed will be shown. The first of these, Ilustración 6, shows the start screen once the player has connected to the server. On this screen, the player has the option to create a game session or join an existing session.

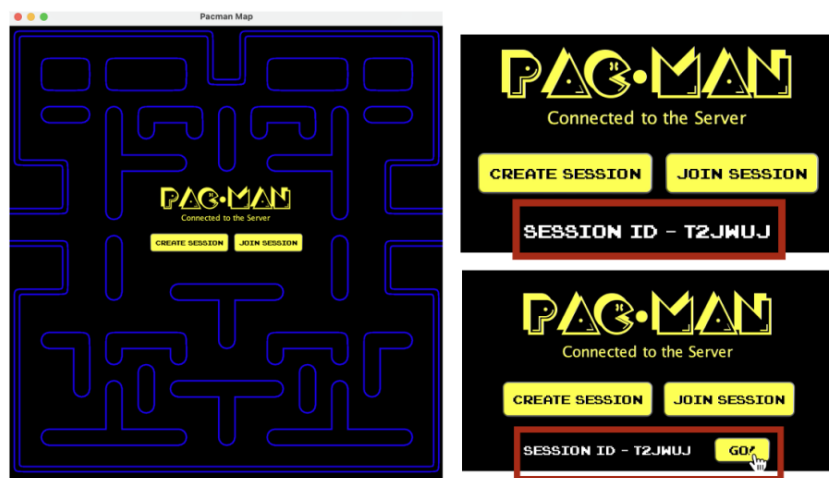


Ilustración 6 – Game Start Screen. Self-produced.

On the other hand, the following image, Ilustración 7, shows the Pac-Man game screen once two players have joined. Additionally, in this screen, you can observe the new maze design, along with the new positions of the ghosts and the players.

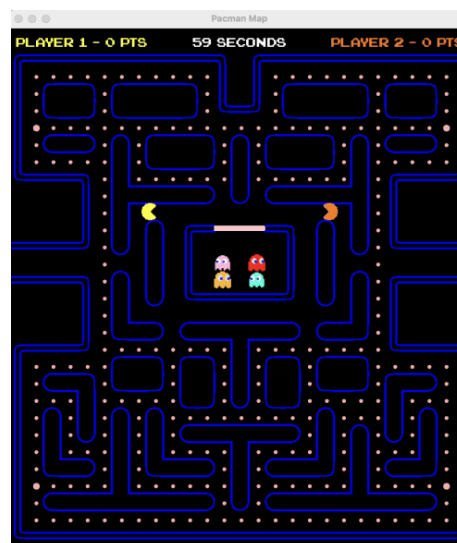


Ilustración 7. Pac-Man game screen. Self-produced.

Finally, due to time constraints, an AI competition could not be held this academic year. However, to demonstrate the competitive spirit and the capabilities of the developed implementations, a special match has been organized between an in-house implementation and the implementation of the project director in the United States, Doctor Adam Gaweda.

To commemorate this match, a poster has been created that not only announces and highlights the battle but also serves as a presentation for the tournament that will take place next academic year. In this tournament, new participants will be invited to try to surpass the winner of this battle, thus encouraging innovation and the development of new strategies.



Ilustración 8. Next academic competition poster. Produced by Adam Gaweda.

The match held on June 5 between both implementations yielded very interesting results. Among these, a bias in one of the players was highlighted, giving it an advantage over the other due to its initial position. This bias was corrected, and a final battle was conducted, obtaining the following results:

	Player 1	Player 2	Ganador	Resultado Final
1	Lucas	Adam	Lucas	141-106
2	Lucas	Adam	Lucas	147-96
3	Lucas	Adam	Adam	113-132
4	Lucas	Adam	Lucas	135-102
5	Lucas	Adam	Adam	120-137
6	Adam	Lucas	Adam	140-125
7	Adam	Lucas	Lucas	114-138
8	Adam	Lucas	Lucas	102-145
9	Adam	Lucas	Lucas	113-135
10	Adam	Lucas	Adam	135-123

The main difference between the two implementations that participated in the match, and which may have helped my agent win a greater number of games (6/10), is the way the target is chosen. While Adam's strategy only considers the points in the maze, my

implementation evaluates the threats posed by the ghosts and also prioritizes power-ups when they appear and are nearby.

5. Conclusions

The project has successfully met all established objectives and expectations. The final result is a robust and scalable platform that allows multiple students to participate in the development of AI algorithms in a controlled environment.

The greatest achievement of this project has been the successful implementation of client-server communication for the management of real-time multiplayer games. This project successfully implemented the design of the Pac-Man game, including necessary modifications to increase the competitiveness and variability of strategies. Additionally, each phase of the game developed has been meticulously tested.

Another highlight is the creation of a flexible configuration file that allows easy adjustment of game parameters. Various challenges arose during the development of this project, such as real-time game synchronization. However, all these challenges were successfully overcome by implementing solutions such as the use of Threads for managing multiple clients and concurrent games. The system has been tested to ensure proper functionality and to ensure that all players have equal conditions, eliminating any existing bias.

The results obtained from the final match between my implementation and that of the project coordinator, Doctor Adam Gaweda, have demonstrated the correct functioning of the system and the effectiveness of the developed algorithms.

6. References

- [1] «The Good Gamer,» [En línea]. Available: <https://thegoodgamer.es/el-papel-de-la-inteligencia-artificial-en-los-videojuegos/>.

Índice de la memoria

Capítulo 1. Introducción	7
Capítulo 2. Descripción de las Tecnologías.....	10
2.1 Lenguaje de programación y entorno de desarrollo.....	10
2.2 Contenerización y despliegue	11
2.3 Control de versiones y repositorio de código.....	11
2.4 Formato de transferencia de datos.....	12
2.5 Serialización y deserialización de datos.....	12
2.6 Comunicación cliente-servidor	13
Capítulo 3. Estado de la Cuestión	15
3.1 Inteligencia Artificial	15
3.1.1 Historia	15
3.1.2 Ventajas, beneficios y aplicaciones de la Inteligencia Artificial	18
3.1.3 Peligros y desafíos de la Inteligencia Artificial	19
3.1.4 Ramas de la inteligencia artificial	21
3.2 IA en los videojuegos.....	23
3.2.1 Competiciones de IA	26
3.3 Pac-Man	30
3.3.1 Otras Variaciones del Pac-Man.....	33
Capítulo 4. Definición del Trabajo	38
4.1 Justificación.....	38
4.2 Objetivos	39
4.3 Metodología	40
4.4 Planificación y Estimación Económica.....	43
Capítulo 5. Sistema Desarrollado.....	47
5.1 Análisis del Sistema	47
5.1.1 Requisitos funcionales.....	47
5.1.2 Requisitos no funcionales.....	48
5.2 Diseño del sistema.....	49

5.2.1	Arquitectura del sistema	49
5.2.2	Diagrama Casos de Uso	51
5.2.3	Diagrama de Clases.....	53
5.2.4	Diagrama de Secuencia	55
5.2.5	Comunicación Cliente-servidor	60
5.2.6	Diseño del juego de Pac-Man.....	62
5.3	Implementación.....	76
5.3.1	Comunicación cliente-servidor	76
5.3.2	Desarrollo del juego	81
5.3.3	Implementación del mapa	82
5.3.4	Archivo de configuración.....	83
5.3.5	Actualizaciones de juego.....	84
5.3.6	Métodos disponibles para desarrollo de IA.....	86
5.3.7	Desarrollo propio de Algoritmos para Agente IA.....	88
5.3.8	Contenerización del Servidor.....	89
Capítulo 6. Análisis de Resultados.....		91
Capítulo 7. Conclusiones y Trabajos Futuros.....		102
Capítulo 8. Bibliografía		105
ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS.....		113
ANEXO II: Implementación propia de IA		117
ANEXO III: Guía de usuario.....		128
ANEXO IV: Manual de instalación.....		133

Índice de figuras

Figura 1. Modelo de red en Java. Obtenido de [6]	13
Figura 2. Test de Turing. Obtenido de [8]	15
Figura 3. Juego Pong. Obtenido de [15]	23
Figura 4. Juego Space Invaders. Obtenido de [16]	23
Figura 5. Juego de Pac-Man. Obtenido de [18]	24
Figura 6. Juego de Donkey Kong. Obtenido de [19]	24
Figura 7. Juego de Half-Life. Obtenido de [20]	25
Figura 8. Ejemplo del MIT BattleCode en 2023. Obtenido de [23]	27
Figura 9. Ejemplo de juego de CodinGame. Obtenido de [25]	28
Figura 10. Interfaz de la RoboCup Soccer Simulation League. Obtenido de [27]	29
Figura 11. Juego del Pac-Man. Obtenido de [18]	31
Figura 12. Glitch nivel 256 en Pac-Man. Obtenido de [32]	33
Figura 13. Juego de Ms. Pac-Man. Obtenido de [34]	34
Figura 14. Juego de Pac-Man Championship Edition. Obtenido de [35]	35
Figura 15. Juego Pac-Man 256. Obtenido de [36]	36
Figura 16. Juego Pac-Man Battle Royale. Obtenido de [37]	37
Figura 17. 12 principios del manifiesto Agile. Obtenido de [40]	42
Figura 18. Agile vs Waterfall. Obtenido de [41]	43
Figura 19. Diagrama Gantt del TFM. Elaboración propia.	44
Figura 20. Salario bruto de un desarrollado software en Madrid. Obtenido de [42]	45
Figura 21. Conversión de salario bruto anual a €/hr. Obtenido de [43]	46
Figura 22. Diferencias entre los requisitos funcionales y no funcionales. Obtenido de [44]	47
Figura 23. Arquitectura del sistema. Elaboración propia.	50
Figura 24. Caso de uso cliente. Elaboración propia.	52
Figura 25. Caso de uso profesor. Elaboración propia.	53

Figura 26. Diagrama de clases. Elaboración propia.	55
Figura 27. Diagrama de secuencia cliente-servidor. Elaboración propia.	57
Figura 28. Diagrama de secuencia de partida (servidor). Elaboración propia.....	59
Figura 29. Comunicación cliente-servidor con sockets. Obtenido de [48]	60
Figura 30. Estructura JSON del protocolo de comunicación entre cliente y servidor. Elaboración propia.....	61
Figura 31. Construcción del mapa de Pac-Man. Elaboración propia.	63
Figura 32. Ejemplo de mini celdas gráficas con sus respectivos String asociados. Elaboración propia.	63
Figura 33. Ejemplo del algoritmo A*. Elaboración propia.	67
Figura 34. Gráfica del algoritmo Simulated Annealing. Obtenido de [49]	67
Figura 35. Ejemplo de máquinas de estado para Pac-Man. Elaboración propia.	68
Figura 36. Ejemplo del algoritmo Minimax. Obtenido de [50].....	68
Figura 37. Ejemplo de red neuronal profunda (Deep Q-Learning) en Pac-Man. Obtenido de [51].....	69
Figura 38. Ejemplo de estructura del sistema experto. Obtenido de [52]	69
Figura 39. Ejemplo algoritmo D* Lite. Obtenido de [53]	70
Figura 40. Máquina de estados de los modos de los fantasmas. Elaboración propia.	71
Figura 41. Reglas de movimiento de los fantasmas. Obtenido de [30]	71
Figura 42. Celdas usadas para los cálculos de movimientos de fantasmas. Obtenido de [30]	72
Figura 43. Ejemplo de cálculo no óptimo al objetivo de los fantasmas. Obtenido de [30].	72
Figura 44. Modo dispersión. Obtenido de [30]	73
Figura 45. Modo persecución del fantasma rojo. Obtenido de [30]	74
Figura 46. Modo persecución del fantasma rosa. Obtenido de [30].....	74
Figura 47. Modo persecución del fantasma naranja. Obtenido de [30].....	75
Figura 48. Modo persecución del fantasma azul. Obtenido de [30].....	75
Figura 49. Diseño de fantasma en modo pánico. Obtenido de [54]	75
Figura 50. Diagrama de paquetes del proyecto. Elaboración propia.	81
Figura 51. Implementación del mapa en formato String. Elaboración propia.	82

Figura 52. Recursos disponibles para las implementaciones de los alumnos. Elaboración propia.	88
Figura 53. Gestión de clientes y partidas en el servidor. Elaboración propia.	91
Figura 54. Pantalla de carga del juego. Elaboración propia.	92
Figura 55. Pantalla de inicio del juego. Elaboración propia.	92
Figura 56. Pantalla del gameplay. Elaboración propia.	93
Figura 57. Fantasmas en modo pánico y ubicación de power-up en el laberinto. Elaboración propia.	94
Figura 58. Pantalla con resultado final. Elaboración propia.	94
Figura 59. Creación de la imagen del servidor en Docker. Elaboración propia.	95
Figura 60. Ejecución del contenedor Docker. Elaboración propia.	95
Figura 61. Cartel de presentación del torneo. Elaboración de Adam Gaweda.	96
Figura 62. Sesgo en el comportamiento de los fantasmas. Elaboración propia.	97
Figura 63. Corrección de posición de los fantasmas y ‘Power Pellets’. Elaboración propia.	98
Figura 64. Comprobación de corrección de errores. Elaboración propia.	98
Figura 65. Corrección fallo de movimiento en el jugador 1. Elaboración propia.	99
Figura 66. Implementación del doctor Adam Gaweda. Elaboración propia.	100
Figura 67. Objetivos de Desarrollo del Milenio (ODM). Elaboración [56].	113
Figura 68. Objetivos de Desarrollo Sostenible. Obtenido de [60].	114
Figura 69. Instalación de Git. Elaboración propia.	133
Figura 70. Importar proyecto Git. Elaboración propia.	134
Figura 71. Clone URI. Elaboración propia.	134
Figura 72. Campos clonación en Git. Elaboración propia.	135
Figura 73. Branch Selection. Elaboración propia.	135
Figura 74. Ruta en la que se almacena el proyecto. Elaboración propia.	136
Figura 75. Verificar la versión java del proyecto. Elaboración propia.	136

Índice de tablas

Tabla 1. Costes Hardware.....	45
Tabla 2. Costes de personal	45
Tabla 3. Resultados del combate	100

Capítulo 1. INTRODUCCIÓN

La Inteligencia Artificial, comúnmente conocida como IA, ha recorrido un largo camino desde sus inicios a mediados del siglo XX, llegándose a convertir en la tecnología más transformadora de la actualidad.

Para entender que es la Inteligencia Artificial y cuáles son sus capacidades, hay que saber a qué hace referencia cada uno de estos términos, individualmente, en el ámbito tecnológico. En primer lugar, el término inteligencia tal y como se describe en un artículo de Iberdrola es, “la capacidad de un sistema para utilizar la información disponible, aprender de ella, tomar decisiones y adaptarse a nuevas situaciones”. Esta habilidad implica resolver problemas de manera eficiente dentro de las limitaciones y circunstancias actuales. Por otro lado, "artificial" indica que esta inteligencia no es propia, sino que se crea mediante la programación y el diseño de sistemas informáticos. [1]

Por tanto, "inteligencia artificial" se refiere al conjunto de algoritmos que tratan de simular procesos de inteligencia humana.

En cuanto a videojuegos se refiere, la inteligencia artificial ha estado presente prácticamente desde sus inicios. Cuando juegas “contra la máquina”, en verdad estás jugando, como se ha descrito anteriormente, contra unos algoritmos que intentan emular una inteligencia en sus movimientos.

Estos algoritmos de IA han ido evolucionando rápidamente desde los primeros juegos que hacían uso de esta tecnología, como Space Invaders o Pacman, hasta juegos como FIFA donde las experiencias de juego son cada vez más realistas y desafiantes. [2]

Por otro lado, el desarrollo de competiciones de inteligencia artificial en entornos controlados, especialmente en juegos, ha experimentado un notable crecimiento en los últimos años. Esta tendencia se ha visto impulsada por avances significativos en algoritmos

de aprendizaje automático y en la capacidad de procesamiento de las computadoras modernas.

En este contexto, este proyecto Fin de Máster tiene como objetivo principal la creación de una infraestructura basada en el modelo-cliente-servidor para la realización y gestión de competiciones de inteligencias artificiales en el juego de Pac-Man. A diferencia del clásico juego del Pac-Man, la versión desarrollada en este proyecto se distingue por ser un entorno multijugador 1 contra 1, donde los jugadores compiten directamente entre sí buscando obtener una puntuación superior a la de su oponente. Esta modificación aporta una nueva dimensión de desafío y estrategia, ampliando las capacidades requeridas tanto en los jugadores como en las inteligencias artificiales involucradas.

Además, tal y como se ha mencionado anteriormente, la inteligencia artificial juega un papel cada vez más relevante en el desarrollo tecnológico y la innovación. Por ello, este proyecto no solo busca establecer una plataforma que facilite la ejecución de torneos, sino que también permita a los participantes y desarrolladores familiarizarse con algoritmos de inteligencia artificial en acción. Observar y analizar cómo las IA se adaptan y responden en un entorno competitivo ofrece una oportunidad única para entender y mejorar las estrategias de juego y los propios algoritmos.

El proyecto presenta desafíos importantes en términos de escalabilidad y gestión de recursos en tiempo real. La infraestructura desarrollada deberá manejar múltiples partidas y competiciones de forma simultánea, debiendo optimizar así el uso de recursos, para poder garantizar una experiencia de usuario continua y sin interrupciones.

Por último, este proyecto no solo contribuye al ámbito académico al proporcionar una base sólida para futuras investigaciones en inteligencia artificial aplicada a juegos, sino que también promete extender su influencia a la industria del entretenimiento digital. Al integrar la competencia directa entre inteligencias artificiales en un entorno controlado, se abre la puerta a nuevas metodologías de enseñanza y aprendizaje automático, donde la máquina no solo aprende del entorno, sino directamente de las tácticas de un oponente. Este enfoque

podría ser extrapolado a otros ámbitos de la inteligencia artificial, mejorando la capacidad de las IA para actuar en situaciones complejas y dinámicas.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este capítulo se van a mencionar y explicar las tecnologías, protocolos y herramientas utilizadas para el desarrollo del proyecto.

2.1 LENGUAJE DE PROGRAMACIÓN Y ENTORNO DE DESARROLLO

El lenguaje de programación empleado para el desarrollo del proyecto es en su totalidad Java, en su versión JDK 11. Este lenguaje de programación orientado a objetos ofrece una extensa API que facilita el desarrollo de aplicaciones red, interfaces de usuario y procesamiento de datos.

Java es mundialmente conocido y usado para el desarrollo de aplicaciones de red, interfaces de usuario e inteligencia artificial, siendo en 2023, el cuarto lenguaje de programación más empleado. [3] Además, este proyecto ha sido desarrollado principalmente con un fin educativo, para la asignatura CSC 411 Artificial Intellegence en la Universidad estatal de Carolina del Norte, en el cual Java es el lenguaje de programación usado. Esto lo convierte en la opción ideal para este proyecto.

Por otro lado, se ha escogido Eclipse como IDE para el desarrollo de código del sistema. Este IDE está diseñado principalmente para el desarrollo de aplicaciones Java. Además, destaca por las herramientas que proporciona para la depuración y test del código, mejorando la eficiencia del desarrollo. [4]



Java



eclipse

2.2 CONTENERIZACIÓN Y DESPLIEGUE

Con el fin de poder desplegar el servidor en cualquier sistema manteniendo su consistencia se ha encapsulado el código del servidor en un contenedor a través de Docker.

Éste permite empaquetar el código del servidor compilado con todas sus dependencias y librerías externas en un contenedor, asegurando que el entorno de ejecución sea el mismo en cualquier lugar donde se despliegue el contenedor. Esta característica es crucial para el despliegue y escalabilidad del juego.



2.3 CONTROL DE VERSIONES Y REPOSITORIO DE CÓDIGO

Para almacenar y llevar a cabo la gestión de versiones del código del proyecto se ha usado GitHub. Esta plataforma basada en Git es mundialmente utilizada para el control de versiones de proyectos de desarrollo de software, puesto que ayuda a mantener el historial de cambios y a asegurar una colaboración eficiente entre los miembros del equipo de desarrollo.

Además, Eclipse, el IDE utilizado en este proyecto, permite trabajar con GitHub de manera fácil y eficiente. Eclipse ofrece integración directa con Git a través de EGit, un plugin que facilita la clonación de repositorios, la creación de ramas y la gestión de commits y push directamente desde el IDE. Esta integración simplifica enormemente el flujo de trabajo, permitiendo sincronizar el código con GitHub sin necesidad de salir del entorno de desarrollo.

En este proyecto se han creado dos repositorios, uno para el cliente y otro para el servidor.



2.4 FORMATO DE TRANSFERENCIA DE DATOS

Saber trabajar y escoger el formato en el que se envían datos dentro de una aplicación, se ha convertido en una tarea crítica. [5] Para ello, en primer lugar, hay que saber cuáles son los requisitos del sistema. En este caso, al ser un juego en tiempo real se priorizarán los formatos ligeros sobre formatos más pesados que incluyan más seguridad en los datos.

Por ello, se ha elegido JSON como formato de transferencia de información entre el cliente y servidor. JSON, destaca por ser un formato ligero y fácil de leer, asegurando una comunicación eficiente y efectiva.

Una de las principales ventajas de JSON es que su estructura basada en texto hace que los sean comprensibles tanto para humanos como para máquinas. Por último, otra de las ventajas que presenta JSON es su compatibilidad con diversos lenguajes de programación.



2.5 SERIALIZACIÓN Y DESERIALIZACIÓN DE DATOS

En el contexto de la comunicación entre cliente y servidor, la serialización y deserialización de datos juega un papel crucial. Al trabajar en Java con objetos complejos estos no pueden ser transmitidos directamente en formato JSON, sino que necesitan ser serializados. Del mismo modo, una vez serializados los datos, para que el receptor pueda entender los datos, estos deben ser transformados a su formato original mediante la deserialización.

Para realizar esta tarea de forma efectiva se ha decidido utilizar una API externa que facilite esta conversión de objetos a JSON y viceversa. Esta librería externa es GSON, una API desarrollada por Google para facilitar la conversión mencionada.

Las principales ventajas que ofrece esta biblioteca son: simplicidad y facilidad de uso, compatibilidad, flexibilidad y eficiencia.

2.6 COMUNICACIÓN CLIENTE-SERVIDOR

Como se ha descrito anteriormente, la comunicación entre el cliente y el servidor es un componente fundamental de este proyecto. Para implementar esta comunicación se ha hecho uso de sockets en Java. Tal y como se muestra en la Figura 1, estos sockets permiten establecer conexiones bidireccionales entre el servidor y el cliente. Además, las conexiones de estos sockets garantizan el envío y recibo de los datos de manera continua y en tiempo real.

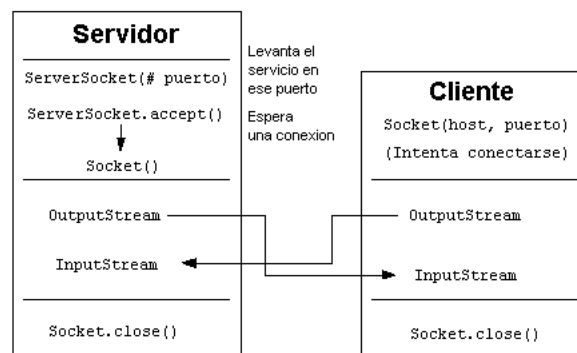


Figura 1. Modelo de red en Java. Obtenido de [6]

Estos sockets son la forma más eficiente y flexible de implementar la comunicación de red en java. La principal diferencia respecto a otros protocolos de comunicación como HTTP, basados en request-response, es que estos sockets permiten una comunicación persistente y bidireccional. Esto último, es crucial para videojuegos en línea, como el de este proyecto, que requieren de una interacción continua y de baja latencia.

Otra de las ventajas que ofrecen estos sockets es su escalabilidad, puesto que, tienen la capacidad de manejar múltiples conexiones simultáneamente. Esto último es un requisito indispensable de este sistema debido a que durante los torneos se llevarán a cabo numerosas partidas simultáneamente.

Por último, hay que destacar que estos sockets corren sobre el protocolo TCP. Este es un protocolo orientado a la conexión que garantiza la entrega confiable de datos entre cliente-servidor.

Capítulo 3. ESTADO DE LA CUESTIÓN

3.1 INTELIGENCIA ARTIFICIAL

3.1.1 HISTORIA

En esta sección se explicarán los puntos más relevantes de la historia de la Inteligencia Artificial. Esta abarca más de medio siglo de innovación y descubrimiento.

El sueño de crear máquinas inteligentes se remonta a la antigüedad, sin embargo, no fue hasta 1956, durante la conferencia de Dartmouth, que la inteligencia emergió de formalmente como un campo de estudio.

Antes de este suceso, en 1943 Warren McCulloch y Walter Pitts habían presentado su modelo de neuronas artificiales que consistía en una red de neuronas simplificadas capaz de realizar cálculos lógicos básicos. Por ello, este modelo es a veces considerado como la primera inteligencia artificial.

Unos años más tarde, en 1950, Alan Turing, considerado como padre de esta disciplina [7], publica en la revista Mind los fundamentos teóricos del ‘Test de Turing’. En su artículo “Computing Machinery and Intelligence” se considera la siguiente cuestión, ¿pueden las máquinas pensar? y propone un criterio para determinar si una máquina puede exhibir comportamiento inteligente indistinguible al de un humano. [8] Hoy en día este test se sigue utilizando.

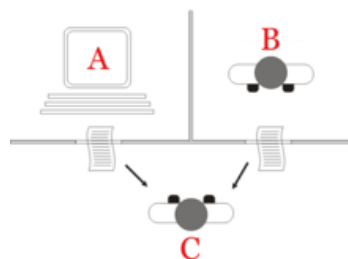


Figura 2. Test de Turing. Obtenido de [8]

Más tarde, en 1956, como se ha mencionado anteriormente tuvo lugar la conferencia de Dartmouth, cuyos principales representantes, John McCarthy, Marvin Minsky, Nathaniel Rochester y Claude Shannon, acuñaron el término de Inteligencia Artificial. En esta conferencia se presentaron los objetivos y visión de la IA y es por ello considerado como el nacimiento de esta disciplina. [9]

En 1958 John McCarthy introdujo LISP, el primer lenguaje de Inteligencia Artificial. Un año más tarde, en 1959, un trabajador de IBM, Arthur Samuel, acuñó el término de Machine Learning.

En 1961, General Motors instaló el primer robot industrial. Este se utilizó para reemplazar las tareas de montaje de los trabajadores.

Otro hito importante ocurrió en 1964 cuando Joseph Weizenbaum del MIT desarrolló ELIZA, uno de los primeros programas de procesamiento de lenguaje natural que podía simular una conversación con un humano. [1]

A pesar de todos estos avances y de las altas expectativas generadas, el entusiasmo inicial comenzó a disminuir hacia finales de los años 60, culminando con el informe de ALPAC (Automatic Language Processing Advisory Committee). Este informe sugería dejar de invertir dinero del gobierno estadounidense en la traducción automática, puesto que, no había alcanzado sus objetivos. Esto junto con el informe de 'Lighthill', en 1973, del gobierno británico, en el que expresaban su decepción con el progreso de la Inteligencia Artificial, hizo que la IA se adentrara en lo que se conoce como el primer invierno. [9]

El lanzamiento de R1 XCON en 1980, un sistema experto para configurar ordenadores VAX, propulsó una gran inversión por parte de Japón y Estados Unidos, gastándose ambos más de mil millones de dólares al año.

A mediados de la década de los 80, los sistemas expertos comenzaron a implementarse en aplicaciones industriales y comerciales, pero sus limitaciones pronto se hicieron evidentes. La dificultad para escalar y adaptar estos sistemas resultó en un segundo "invierno de la IA" a finales de los años 80.

La década de 1990 marcó un punto de inflexión con el resurgimiento del aprendizaje automático. El acceso a grandes conjuntos de datos y los avances logrados en la potencia informática permitieron el desarrollo de diferentes técnicas y algoritmos como árboles de decisión, o máquinas de vectores de soporte. Esta década estableció las bases de los avances futuros de la IA.

En 1997 se produce un hito histórico, la IA Deep Blue de IBM ganó al campeón mundial de ajedrez Gary Kasparov, siendo la primera vez en la historia que el hombre es superado por una máquina.

El cambio de milenio trajo consigo el auge del Deep Learning o aprendizaje profundo. Esta técnica, basada en redes neuronales profundas, ha revolucionado la manera en que las máquinas procesan la información y aprenden de los datos.

Durante las últimas dos décadas la inteligencia artificial ha alcanzado niveles sin precedentes de capacidad y adopción, transformando numerosos sectores.

Las aplicaciones de la IA ahora abarcan una amplia gama de industrias, desde la atención médica y la automoción hasta las finanzas y el entretenimiento. Los algoritmos avanzados de IA se utilizan para desarrollar asistentes virtuales como Siri y Alexa, que facilitan la interacción con dispositivos a través del reconocimiento y procesamiento del lenguaje natural. Además, los sistemas de recomendación de plataformas como Netflix y Amazon personalizan la experiencia del usuario mediante el análisis de grandes volúmenes de datos.

El futuro de la inteligencia artificial promete ser aún más transformador. Los desarrollos en áreas como la inteligencia artificial general (AGI), que busca crear sistemas con capacidades cognitivas generales similares a las humanas, y la inteligencia artificial explicable (XAI), que pretende desarrollar modelos que puedan justificar y explicar sus decisiones, son solo algunos de los desafíos que los investigadores están abordando.

Por último, la ética en la IA está ganando importancia. Es crucial garantizar que los sistemas de IA se desarrollen y desplieguen de manera justa y responsable, evitando sesgos y asegurando el respeto a los derechos humanos. Los investigadores y desarrolladores están

trabajando en crear marcos éticos y normativos que guíen el uso responsable de la inteligencia artificial, asegurando que esta poderosa tecnología beneficie a toda la sociedad.

3.1.2 VENTAJAS, BENEFICIOS Y APLICACIONES DE LA INTELIGENCIA ARTIFICIAL

Como se ha demostrado en la primera sección de este capítulo, la inteligencia artificial ha emergido como una tecnología transformadora que ofrece una amplia gama de beneficios y ventajas en diversos sectores. Su capacidad para procesar grandes volúmenes de datos, aprender de manera autónoma y realizar tareas complejas ha permitido mejorar la eficiencia, precisión y efectividad en múltiples aplicaciones. A continuación, se describen algunas de las principales ventajas de la inteligencia artificial [10] [11]:

- Automatización de Procesos: La IA permite que las máquinas realicen tareas repetitivas y monótonas que anteriormente requerían intervención humana. Esto no solo incrementa la eficiencia operativa, sino que también permite a los empleados enfocarse en actividades más estratégicas y creativas. La automatización mejora la productividad y permite a las empresas mantener un alto nivel de consistencia en la calidad de sus productos.
- Reducción del Error Humano: Los sistemas de IA pueden manejar tareas críticas con una precisión superior a la humana, minimizando el riesgo de errores.
- Mejora de la precisión y la toma de decisiones: La IA puede procesar y analizar grandes volúmenes de datos en tiempo real, proporcionando información precisa y actualizada para la toma de decisiones estratégicas.
- Avances en la Atención Médica: La IA está revolucionando el diagnóstico y tratamiento de enfermedades. Los sistemas de IA pueden analizar imágenes médicas con alta precisión, ayudando a los médicos a detectar enfermedades como el cáncer en sus etapas iniciales.
- Incremento de la Seguridad: En el ámbito de la ciberseguridad, la IA puede detectar patrones anómalos y posibles amenazas en tiempo real, ayudando a prevenir fraudes y ciberataques. Además, la IA se utiliza en la planificación y gestión de respuestas

rápidas ante emergencias y desastres naturales, mejorando la capacidad de respuesta y mitigación.

- *Innovación y Creatividad*: La IA también está siendo utilizada en campos creativos como el arte, la música y el diseño. Algoritmos de IA pueden generar obras de arte originales, componer música y diseñar productos innovadores. Estas aplicaciones no solo inspiran a los artistas, sino que también abren nuevas posibilidades para la innovación en diferentes industrias.
- *Asistencia y Soporte*: Los asistentes virtuales y chatbots basados en IA están transformando la manera en que las empresas interactúan con sus clientes. Estos sistemas pueden manejar consultas comunes de manera eficiente, proporcionando soporte técnico y mejorando la satisfacción del cliente al ofrecer respuestas rápidas y precisas.
- *Mejora en la Educación*: La IA está facilitando la creación de plataformas de aprendizaje personalizadas que adaptan el contenido educativo a las necesidades individuales de cada estudiante. Estas plataformas pueden identificar áreas de dificultad y ajustar el ritmo y el nivel de dificultad del material, mejorando así los resultados educativos.

3.1.3 PELIGROS Y DESAFÍOS DE LA INTELIGENCIA ARTIFICIAL

Tal y como se ha mencionado anteriormente, la inteligencia artificial ha demostrado ser una tecnología transformadora con el potencial de revolucionar múltiples industrias y mejorar significativamente nuestra calidad de vida. Sin embargo, junto con sus numerosos beneficios, la IA también plantea una serie de peligros y desafíos que deben ser cuidadosamente gestionados para asegurar su desarrollo y utilización responsable.

Los principales peligrosos y desafíos que afronta la Inteligencia Artificial son los siguientes, descritos en la revista Forbes [12] [13]:

- *Falta de Transparencia*: La complejidad de los modelos de aprendizaje profundo, a menudo descritos como "cajas negras", dificulta la comprensión de cómo se toman las decisiones. Esta opacidad puede generar desconfianza y resistencia hacia la

adopción de la IA. La falta de transparencia también complica la identificación de errores y la atribución de responsabilidades, planteando un reto importante en su aplicación práctica.

- *Prejuicios y Discriminación*: Los sistemas de IA pueden perpetuar o incluso amplificar los sesgos presentes en los datos de entrenamiento. Esto puede llevar a decisiones discriminatorias en áreas críticas como la contratación, la justicia penal y el crédito financiero. Es primordial desarrollar algoritmos imparciales y conjuntos de datos diversos para minimizar estos riesgos, asegurando un uso equitativo de la tecnología.
- *Preocupaciones por la Privacidad*: Las tecnologías de IA a menudo requieren el acceso y análisis de grandes volúmenes de datos personales, lo que plantea serios problemas de privacidad. La protección de estos datos contra el acceso no autorizado y el uso indebido es crucial para mantener la confianza del público.
- *Dependencia de la IA*: La dependencia excesiva de los sistemas de IA puede reducir la creatividad, el pensamiento crítico y la intuición humana. Mantener un equilibrio entre la toma de decisiones asistida por IA y la aportación humana es vital para preservar nuestras capacidades cognitivas. La interacción humana en los procesos de toma de decisiones sigue siendo crucial para garantizar que estas sean comprensibles y justificadas.
- *Desplazamiento Laboral*: La automatización impulsada por la IA puede llevar a la pérdida de puestos de trabajo en varios sectores, afectando principalmente a los trabajadores poco cualificados. Aunque se espera que la IA también genere nuevos empleos, es fundamental proporcionar programas de formación y capacitación para ayudar a los trabajadores a adaptarse al nuevo entorno laboral.
- *Pérdida de Conexión Humana*: La creciente dependencia de la IA en la comunicación y las interacciones podría reducir la empatía, las habilidades sociales y las conexiones humanas. Es importante mantener un equilibrio entre la tecnología y la interacción humana para preservar la esencia de nuestra naturaleza social. La IA debe ser una herramienta que complemente, y no reemplace, las relaciones humanas.

- Desinformación y Manipulación: Los contenidos generados por IA, como los deepfakes, pueden facilitar la difusión de información falsa y la manipulación de la opinión pública. Es crucial desarrollar herramientas para detectar y combatir la desinformación generada por IA. La educación y la concienciación pública son también esenciales para reconocer y resistir estos engaños.
- Riesgos de Ciberseguridad: La IA puede ser utilizada para realizar ciberataques más sofisticados. Es fundamental fortalecer las medidas de seguridad y desarrollar defensas robustas contra estas amenazas, protegiendo tanto la infraestructura digital como los datos sensibles.

3.1.4 RAMAS DE LA INTELIGENCIA ARTIFICIAL

Dentro de la Inteligencia Artificial encontramos múltiples ramas, cada una con su enfoque aplicaciones específicas. Estas ramas abarcan desde la simulación de procesos cognitivos humanos hasta la creación de algoritmos que permiten a las máquinas aprender y adaptarse de manera autónoma.

A continuación, se presentan las principales ramas de la IA [11]:

- Aprendizaje Automático (Machine Learning): El aprendizaje automático se refiere a la capacidad de las máquinas para aprender de los datos sin ser programadas explícitamente para cada tarea. Este proceso se basa en el uso de algoritmos que mejoran su rendimiento a medida que reciben más datos. Dentro de esta rama, se encuentran varios subtipos, como:
 - Aprendizaje Supervisado: Los algoritmos se entrenan con datos etiquetados, permitiendo a la máquina hacer predicciones o clasificaciones basadas en nuevos datos.
 - Aprendizaje No Supervisado: Los algoritmos buscan patrones y estructuras ocultas en datos no etiquetados.
 - Aprendizaje por Refuerzo: La máquina aprende a través de prueba y error, recibiendo recompensas o castigos por sus acciones, optimizando así su comportamiento a lo largo del tiempo.

- Aprendizaje Profundo (Deep Learning): El aprendizaje profundo es una subcategoría del aprendizaje automático que utiliza redes neuronales profundas para modelar patrones complejos en grandes volúmenes de datos. Las redes neuronales están inspiradas en la estructura del cerebro humano y consisten en múltiples capas de nodos interconectados. Este enfoque es especialmente efectivo en el reconocimiento de imágenes, el procesamiento del lenguaje natural y otras tareas que requieren la identificación de patrones complejos.
- Procesamiento de Lenguaje Natural (NLP): el procesamiento del lenguaje natural se centra en la interacción entre los ordenadores y el lenguaje humano. Estos sistemas NLP permiten que las máquinas puedan interpretar y comprender nuestro lenguaje de forma coherente. Entre las principales aplicaciones de esta rama se pueden encontrar asistentes virtuales, traductores automáticos y herramientas de análisis de texto.
- Visión por computadoras (Computer Vision): La visión por computadora permite a las máquinas interpretar y comprender el mundo visual a través de imágenes y videos. Los algoritmos de esta rama pueden realizar tareas como el reconocimiento de objetos, detección de movimiento o reconstrucciones tridimensionales. Esto es especialmente útil en aplicaciones como la conducción autónoma, vigilancia o realidad aumentada.
- Realidad aumentada (AR): La realidad aumentada es una combinación del mundo real con elementos virtuales. Esta tecnología permite a los usuarios interactuar con su entorno de forma única. Además, es especialmente útil e interesante sus aplicaciones en el campo de la medicina.
- Reconocimiento de voz (Speech Recognition): El reconocimiento de voz permite que las máquinas comprendan y respondan al lenguaje hablado. Sus principales aplicaciones son sistema de dictado, asistente virtual o aplicaciones de accesibilidad para personas con discapacidad.

3.2 IA EN LOS VIDEOJUEGOS

En el capítulo de la introducción se mencionó que la Inteligencia Artificial ha desempeñado un papel fundamental en la evolución de los videojuegos. En esta sección se pretende explorar más en profundidad como la IA ha sido aplicada a los videojuegos, sus principales avances y la situación actual en la industria del entretenimiento digital.

En primer lugar, es necesario introducir el concepto de NPC (non playable character). Los NPC son agentes (o personajes) controlados por una Inteligencia Artificial que puede tener diferentes roles dentro del juego. Estos comportamientos suelen estar predefinidos, haciendo que su interacción con el jugador sea limitada. [14]

La IA empezó en los videojuegos con los famosos juegos de ‘Pong’ (1972) y ‘Space Invaders’ (1978). En estos juegos, los comportamientos de los NPCs eran muy básicos y predecibles, puesto que, estaban basados en patrones fijos. Por ejemplo, en ‘Pong’, la paleta controlada por la máquina simplemente seguía la pelota de manera lineal, mientras que en ‘Space Invaders’, los invasores alienígenas se movían en una formación específica de izquierda a derecha, descendiendo lentamente. A pesar de su simplicidad, estos primeros usos de la IA sentaron las bases para futuros desarrollos más complejos.

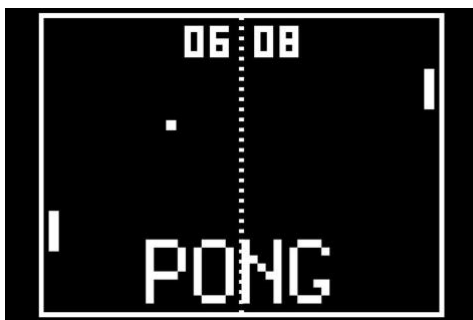


Figura 3. Juego Pong. Obtenido de [15]

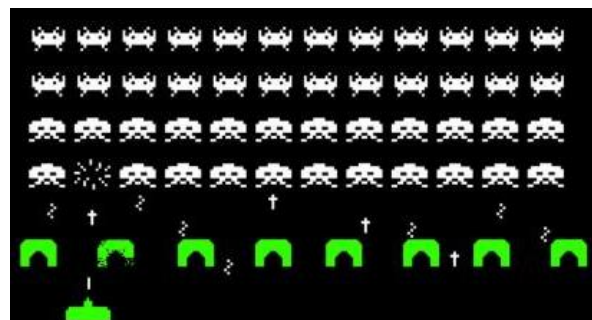


Figura 4. Juego Space Invaders. Obtenido de [16]

Los años 80 también marcaron un avance significativo con el lanzamiento de los juegos de ‘Pacman’ en 1980 y “Donkey Kong” en 1981.

En ‘Pacman’, la inclusión de 4 fantasmas (Blinky, Pinky, Inky y Clyde) con patrones de movimiento específicos distintos, añadió u nivel más alto de estrategia y, ofreció una experiencia de juego más dinámica y compleja. [17]



Figura 5. Juego de Pac-Man. Obtenido de [18]

Por otro lado, el juego de ‘Donkey Kong’, anteriormente mencionado fue otro hito. En este juego los movimientos del gorila y los barriles lanzados seguían patrones determinísticos, pero con una lógica interna que aumentaba la dificultad progresivamente. Este tipo de IA introdujo la idea de que los NPCs podrían responder a las acciones del jugador, aunque de forma limitada.

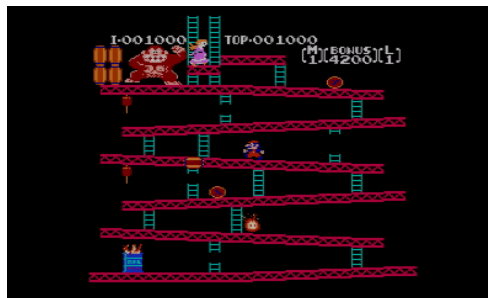


Figura 6. Juego de Donkey Kong. Obtenido de [19]

En la década de 1990 se produjo la transición de los juegos en 2D a entornos 3D. Esto condujo al desarrollo de técnicas más avanzadas de IA. En juegos como ‘Doom’ o ‘Quake’ se implementaron algoritmos de búsqueda (‘pathfinding’), como el algoritmo A*, que permitió a los NPCs moverse con mayor eficiencia en estos entornos tridimensionales.

Por otro lado, el juego ‘Half-Life’ lanzado en 1998, incluyó comportamientos tácticos avanzados en los NPCs, revolucionando la interacción en el mundo de los videojuegos. En

este juego los personajes eran capaces de buscar cobertura de manera inteligente, coordinar ataques en equipo y flanquear al jugador para aumentar su efectividad en combate. Con este juego se demostró como la IA podía simular comportamientos humanos complejos proporcionando una experiencia de combate mucho más inmersiva y desafiante. Este juego estableció el nuevo estándar en la industria.



Figura 7. Juego de Half-Life. Obtenido de [20]

Con el cambio de milenio, los videojuegos comenzaron a incorporar técnicas más sofisticadas de IA, incluyendo algoritmos de aprendizaje automático y redes neuronales. El juego ‘Black & White’ (2001) fue pionero en el uso de la IA adaptativa. Este juego permitía que las criaturas del juego aprendieran y evolucionaran en función de las acciones del jugador. [21] Además, estas criaturas podían desarrollar personalidades únicas, lo que añadía una dimensión impredecible en el juego.

Otro avance significativo fue la introducción de la IA para la generación de diálogos y narrativas adaptativas en los juegos de rol. Un ejemplo de esto es ‘The Elder Scrolls IV: Oblivion’ (2006), donde los NPCs podían interactuar entre sí y con el jugador afectando el desarrollo de la historia.

Durante la última década, los videojuegos han experimentado un incremento muy elevado en su complejidad y realismo. Juegos como ‘The Last of Us’ (2013) y ‘Red Dead Redemption 2’ (2018) han elevado los estándares de comportamiento de los NPCs. En el primero de ellos, ‘The Last of Us’, los enemigos pueden trabajar en equipo, comunicarse

entre sí y adaptarse a las tácticas del jugador. Esto ha conseguido generar un entorno y experiencia de juego extremadamente complejo, dinámico y sobre todo desafiante.

En el segundo, ‘Red Dead Redemption 2’, la simulación de la IA llegó a un nuevo nivel, donde los NPCs también eran capaces de llevar a cabo rutinas diarias realistas y reaccionar de manera verosímil a las acciones del jugador. Esto no solo mejora la inmersión, sino también establece un nuevo estándar para la interacción en juegos de mundo abierto.

Actualmente, la IA en el contexto de los videojuegos está explorando el uso de técnicas avanzadas de aprendizaje profundo y redes generativas adversarias (GANs) para crear contenidos y personajes aún más realistas.

Aun así, el futuro de la IA en los videojuegos promete ser aún más innovador. Con la integración de tecnologías como la realidad aumentada (AR) y la realidad virtual (VR), la IA tiene el potencial de crear experiencias de juego completamente inmersivas y personalizadas.

3.2.1 COMPETICIONES DE IA

El crecimiento exponencial de la IA ha conllevado la creación de numerosas competiciones dedicadas a medir y comparar las capacidades de distintos sistemas de IA. Estas competiciones son ideales para fomentar el aprendizaje, la innovación y el interés en esta emergente tecnología.

Debido a la larga lista de competiciones existentes, este apartado se va a centrar únicamente en aquellas que son más relevantes en el ámbito académico, puesto que, tiene el mismo propósito que este proyecto.

Como se ha mencionado anteriormente, estas competiciones desempeñan una función crucial proporcionando a los estudiantes e investigadores un entorno práctico para desarrollar y probar sus algoritmos.

A continuación, se describen algunas de las competiciones más relevantes y destacadas en este campo.

3.2.1.1 MIT BattleCode

Dentro de las competiciones más relevante entre Inteligencias Artificiales encontramos el MIT BattleCode. Esta prestigiosa competición es organizada y celebrada anualmente en Boston por la universidad de Massachusetts Institute of Technology (MIT).



Esta competición destaca por su complejidad técnica, su enfoque educativo y su capacidad para inspirar a los estudiantes a profundizar en el campo de la IA.

En esta competición la universidad invita a equipos de estudiantes a programar robos virtuales, en Java, que compiten entre sí en tiempo real en un entorno de batalla. Cada robot debe cumplir con tareas específicas y enfrentarse a desafíos complejos utilizando estrategias avanzadas de IA y algoritmos de toma de decisiones. Esta competencia se centra en la planificación estratégica, la cooperación entre agentes y la adaptabilidad en tiempo real. [22]

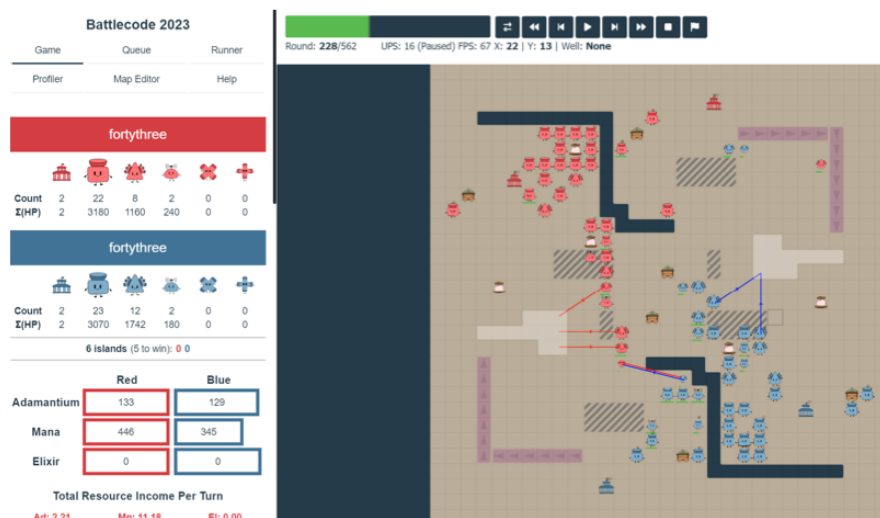


Figura 8. Ejemplo del MIT BattleCode en 2023. Obtenido de [23]

La estructura y objetivos de este sistema lo hacen especialmente relevante para proyectos educativos y de investigación en IA, siendo una fuente de inspiración directa para el proyecto de Pac-Man desarrollado en este Trabajo de Fin de Máster.

3.2.1.2 *CodinGame AI Contests*

CodinGame AI Contest es una serie de competiciones de Inteligencia Artificial que lleva a cabo la organización de CodinGame. Esta cuenta con una plataforma educativa online que desafía a desarrolladores de todo el mundo a resolver diversos problemas de programación [24].

Estas competiciones son consideradas como un punto de referencia para todos aquellos que busquen mejorar sus habilidades en los algoritmos de IA.

Además, CodinGame organiza de forma regular concursos en los que los participantes, deben programar bots para competir en juegos diseñados específicamente para evaluar diferentes habilidades de programación y diseño de algoritmos.

Una de las principales características de esta plataforma es que cuenta con desafíos para programadores de todos los niveles, desde principiantes hasta expertos. Esto último, fomenta que haya un alto grado de participación en los desafíos.

Dentro de los diferentes juegos se pueden encontrar desde simples juegos de estrategia hasta simulaciones complejas.



Figura 9. Ejemplo de juego de CodinGame. Obtenido de [25]

Además, durante la competición los bots son evaluados en función de su rendimiento a través de múltiples partidas. Al final de estas, los participantes reciben retroalimentación.

Por último, una de las grandes ventajas de esta plataforma es la gran comunidad con la que cuenta, donde los participantes pueden interactuar para compartir ideas y colaborar entre ellos.

3.2.1.3 RoboCup Soccer Simulation League

RoboCup Soccer Simulation League es una de las competiciones más reconocidas en el ámbito de la IA y la robótica. En esta competición, los participantes programan estrategias complejas con técnicas avanzadas de IA para partidos simulados de fútbol con robots.

La RoboCup fue fundada en 1997 con el ambicioso objetivo de conseguir desarrollar robots autónomos capaces de ganar al equipo campeón humano para 2050. [26]

Esta liga está atrayendo a numerosos investigadores, estudiantes y profesionales de todo el mundo. Los equipos participantes tienen que desarrollar algoritmos que permitan a los robots tomar decisiones estratégicas rápidas en tiempo real, cooperar con sus compañeros de equipo y adaptarse a las tácticas del oponente. Para ello, los equipos deberán implementar técnicas de visión por computadora, aprendizaje por refuerzo y planificación estratégica.



Figura 10. Interfaz de la RoboCup Soccer Simulation League. Obtenido de [27]

3.2.1.4 Kaggle Competitions

Por último, me gustaría destacar la plataforma de Kaggle. Esta mundialmente conocida plataforma permite organizar competiciones de data science y aprendizaje de automático, con el fin de desarrollar los mejores modelos predictivos.

Las competiciones de Kaggle se caracterizan por su enfoque práctico y su alto nivel de competencia. Los problemas planteados son complejos y requieren una combinación de técnicas avanzadas de análisis de datos, modelado predictivo y optimización de algoritmos.

La evaluación de los modelos se basa en métricas de rendimiento específicas, como la precisión, el área bajo la curva ROC (AUC-ROC) o el error cuadrático medio (MSE). Después de subir los resultados obtenidos, los participantes reciben retroalimentación en tiempo real. Para ello, tienen a su disposición el tablero de líderes, lo que les permite iterar y mejorar sus modelos a lo largo de la competición.

3.3 PAC-MAN

El juego original de Pac-Man fue creado por Toru Iwatani en 1980. Desde el comienzo el juego fue un éxito mundial, superando al famoso juego de Space Invaders y llegando a obtener el récord Guinness del arcade con más éxito de todo el mundo. [28]

Este juego además de revolucionar la industria de los juegos se convirtió también en un fenómeno cultural, influyendo durante las siguientes décadas a jugadores y desarrolladores.

El desarrollo del juego de Pacman tuvo lugar en 1979 con un joven diseñador de 22 años de Namco. El desarrollo de este juego duró tan solo un año y medio siendo lanzado al mercado el 21 de mayo de 1980. Este joven diseñador tuvo su inspiración a través de una pizza a la que le faltaba un trozo. [29]

En una época donde en los juegos de arcade predominaban las temáticas violentas o deportivas, este juego buscaba atraer tanto a hombres como a mujeres.

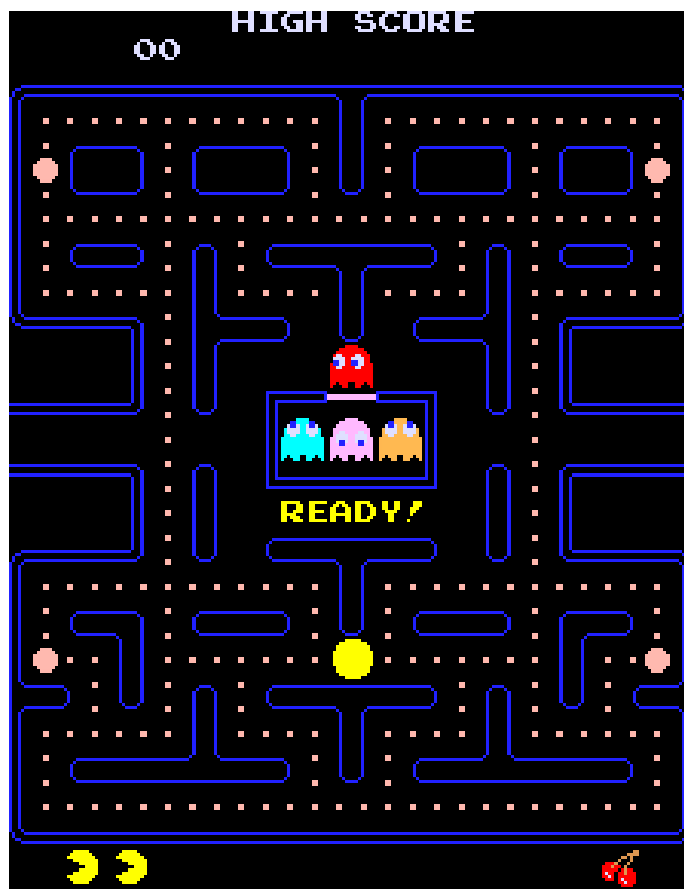


Figura 11. Juego del Pac-Man. Obtenido de [18]

El objetivo principal de este juego es comer todos los puntos del laberinto mientras se evita a los fantasmas, Blinky, Pinky, Inky y Clyde. Cada uno de estos fantasmas cuenta con un patrón de movimiento específico [30]:



Blinky: Este fantasma es conocido como “Shadow” o “Chaser” debido a que persigue de forma directa a Pac-Man. A medida que el jugador come más puntos, Blinky aumenta su velocidad, volviéndose más agresivo



Pinky: Este fantasma intenta emboscar a Pac-Man, intentado anticiparse a sus movimientos. Para ello, Pinky intenta posicionarse cuatros espacios por delante de la dirección actual de Pac-Man.



Clyde: Este fantasma alterna entre dos modos en función de la distancia a la que se encuentra de Pac-Man. Si está lejos, a más de 8 casillas, Clyde actuar de forma similar a Blinky, sin embargo, si se acerca demasiado, se dirige a la esquina inferior izquierda.



Inky: Este fantasma cuenta con el comportamiento más complejo e impredecible. El movimiento de Inky es una combinación de la de Blinky y la posición actual de Pac-Man.

Además, como se puede observar en la Figura 11, el mapa cuenta con cuatro puntos especiales denominados “Power Pellets”. Estos puntos cambian las tornas del juego, permitiendo a Pac-Man poder comerse a los fantasmas durante un tiempo limitado y, una oportunidad de ganar más puntos.

De esta forma, los fantasmas cuentan con tres modos de comportamiento distintos:

- Modo persecución ('Chase Mode'): En este modo, los fantasmas utilizan sus algoritmos específicos, anteriormente mencionados, para perseguir a Pac-Man.
- Modo dispersión ('Scatter Mode'): En este modo, los fantasmas se dirigen a las esquinas que han sido inicialmente designadas.
- Modo asustado ('Frightened Mode'): En este modo, los fantasmas huyen de Pac-Man.

Durante el juego, los modos de persecución y dispersión se van alternando en función de un temporizador. Por otro lado, el modo asustado únicamente se activa cuando Pac-Man se come un “Power Pellet”.

A medida que el jugador va avanzando de nivel, la velocidad del temporizador entre modos y la agresividad de los fantasmas, en sus algoritmos, aumenta, elevando el nivel de dificultad.

Por último, a nivel anecdótico, el juego original de Pac-Man se caracteriza por un glitch después de completar el nivel 255, conocido como “Nivel de muerte”. En este glitch, como se puede observar en la Figura 12, el lado derecho de la pantalla se vuelve en una mezcla de

números y letras, mientras el lado izquierdo se mantiene en su modo normal. Este glitch imposibilita poder superar este nivel. [31]

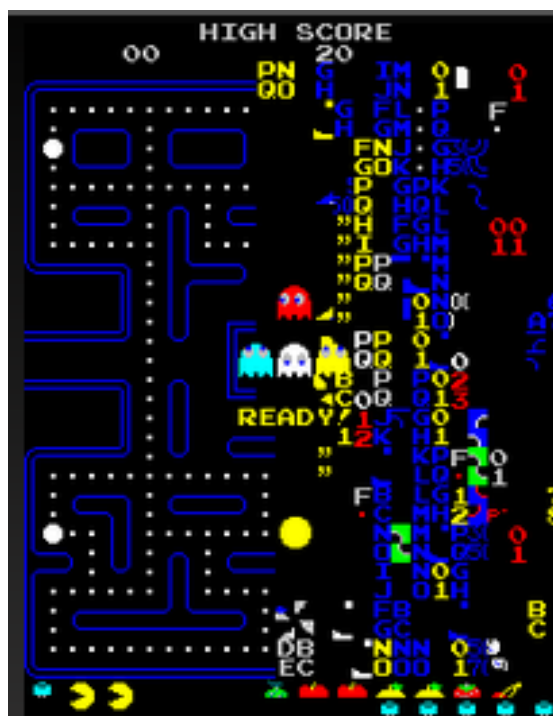


Figura 12. Glitch nivel 256 en Pac-Man. Obtenido de [32]

Por último, cabe destacar que los algoritmos específicos empleados por los fantasmas, reglas del juego y explicación de como se ha construido el juego, serán explicados con un mayor nivel de detalle, en los siguientes capítulos.

3.3.1 OTRAS VARIACIONES DEL PAC-MAN

Desde el lanzamiento del juego de Pac-Man en 1980, han surgido numerosas variaciones y adaptaciones, cada una de ellas añadiendo elementos y mecánicas nuevas. A continuación, se van a mencionar y describir alguna de las adaptaciones más importantes y reconocidas:

3.3.1.1 Ms. Pac-Man

Ms Pac-Man es una de las primeras variaciones que surgieron del juego original. Esta versión lanzada al año siguiente, en 1981, es una de las más queridas. Inicialmente esta versión no

era oficial, pero Midway impresionada por su éxito decidió lanzarlo oficialmente como un nuevo juego. [33]

Ms Pac-Man sigue teniendo el mismo objetivo principal que el juego original, comer todos los puntos del laberinto mientras se evita a los fantasmas. Sin embargo, este nuevo juego presenta algunas diferencias respecto a la versión original.

La primera y principal diferencia es que Ms Pac-Man cuenta con cuatro laberintos distintos, respecto al único que presenta el Pac-Man original. Estos laberintos van rotando a medida que el jugador avanza y cada uno de estos cuenta con configuraciones y colores únicos. Además, los patrones de movimiento de los fantasmas son menos predecibles, exigiendo a los jugadores a planear estrategias más dinámicas y reactivas.

Otra de las mejoras significativas de este juego es la inclusión de movimiento en las frutas bonus, añadiendo una capa más de desafío.

Por último, en este juego se introducen escenas que narran la historia de amor entre Pac-Man y Ms Pac-Man.



Figura 13. Juego de Ms. Pac-Man. Obtenido de [34]

3.3.1.2 Pac-Man Championship Edition

Pac-Man Championship Edition lanzado en 2007, es una modernización del clásico juego de Pac-Man. Esta versión fue desarrollada para conmemorar el 30 aniversario de juego original. Además, se diseñó para aprovechar las capacidades gráficas contemporáneas.

Este juego, se distingue por la transformación de los laberintos estáticos de la versión clásica a laberintos dinámicos que cambian a medida que el jugador progresa en la partida.

Otra de las características principales de este juego es, la inclusión de un temporizador. Con esto, los jugadores deben maximizar su puntuación dentro del tiempo establecido.

Por último, en esta versión, también fue introducido un nuevo sistema de puntuación y bonificación, con el fin de incentivar a los jugadores a tomar mayores riesgos para obtener mayores recompensas.



Figura 14. Juego de Pac-Man Championship Edition. Obtenido de [35]

3.3.1.3 Pac-Man 256

Como se ha comentado anteriormente, el juego original de Pac-Man cuenta con un glitch en el nivel 256 que hace que sea imposible de superarlo. Por ello, Hipster Whale y Bandai

Namco Studios decidieron lanzar en 2015, el juego de ‘Pac-Man 256’, haciendo honor a dicho glitch.

Esta versión de juego se caracteriza por tener un laberinto infinito. Por otro lado, el objetivo sigue siendo el mismo, conseguir obtener el máximo número de puntos evitando a los fantasmas.

Como se ha mencionado, este juego hace honor al glitch del juego original en el que en la parte derecha del laberinto hay una mezcla de números y letras. Por ello, del mismo modo, los desarrolladores decidieron crear una amenaza en la parte inferior del laberinto con el mismo formato que el glitch que fuerza al jugador a avanzar constantemente hacia la parte superior.

Además, en este juego se introducen bonificaciones y power-ups que no estaban presentes en la versión clásica.



Figura 15. Juego Pac-Man 256. Obtenido de [36]

3.3.1.4 Pac-Man Battle Royale

Pac-Man Battle Royale fue lanzado en 2011 por Bandai Namco y destaca por introducir una variante multijugador respecto al juego original. Esta versión permite jugar hasta entre cuatro jugadores simultáneamente.

Además, en esta adaptación, cada jugador tiene como objetivo sobrevivir mientras acumulan puntos al comer los puntos y frutas del laberinto.

Una de las principales diferencias de este juego es, que al comer los ‘Power Pellets’ los jugadores no solo pueden comerse a los fantasmas, sino que también pueden atacar y eliminar temporalmente al resto de jugadores.

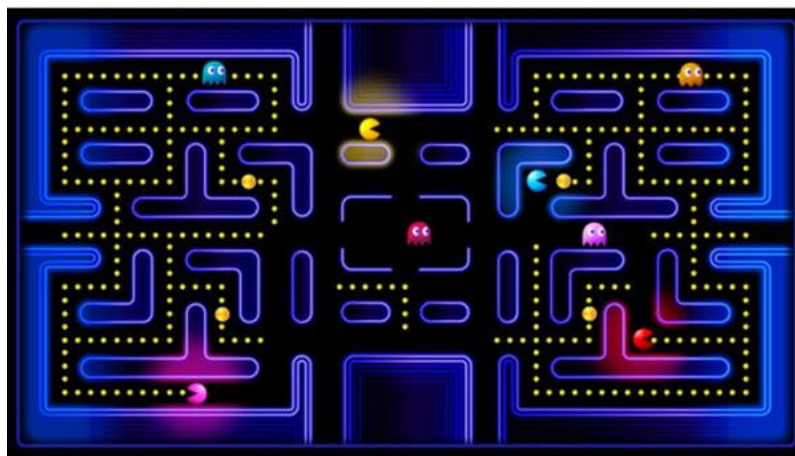


Figura 16. Juego Pac-Man Battle Royale. Obtenido de [37]

Capítulo 4. DEFINICIÓN DEL TRABAJO

En este capítulo se va a explicar el valor que aporta este proyecto respecto a otras soluciones existentes. Del mismo modo, se explicarán también los objetivos que persigue el proyecto, la metodología y planificación seguida, junto con la estimación económica del proyecto.

4.1 JUSTIFICACIÓN

Este proyecto Fin de Máster, con título "Diseño y Desarrollo de Infraestructura Cliente-Servidor para Competiciones de IA en Pac-Man", introduce varias innovaciones y mejoras significativas que lo distinguen de otras competiciones y juegos de inteligencia artificial previamente mencionados.

La principal innovación de este proyecto es, la creación de una infraestructura basada en el modelo cliente-servidor para competiciones de IA en el entorno del juego de Pac-Man, con un enfoque multijugador 1vs1. Este enfoque incrementa la complejidad del desarrollo de estrategias.

En comparación con la competición de BattleCode del MIT, principal fuente de inspiración para este proyecto, que se centra más en estrategias controladas por múltiples agentes, este proyecto simplifica las reglas del juego. Esta simplificación de las reglas ofrece varios beneficios significativos. El principal de ellos es, que permite a los participantes concentrarse en desarrollar y perfeccionar estrategias específicas, sin necesidad de gestionar múltiples unidades, lo cual puede ser extremadamente complejo y demandante. Por otro lado, este proyecto tiene como principal objetivo que los participantes exploren técnicas específicas de IA como la navegación eficiente y la anticipación de movimientos en tiempo real. Además, a diferencia del BattleCode, este proyecto permite la competición individual.

Por otro lado, este proyecto también fue inspirado en competiciones como CodinGame AI Contests y RoboCup Soccer Simulation League, que permiten a los participantes concentrarse únicamente en perfeccionar las estrategias desarrolladas.

Al comparar este proyecto con las variaciones que han ido surgiendo del Pac-Man a lo largo de las últimas décadas, se puede comprobar que ninguna ha adoptado un enfoque competitivo 1vs1 basado en IA. Las variaciones mencionadas incluyen nuevas mecánicas y desafíos, pero únicamente entre jugadores humanos. En contraste, este proyecto combina la estrategia junto con la inteligencia artificial en un entorno multijugador. Además, se han incluido algunas mecánicas, como el temporizador de tiempo o los power-ups que surgieron en versiones posteriores.

Por último, cabe destacar que este proyecto ha sido diseñado para la clase ‘CSC 411 Artificial Intelligence’ de la Universidad Estatal de Carolina del Norte, alineándose con los objetivos de aprendizaje de este curso. Sin embargo, no es exclusivo para este curso, dado que cualquier persona interesada en ella puede hacer uso de esta plataforma. Esto permite que el alcance del proyecto sea mayor, teniendo, tanto estudiantes como desarrolladores de diferentes contextos educativos, la oportunidad de aprender y aplicar algoritmos de IA en este entorno de juego.

4.2 OBJETIVOS

Como se ha explicado en la **Introducción**, el proyecto consiste en la realización y gestión de competiciones de inteligencias artificiales en el juego de Pac-Man. Por ello, los principales objetivos de este proyecto son los que se describen a continuación:

- Desarrollo y despliegue de una infraestructura cliente-servidor capaz de gestionar partidas simultáneas en tiempo real con alta disponibilidad y rendimiento. Además, se encapsulará el servidor en Docker, con el fin de asegurar la portabilidad y despliegue del servidor en diferentes entornos y sistemas operativos.

- Modificación y desarrollo del juego del Pac-Man, en java, para implementar un entorno de competición una contra uno, donde los estudiantes puedan aplicar y evaluar sus algoritmos de IA.
- Diseño y elaboración propia de un agente de IA utilizando algoritmos avanzados de inteligencia artificial para demostrar la aplicabilidad y efectividad de estos.
- Creación de una guía de usuario que proporcione instrucciones detalladas sobre cómo interactuar y configurar el sistema y cómo usar la interfaz gráfica. Además, se incluirá un apartado de recomendaciones para optimizar el rendimiento de los algoritmos junto con un apartado de soluciones a problemas comunes.

4.3 METODOLOGÍA

Una de las decisiones más difíciles e importantes que se enfrenta cualquier equipo al comenzar un proyecto es que metodología escoger. [38] Para ello, es de vital importancia conocer las metodologías existentes junto con sus principales características.

Las dos metodologías más usadas en el mundo empresarial son Waterfall (cascada) y Agile (ágil). A continuación, se muestra una breve descripción de cada una de estas.

La metodología Waterfall es más tradicional y es la que se ha estado utilizando durante los últimos 30 años. Esta metodología es adecuada para proyectos con requisitos bien definidos y no cambiantes desde el principio. Cuenta con 6 fases principales que se producen de forma secuencial. En primer lugar, se lleva a cabo un análisis de requisitos del sistema para posteriormente llevar a cabo el diseño. Una vez diseñado el sistema se lleva a cabo su implementación, al cual se realizan varios tests (unitarios, integración, etc) tras su finalización. Una vez el código desarrollado ha superado todos los test se despliega el sistema y se lleva a cabo un mantenimiento de este durante toda su vida.

Algunas de las principales ventajas de la metodología Waterfall son las siguientes.

- Facilidad para medir el progreso del proyecto
- Simplicidad en la planificación y diseño del sistema

- No es necesaria la presencia del cliente
- Posibilidad de diseño en paralelo de componentes software

Por otro lado, las principales desventajas de esta metodología son las siguientes:

- Dificultad en la identificación de requisitos
- Dificultad para cambiar el software ante imprevistos o insatisfacción del cliente

La metodología Agile, destaca por su enfoque iterativo y su respuesta a cambios. Esta metodología, se centra en la entrega rápida de software funcional.

Con el objetivo de tener una rápida respuesta a posibles cambios, la metodología Agile funciona en ciclos denominados Sprint de entre 2 y 4 semanas de duración. En cada uno de estos Sprint se establecen una serie de entregables que son priorizados por el valor que aportan al proyecto en cuestión. Tras la finalización de cada uno de estos ciclos se lleva a cabo una evaluación de los objetivos cumplidos, tanto por el equipo como por el cliente. Por ello, esta metodología destaca también por el alto grado de participación del cliente durante todo el proyecto.

Dentro del manifiesto Agile se recoge como primordial los siguientes 4 valores [39]:

- Individuos e interacciones sobre procesos y herramientas
- Software funcionando sobre documentación extensiva
- Colaboración con el cliente sobre negociación contractual
- Respuesta ante el cambio sobre seguir un plan

Por otro lado, también hay que destacar los siguientes 12 principios recogidos en el mismo manifiesto:

Los doce principios del manifiesto



Figura 17. 12 principios del manifiesto Agile. Obtenido de [40]

Entre las principales ventajas de la metodología Agile, destacan las que se citan a continuación:

- Fuerte sentido de pertenencia por parte del cliente, lo que facilita un mejor producto y una mayor satisfacción
- Se puede obtener un primer producto básico funcional antes

Por otro lado, alguna de las desventajas de esta metodología son las siguientes:

- Dedicación exclusiva al proyecto por parte de los miembros del equipo
- Posible incremento del coste por incumplimiento con los plazos asignados

A continuación, a modo de resumen se muestra una comparativa visual de la diferencia entre una metodología y otra.

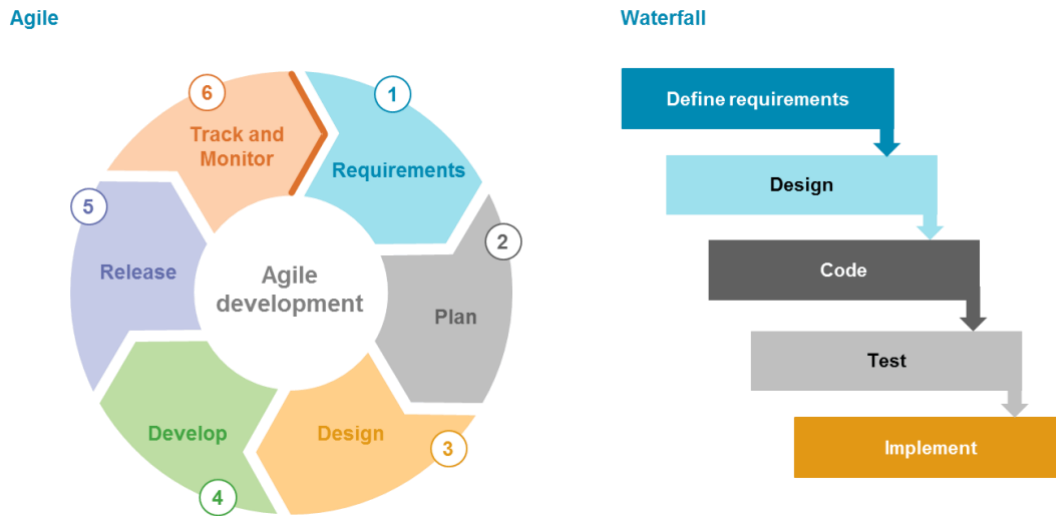


Figura 18. Agile vs Waterfall. Obtenido de [41]

Para este proyecto, se ha decidido implementar una metodología híbrida, aprovechando los beneficios que presenta cada una de ellas. Esto se debe a que este proyecto requiere tanto flexibilidad como estructura. Como se ha mencionado el proyecto necesita tener bien definido la estructura de las fases críticas del proyecto como la planificación y diseño inicial. Por otro lado, este proyecto también ha necesitado estar abiertos a cambios y mejoras durante toda la fase de desarrollo e implementación.

Por ello, para mantener un progreso constante y alineado con las expectativas del proyecto, fueron establecidas reuniones semanales con el doctor Adam Gaweda.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

El proyecto fin de Master tiene como duración un curso académico, en este caso, de Septiembre de 2023 a Junio de 2024.

Con el objetivo de llevar a cabo con éxito el proyecto se han definido las siguientes tareas:

- Análisis e instalación de herramientas: configuración del entorno de desarrollo y definición de los requisitos técnicos.
- Aprendizaje de algoritmos de IA.

- Modelo básico cliente-servidor: Desarrollo de un modelo básico cliente-servidor con el fin de obtener una profunda comprensión de los posibles desafíos que podrían surgir durante el desarrollo del sistema.
- Desarrollo de Pacman: diseño, implementación y configuración de la nueva versión de Pac-Man.
- Test y simulación del sistema: Se llevará a cabo una elaboración propia de IA que competirá contra una implementación realizada por el doctor Adam Gaweda.
- Documentación: Elaboración de los Anexos A y B junto con la memoria.

A continuación, se muestra el diagrama Gantt con las principales fases del proyecto y cronograma estimado.

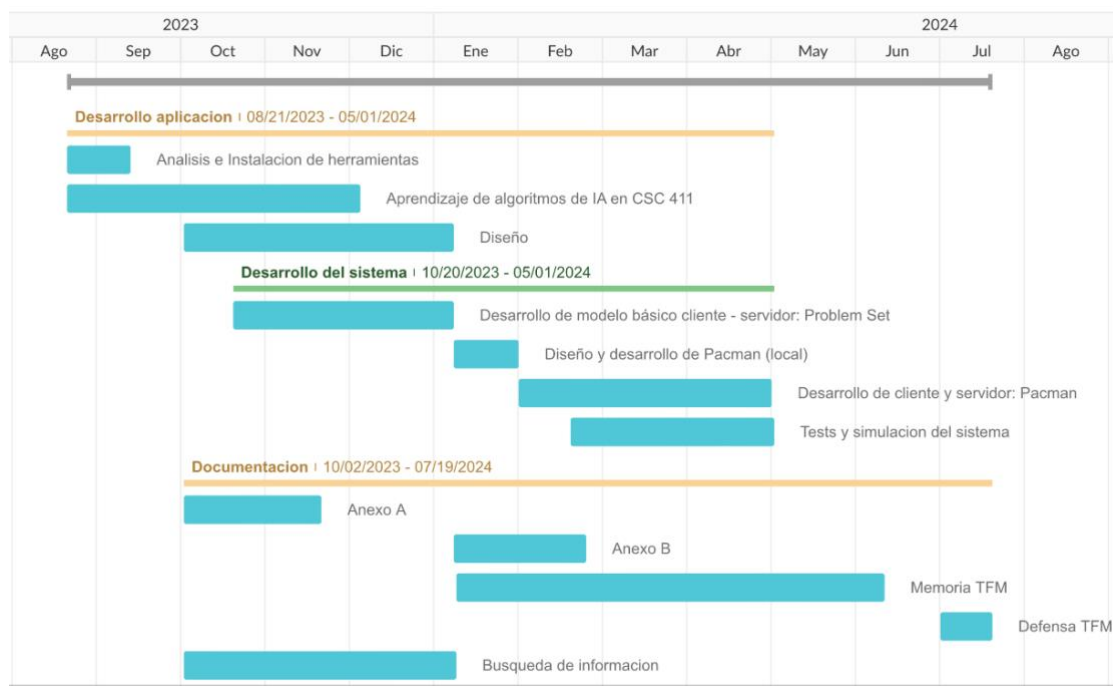


Figura 19. Diagrama Gantt del TFM. Elaboración propia.

Por último, para terminar este capítulo y esta sección se va a llevar a cabo una estimación económica del proyecto.

En primer lugar, para llevar a cabo una estimación precisa es fundamental identificar y considerar tanto los costes directos de hardware como los costes asociados al desarrollo de la infraestructura.

A continuación, se adjuntan los costes hardware:

Costes Hardware	
	Precio
Ordenador MacBook Pro 2017	2,000.00 €
Monitor LG	300.00 €
Cable HDMI	6.00 €
Adaptador	10.00 €
Total:	2,316.00 €

Tabla 1. Costes Hardware

Por otro lado, se adjunta la tabla con los costes de personal:

Costes de Personal				
	Nº trabajadores	Nº horas	€/hr	Total
Desarrolladores de Software	2	480	20	€ 19,200.00
Ingenieros de Sistemas/DevOps	1	200	20	€ 4,000.00
Especialistas en IA	1	300	17.5	€ 5,250.00
Diseñadores UI/UX	1	480	15	€ 7,200.00
Director de Proyecto	1	480	21	€ 10,080.00
			Total:	€ 45,730.00

Tabla 2. Costes de personal

Cabe destacar que los salarios han sido obtenidos de Infojobs [42], mientras que la conversión a €/hr se ha realizado con Talent [43].

Por ejemplo, como se puede comprobar en la siguiente imagen, el salario medio de un desarrollador software en Madrid es de 41.000 € brutos.



Figura 20. Salario bruto de un desarrollado software en Madrid. Obtenido de [42]

Utilizando la aplicación de Talent, se obtienen un salario de aproximadamente 20 € por hora trabajada.

¿Cuál es su salario ?	
Salario anual	€ 41.000,00
Salario mensual	€ 3.416,67
Salario quincenal	€ 1.708,33
Salario semanal	€ 788,46
Salario por día	€ 157,69
Salario por hora	€ 19,71
Horas trabajadas a la semana	🕒 40

Figura 21. Conversión de salario bruto anual a €/hr. Obtenido de [43]

Por tanto, teniendo en cuenta ambos costes, hardware y personal, se obtiene una estimación total de 48,046.00 €.

Capítulo 5. SISTEMA DESARROLLADO

En este capítulo, se va a explicar el sistema desarrollado en este proyecto Fin de Máster. Para ello, se ha decidido separar el capítulo en tres secciones distintas, análisis, diseño e implementación del sistema.

5.1 ANÁLISIS DEL SISTEMA

En primer lugar, se encuentra el análisis del sistema. Esta primera fase es crítica en cualquier proyecto de desarrollo software debido a que establece las bases para el diseño e implementación de la aplicación.

Para ello, se ha llevado a cabo un estudio de los requisitos funcionales y no funcionales del sistema. A continuación, en la Figura 22, se puede observar las principales diferencias entre estos dos tipos de requisitos.



Figura 22. Diferencias entre los requisitos funcionales y no funcionales. Obtenido de [44]

5.1.1 REQUISITOS FUNCIONALES

Los requisitos funcionales son especificaciones detalladas de las funciones que el sistema debe ofrecer. Estos requisitos son esenciales para definir el comportamiento, capacidades y alcance del sistema. A continuación, se describen los principales requisitos funcionales identificados:

- El sistema permitirá la creación, gestión y finalización de partidas de Pac-Man.
- El sistema podrá gestionar múltiples partidas de forma simultánea.
- Los participantes del juego, representados por agentes de IA, podrán interactuar con el sistema en tiempo real, enviando y recibiendo peticiones y respuestas.
- El sistema soportará la integración y control de agentes de IA, permitiendo a los desarrolladores implementar y probar diferentes estrategias de juego.
- El sistema proporcionará una interfaz gráfica intuitiva para la configuración de partidas y visualización de resultados.
- El sistema permitirá la interacción en tiempo real entre los jugadores y el servidor.
- El sistema gestionará la sincronización de los estados del juego entre el servidor y los clientes en tiempo real.

5.1.2 REQUISITOS NO FUNCIONALES

Por otro lado, los requisitos no funcionales son aquellos que describen cómo debe comportarse el sistema en vez de qué debe hacer. Mientras que los requisitos funcionales especifican las características y funciones del sistema, los no funcionales se centran en aspectos como rendimiento, usabilidad, fiabilidad y escalabilidad. Es de vital importancia identificar estos requisitos para asegurar la calidad y eficiencia del sistema, puesto que, pueden influir significativamente en la satisfacción del usuario final.

A continuación, se describen los requisitos no funcionales más importantes para el desarrollo de este proyecto:

- En términos de escalabilidad, el sistema debe ser capaz de soportar un aumento en el número de partidas simultáneas sin comprometer el rendimiento. Es esencial que el sistema pueda crecer y adaptarse a una mayor participación en el juego, debiéndose garantizar una experiencia de juego fluida, incluso en los momentos de mayor carga.
- En términos de rendimiento, es fundamental que la comunicación entre cliente y servidor se lleve a cabo de manera rápida y eficiente, minimizando el tiempo de latencia.

- Un requisito de vital importancia es la disponibilidad. El sistema debe estar disponible y operativo la mayor parte del tiempo
- En términos de seguridad, se implementarán medidas de seguridad para evitar accesos no autorizados.
- El sistema garantizará una buena experiencia de usuario a través de una experiencia de juego fluida y eficiente, con tiempos de respuesta rápidos y navegación intuitiva.
- El sistema será compatible con diferentes sistemas operativos Windows, macOS, ...

5.2 DISEÑO DEL SISTEMA

En esta sección se va a explicar el diseño del sistema desarrollado para la competición de agentes de Inteligencia Artificial en el juego del Pac-Man. Este diseño es esencial para el correcto desarrollo y funcionamiento de la aplicación. Este diseño asegura que todos los componentes se integran de forma exitosa en el sistema y funcionan de manera eficiente.

Esta sección se divide en varios apartados, en los que en cada uno de ellos se aborda aspectos específicos fundamentales para el correcto funcionamiento del sistema.

5.2.1 ARQUITECTURA DEL SISTEMA

La arquitectura del sistema se basa en el modelo cliente-servidor, donde los clientes representan a los jugadores, en este caso sus implementaciones de IA, y el servidor es el encargado de gestionar las partidas. Esta gestión incluye numerosas tareas, entre ellas se encuentra, crear un entorno individual de partida, actualizar el estado del juego o coordinar las interacciones entre los diferentes agentes.

Los principales componentes del sistema son los que se muestran a continuación:

1. Cliente: En primer lugar, el cliente proporciona una interfaz gráfica (UI), en la que los jugadores pueden configurar sus partidas, observar el desarrollo del juego y analizar sus estrategias. Además, el cliente es el encargado de desarrollar la lógica que controla el comportamiento del agente en tiempo real, en función del estado del

juego. Por último, el cliente es el encargado de manejar la comunicación con el servidor a través de los sockets, enviando sus acciones y recibiendo las actualizaciones del entorno de juego.

2. Servidor: El servidor controla la creación, gestión y finalización de las partidas. Es el encargado de actualizar el estado del juego y asegurar que la sincronización de las interacciones entre los agentes se realiza de forma coherente. Por otro lado, el servidor es el encargado de simular el comportamiento de los fantasmas a través de diferentes algoritmos de IA. Por último, el servidor también facilita la comunicación a través de sockets con el cliente, para el intercambio de mensajes en formato JSON a través del protocolo TCP/IP.

A continuación, se muestra una representación visual de los componentes de la arquitectura del sistema, Figura 23, junto con la relación entre dichos componentes.

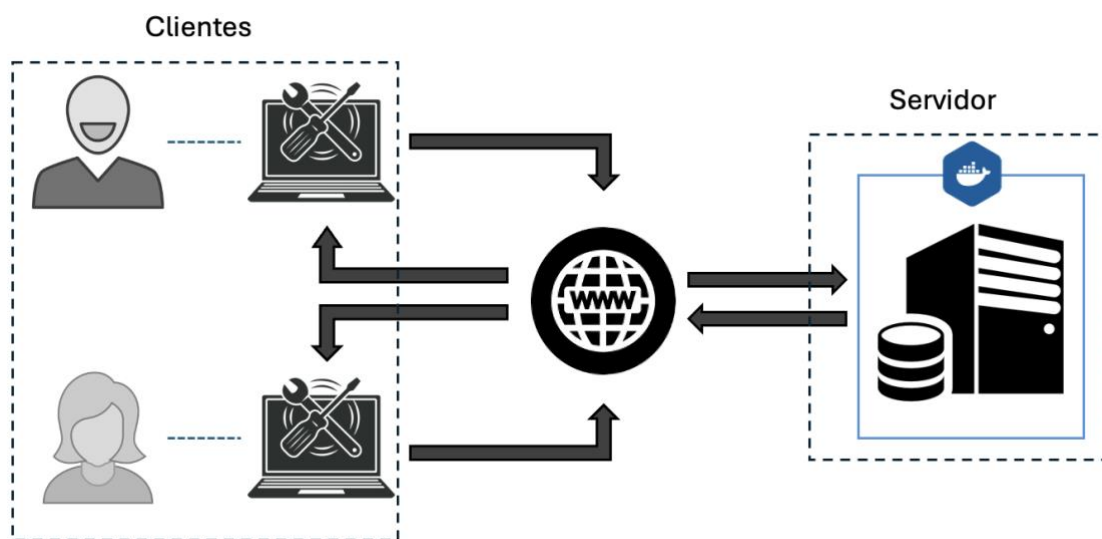


Figura 23. Arquitectura del sistema. Elaboración propia.

En los próximos apartados se detallará, a través de diferentes diagramas, como se relacionan internamente las clases y componentes del sistema, obteniendo una visión más granular del sistema. Se describirán los diferentes gestores y módulos de tanto el cliente como del servidor y como se relacionan entre sí.

Los diagramas UML (Lenguaje Unificado de Modelado) son una herramienta importante en el desarrollo software. Estos diagramas facilitan la identificación de posibles errores en las estructuras de la aplicación. [45]

Dentro de estos diagramas UML encontramos dos tipos de diagramas, los de estructuras y los de comportamiento.

Los diagramas estructurales representan la estructura estática del sistema. Además, estos diagramas se centran como se organizan y relacionan los componentes del sistema. Dentro de estos diagramas destacan los diagramas de clases y los diagramas de paquete.

Por otro lado, los diagramas de comportamiento se centran en cómo se comporta el sistema a lo largo del tiempo. Los principales diagramas de comportamiento son los diagramas de caso de uso, diagramas de secuencia y diagramas de máquina de estados.

5.2.2 DIAGRAMA CASOS DE USO

Como se ha mencionado anteriormente, los diagramas de caso son un tipo de diagrama de comportamiento en el que se visualizan las necesidades de diferentes actores en el sistema, y como estos interactúan entre sí. [46] Además, estas necesidades son una expresión de los requisitos funcionales anteriormente identificados.

Uno de los beneficios que presentan estos diagramas es asegurarse desde las primeras fases del proyecto que las expectativas de los stakeholders son comprendidas por los desarrolladores. Por otro lado, estos diagramas permiten identificar y priorizar las tareas más importantes para el cumplimiento de los requisitos del sistema.

Dentro de estos actores se han identificado dos actores principales:

1. Jugador (desarrollador de agente de IA). Esta persona es responsable de crear y entrenar los agentes de IA que competirán en el juego.
2. Profesor: Esta persona es responsable de realizar torneos y gestionar el sistema.

En la Figura 24, se puede observar los diferentes casos de usos del actor jugador. En primer lugar, un caso de uso imprescindible es el de ‘construir una implementación de IA’, debido a que, sin esto no es posible participar en ningún juego.

Además, el jugador, alumno o desarrollador, podrá tanto crea una partida como unirse a una sesión de juego. Para poder comprobar cómo funcionan sus estrategias desarrolladas junto con las de su adversario, el cliente necesita visualizar el juego, junto con el resultado final de la partida.

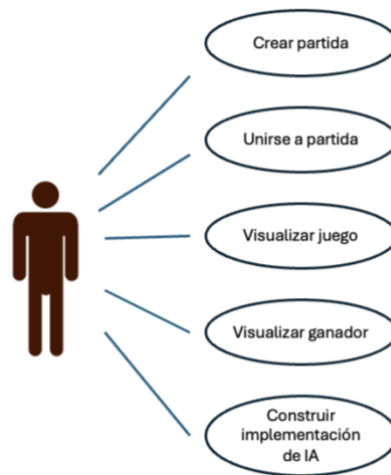


Figura 24. Caso de uso cliente. Elaboración propia.

Por otro lado, en la Figura 25, se puede observar los diferentes casos de uso que presenta un profesor. En primer lugar, un profesor va a necesitar configurar las reglas de las partidas, como pueden ser la duración del juego o el número de powerups que aparecen durante la partida.

En segundo lugar, el profesor necesitar conocer historial de resultados de las partidas para conocer los ganadores. Por ello, el servidor almacenará internamente todos los estados finales de las partidas.

Por último, un profesor puede visualizar una partida entre dos alumnos en la cual él no participa.

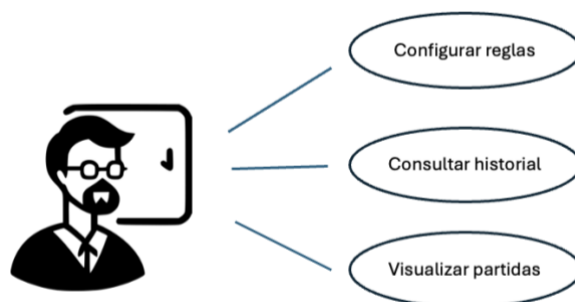


Figura 25. Caso de uso profesor. Elaboración propia.

5.2.3 DIAGRAMA DE CLASES

A diferencia de los diagramas de caso de uso, los diagramas de clase son una representación estática de las clases que componen el sistema, en este caso, el juego del Pac-Man.

En la Figura 26, se ha representado el diagrama de clases de este proyecto. Dada la complejidad y longitud de este proyecto, el diagrama representado únicamente incluye las clases de mayor importancia.

Además, tanto el desarrollo del cliente como el del servidor comparten numerosas clases. Por ello, en este diagrama se han incluido todas las clases comunes a estos dos componentes del sistema. Por otro lado, se han incluido algunas clases individuales de cada uno de estos componentes en recuadros de diferentes colores.

En las clases comunes a ambos componentes encontramos la clase ‘Ghost’ de la cual heredan cada uno de los fantasmas individuales. Todos los fantasmas cuentan con los atributos de posición actual y último movimiento realizado, por lo que se ha incluido una relación de asociación con la clase ‘Position’ y con la enumeración ‘Move’. También cuenta con un atributo de ‘Mode’ que contine el modo de movimiento.

Dentro de estas clases comunes se encuentra la clase ‘Pacman’, el cual cuenta con un ID para diferenciarlo de su oponente. Esta clase cuenta con los atributos de ‘position’ y ‘score’ donde se almacena la posición y los puntos de cada jugador, respectivamente.

Por último, dentro de las clases comunes a ambos componentes se encuentra la clase ‘Game’ con toda la información del estado de juego. Un juego tiene dos pac-man y 4 fantasmas, cada uno de los cuales es una instancia de las subclases específicas de ‘Ghost’. Además, cada partida cuenta con un id único, para poder ser localizado fácilmente. Estos id tienen una longitud de seis caracteres y son generados de manera aleatoria.

En cuanto a las clases específicas del cliente, representadas dentro del recuadro azul, se encuentran la interfaz gráfica, ‘BoardUI’, y el que se encarga de gestionar las actualizaciones del juego ‘Game Controller’. Esta última clase se recibe dichas actualizaciones de la clase ‘ServerHandler’. Esta clase es la encargada de llevar a cabo la comunicación con el servidor a través de sockets.

Por otro lado, las clases específicas del servidor han sido representadas dentro del recuadro rojo. Este recuadro contiene en primer lugar la clase ‘GameHandler’, cuya función es llevar a cabo un seguimiento de todas las partidas que se estén ejecutando en ese momento, de forma simultánea. Esta clase se comunica con el resto de las clases del servidor a través de ‘ClientHandler’. Esta última clase es la encargada de procesar las acciones enviadas por los jugadores a su partida individual. Para ello, estos jugadores envían sus movimientos incluyendo en un campo específico el id de su sesión de juego.

A continuación, se muestra el diagrama de clases realizado:

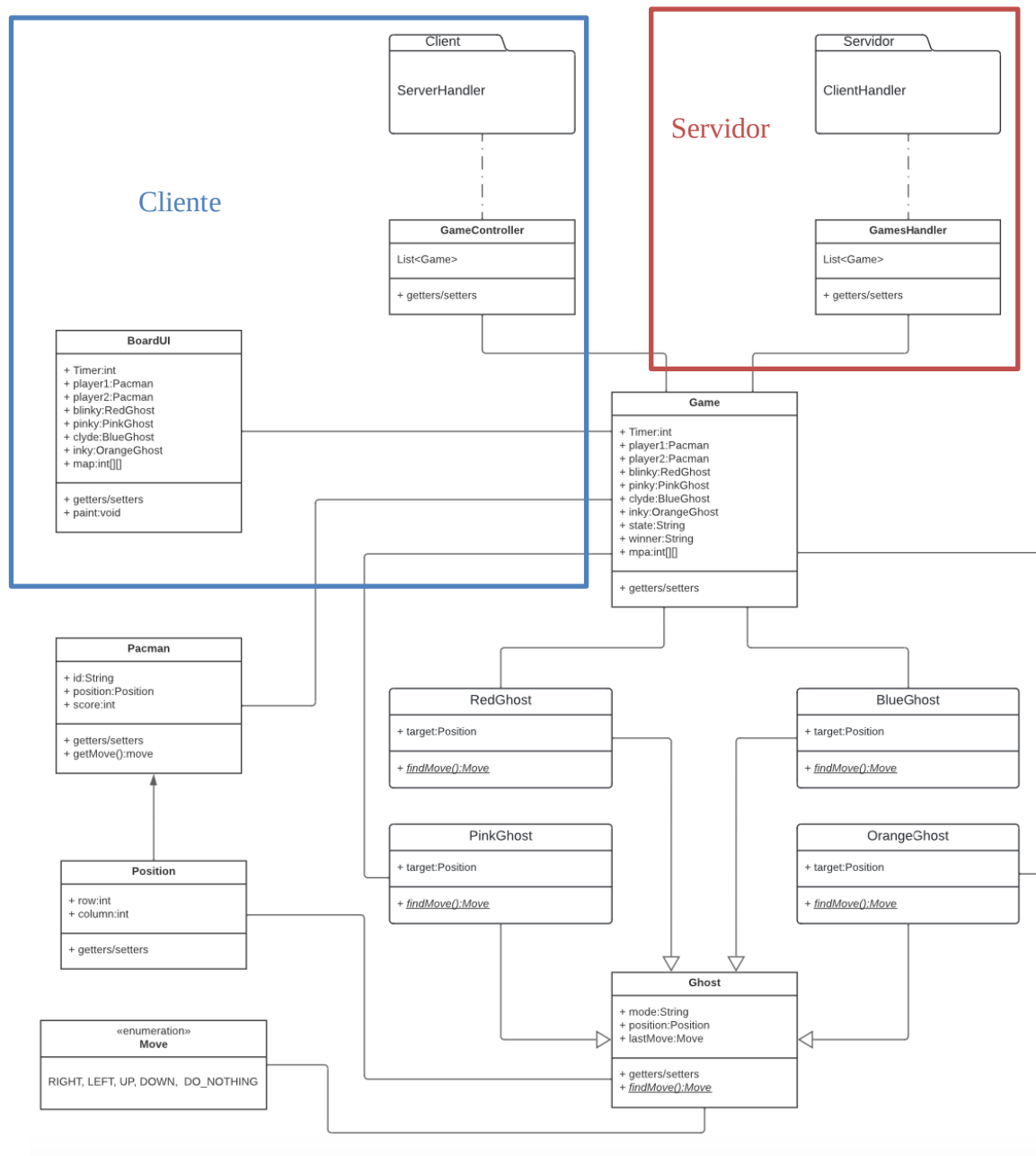


Figura 26. Diagrama de clases. Elaboración propia.

5.2.4 DIAGRAMA DE SECUENCIA

Dentro de los diagramas de comportamiento destaca principalmente los diagramas de secuencia, cuyo propósito principal es mostrar como interactúan diferentes componentes o

clases en tiempo real. Además, estos diagramas de secuencia muestran el flujo de mensajes de diferentes casos de uso.

En primer lugar, en la Figura 27, se ha representado el diagrama de secuencia entre los dos principales componentes del sistema, cliente y servidor. Esta es una representación a alto nivel, de cómo interactúan dichos componentes durante la ejecución del juego. Además, este diagrama muestra el intercambio de mensajes y métodos entre cliente y servidor, destacando el orden cronológico de los eventos.

La primera interacción, entre los componentes del sistema mencionados, se produce cuando un cliente solicita conectarse al servidor. Una vez conectado el cliente, este solicita crear una sesión de juego. El servidor tras crear la sesión de juego devuelve al cliente el id de sesión de su partida y espera a que otro jugador se conecte.

Una vez conectado el contrincante, el servidor avisa al cliente que inició la sesión que el juego va a comenzar, y le manda en el mensaje en un campo adicional el estado de juego. El cliente notifica al servidor que ha recibido su mensaje y en función de dicho estado de juego enviado por el servidor, el cliente envía su primer movimiento.

El servidor tras recibir los movimientos de ambos jugadores actualiza el estado de juego. Esta actualización del juego se realiza en cuatro subtareas, puesto que, para que los movimientos sean más suaves, estos se incrementan en 0.25. En la cuarta subtask al estar ya actualizado de manera correcta las nuevas posiciones, se pide a cada uno de los jugadores que vuelvan a enviar un nuevo movimiento en función del nuevo estado de juego. Así sucesivamente hasta que la partida finaliza.

En cualquier caso, en el próximo apartado de esta sección se va a explicar más en profundidad el proceso de comunicación entre cliente y servidor.

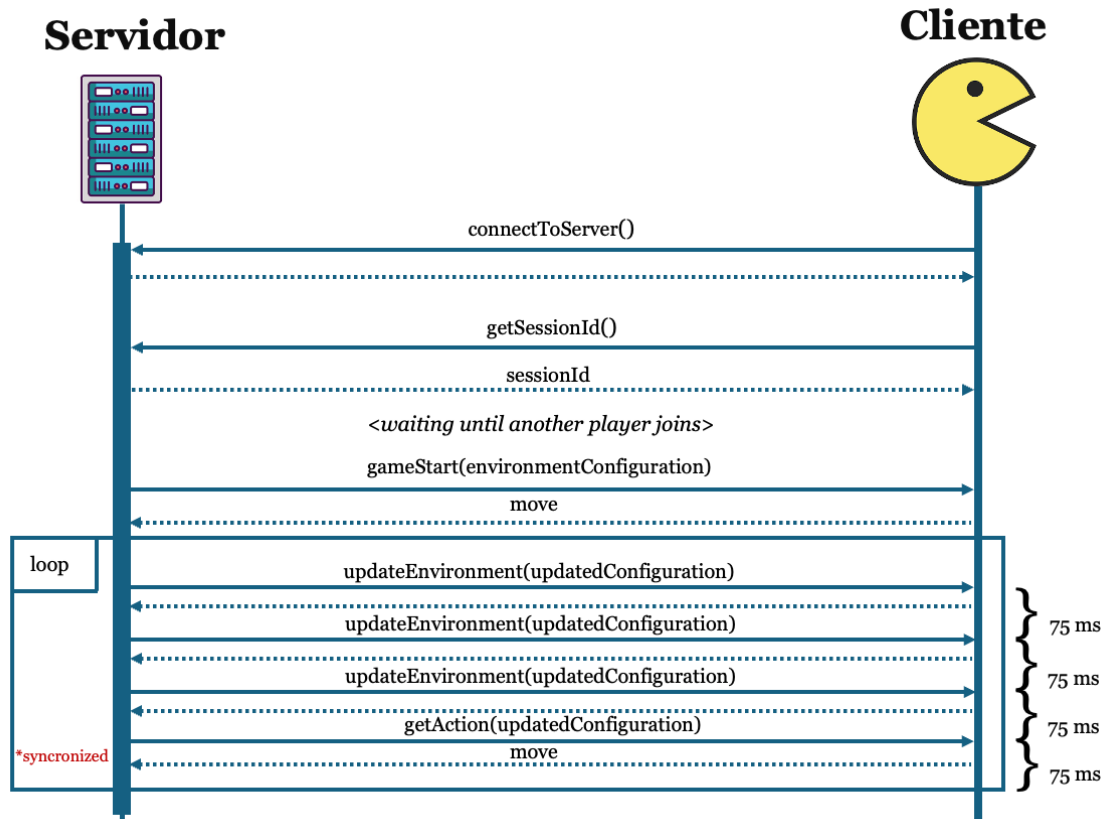


Figura 27. Diagrama de secuencia cliente-servidor. Elaboración propia.

En segundo lugar, se ha realizado un diagrama de secuencia, Figura 28, que ilustra el funcionamiento del servidor durante la conexión de los clientes y cómo este gestiona las partidas.

En primer lugar, cuando se instancia el servidor, este crea un gestor de partidas, ‘GamesManager’. Este gestor se encarga de manejar las diferentes partidas que se estén realizando.

A su vez, al iniciarse el servidor, este se queda escuchando posibles peticiones de conexión de diferentes clientes de forma indefinida. Al llegar una petición, el servidor la acepta y crea una instancia de un gestor individual de cliente, ‘ClientHandler’, que escucha y manda mensajes a través de un socket específico.

Cuando dos jugadores se han unido a la misma sesión, el juego comienza. El servidor crea el entorno de juego y lo manda a los participantes a través del gestor de partidas, quien a su vez lo reenvía a los respectivos gestores de cliente.

Estos clientes, con sus estrategias implementadas, mandan el primer movimiento que desean realizar. Además, como se ha explicado en el anterior diagrama, las actualizaciones de movimiento de los jugadores y de los fantasmas se realizan en incrementos, dando así una mayor fluidez al juego. Estos incrementos son transmitidos a los jugadores en intervalos de 75ms. Una vez transmitido el movimiento completo de los jugadores y de los fantasmas, los jugadores envían su próximo movimiento.

Debido a la magnitud del proyecto y la alta complejidad del servidor no se ha podido representar todas las clases y componente involucrados. Sin embargo, es importante destacar que, en el servidor, la clase ‘Game’ es la encargada de crear los fantasmas y de solicitar sus próximos movimientos, así como de gestionar la puntuación de los jugadores junto con el estado del mapa. Además, el servidor se encarga a su vez de validar todos los movimientos.

Una vez finalizada la partida, o lo que es lo mismo, una vez el temporizador de la partida llega a cero, el servidor manda a los jugadores de esa sesión el resultado final con el ganador.

Es importante mencionar que, todos los métodos y recursos compartidos en el lado del servidor deben ser síncronos, para evitar una pérdida de información.

Por último, en el siguiente diagrama de secuencia se ha separado el cliente del servidor a través de una línea discontinua de color rojo.

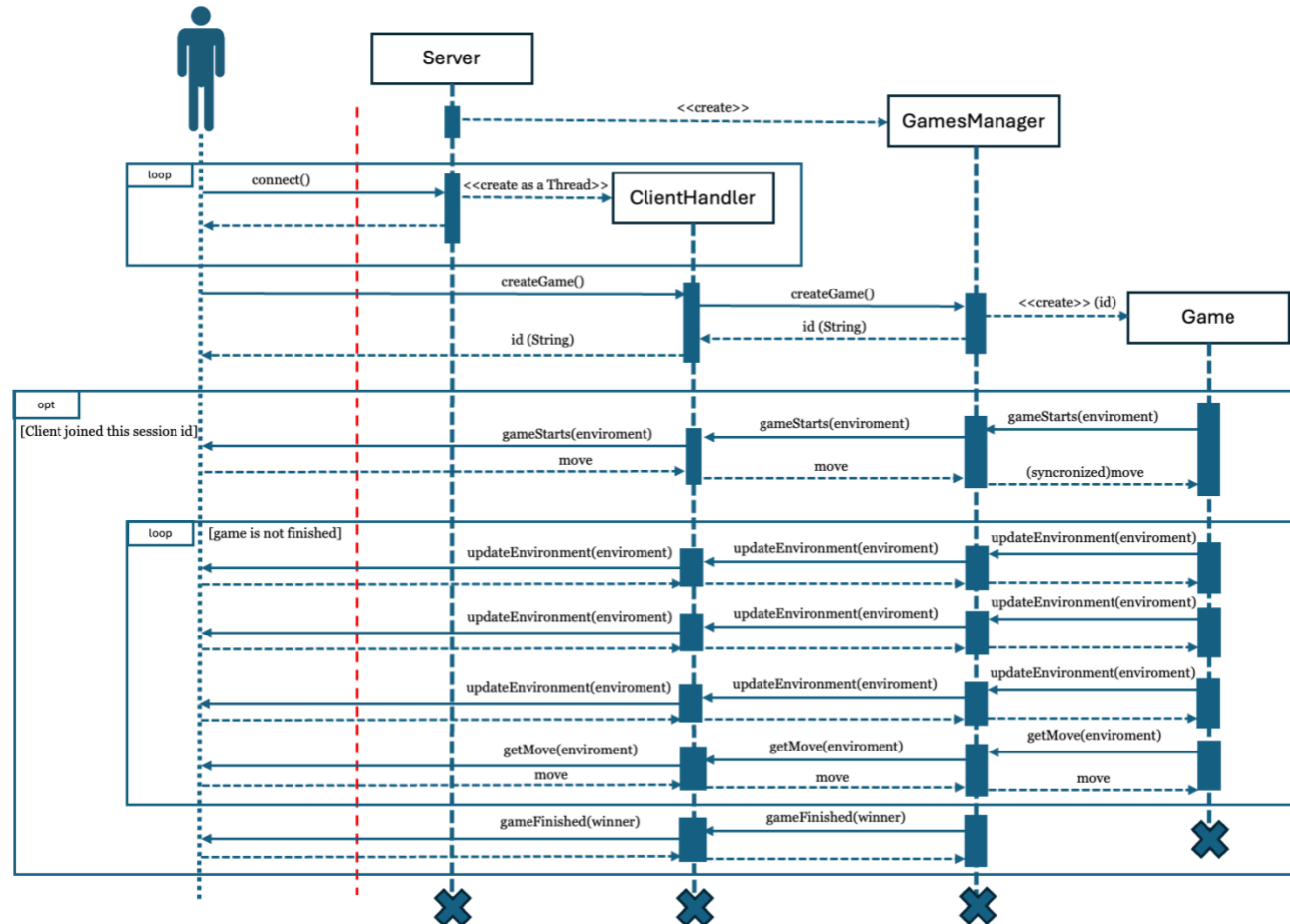


Figura 28. Diagrama de secuencia de partida (servidor). Elaboración propia.

5.2.5 COMUNICACIÓN CLIENTE-SERVIDOR

La comunicación entre cliente y servidor de este proyecto se va a realizar a través de la API de sockets en Java. La programación de sockets en Java permite que dos máquinas se puedan comunicar entre sí a través de la red. Además, estos sockets proporcionan un canal de comunicación continua y bidireccional. [47]

La comunicación se basa en el protocolo TCP. Este protocolo destaca por la entrega fiable y ordenada de datos. Además, TCP es un protocolo adecuado para este tipo de aplicaciones en tiempo real debido a su capacidad para gestionar conexiones persistentes y asegurar que todos los datos lleguen a su destino correctamente.

Cabe destacar que la transmisión de los datos se realiza de forma binaria, lo que permite una rápida y eficiente transmisión de datos.

A continuación, se adjunta un esquema, Figura 29, de cómo se realiza la comunicación entre cliente y servidor.

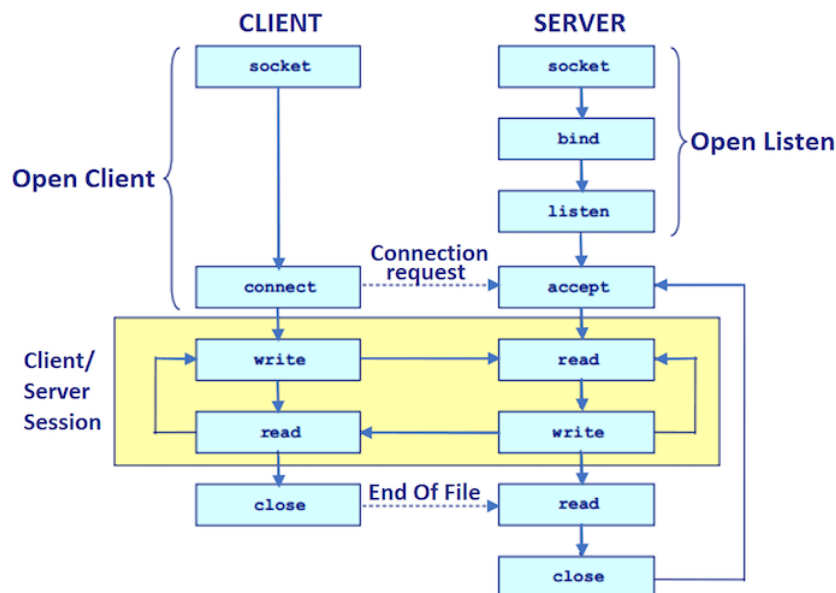


Figura 29. Comunicación cliente-servidor con sockets. Obtenido de [48]

Por otro lado, en este proyecto se trabaja con objetos complejos que la API de Java no soporta, por ejemplo, la clase ‘Position’. Por ello, es necesario el uso de JSON para la transmisión de los datos. Este formato destaca por ser ligero y por su facilidad de lectura.

Para la transmisión de los objetos complejos mencionados, es necesario llevar a cabo una serialización, es decir, convertir dicho objeto a una cadena JSON que pueda ser transmitida. Por otro lado, el receptor deberá llevar a cabo la deserialización del objeto complejo.

Para asegurar que tanto el cliente como el servidor sepan que datos se están recibiendo es necesario diseñar un protocolo de comunicación específico. Por ello, todos los mensajes enviados llevarán la misma estructura JSON, Figura 30.

El primer campo, ‘type’, indica el tipo de mensaje, es decir, si es una petición o una respuesta, para así poder procesar adecuadamente el mensaje y reaccionar conforme a este. El segundo campo, indica el motivo del mensaje o el evento que se está llevando a cabo en ese momento. Por último, el mensaje incluirá opcionalmente un campo de datos, ‘data’, en el que se incluirá cualquier información adicional necesaria para el correcto procesamiento del mensaje.



Figura 30. Estructura JSON del protocolo de comunicación entre cliente y servidor. Elaboración propia.

5.2.6 DISEÑO DEL JUEGO DE PAC-MAN

Una de las fases más importantes y críticas de este proyecto es el diseño del juego de Pac-Man junto con las modificaciones realizadas sobre el juego original. Estas modificaciones deben aumentar el nivel de competitividad entre los agentes de IA, y deben permitir una mayor variabilidad de estrategias.

A continuación, se describen los aspectos clave del diseño.

5.2.6.1 Objetivo

El principal objetivo de este juego es el diseño sofisticado de un agente de IA en Java que pueda superar a su oponente, en número de puntos, en el juego de Pac-Man multijugador. Cada partida tiene una duración de 60 segundos, durante los cuales los agentes de IA deben usar sus estrategias y algoritmos desarrollados para maximizar sus puntuaciones.

5.2.6.2 Duración

La duración predeterminada de las partidas es de 60 segundos, aunque esta puede ser fácilmente modificada por el organizador del torneo.

5.2.6.3 Construcción del mapa

El sistema ha sido desarrollado según las necesidades del principal ‘*stakeholder*’ de este proyecto, en este caso, Adam Gaweda. Por ello, el mapa ha sido generado como una estructura bidimensional de Strings, donde cada String está asociado a unas curvas específicas en Java.

Por tanto, el mapa del juego de Pac-Man de este proyecto está construido en mini celdas, Figura 31, donde cada celda corresponde a un String y este a su vez a una configuración gráfica en java específica. Esto permite una mayor flexibilidad y precisión para el diseño del laberinto. Además, de esta forma, es más fácil diseñar diferentes mapas, puesto que, únicamente se necesita seleccionar las celdas adecuadas.

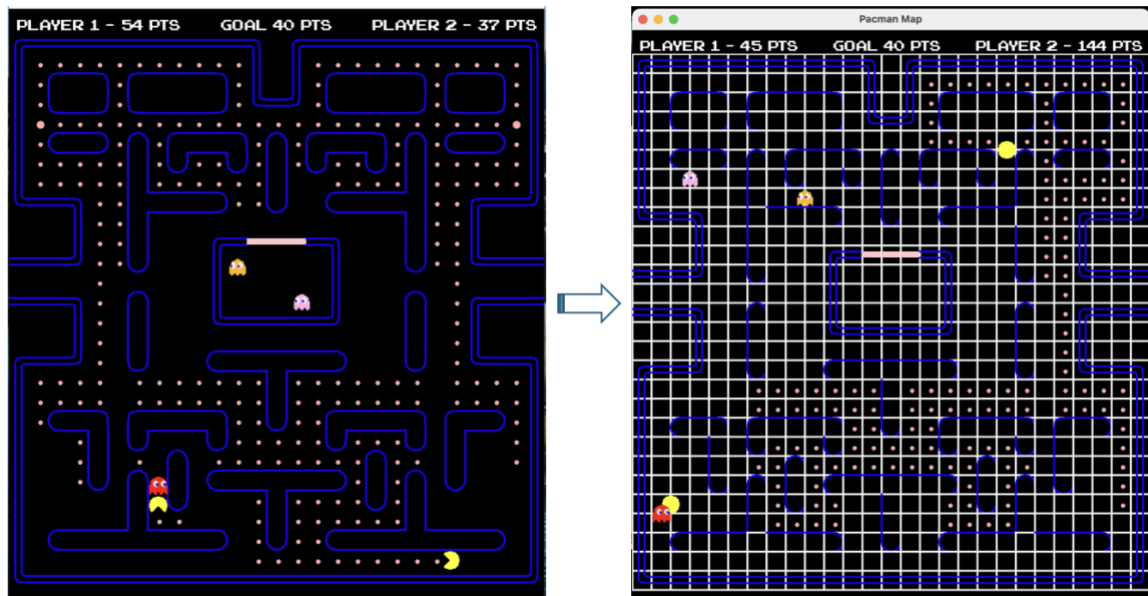


Figura 31. Construcción del mapa de Pac-Man. Elaboración propia.

A continuación, a modo de ejemplo, se adjuntan algunos ejemplos gráficos de mini celdas junto con sus String asociados.

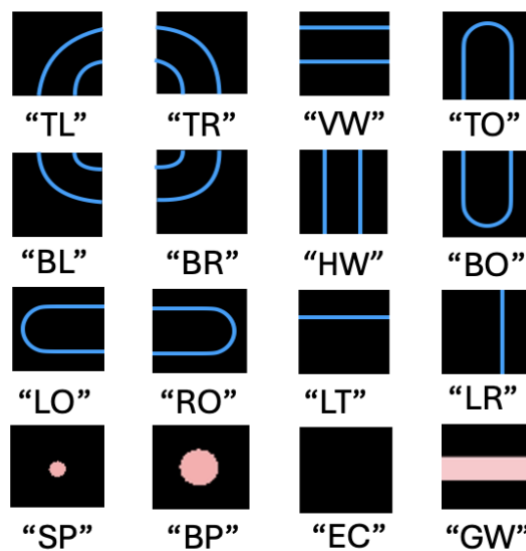


Figura 32. Ejemplo de mini celdas gráficas con sus respectivos String asociados. Elaboración propia.

Además, el proceso de construcción del mapa incluye los siguientes aspectos:

1. Definición de las dimensiones del laberinto: Ancho y alto del mapa, en función del tamaño de celda y del número de columnas y filas deseadas. El principal laberinto de este proyecto incluye un tamaño de celda predeterminado en 25x25 píxeles (ajustable), y 30 filas por 27 columnas.
2. Distribución de puntos y powerups: Importante la distribución estratégica de puntos para que sea más desafiante para los jugadores.
3. Ubicación de las zonas de inicio: Posicionamiento inicial de los jugadores, Pac-Man, y de los fantasmas.
4. Diseño de estructuras en el mapa: Caminos que permitan escapar a Pac-Man y por otro lado, que puedan permitir a los fantasmas emboscar a los jugadores.

5.2.6.4 Power-Ups

En esta variación del juego de Pac-Man se ha decidido introducir diferentes Power-Ups. Estos son elementos especiales que dan ventajas significativas a los jugadores que los consiguen. Además, estos elementos han sido diseñados para aumentar el nivel de competitividad y permitir el desarrollo de estrategias más avanzadas.

Los Power-Ups aparecen en momentos y lugares específicos del juego, y cada uno de ellos tiene un efecto único que pueden cambiar el curso de la partida.

- Aparición: El primer Power-Up aparece cuando el tiempo restante de partida baja a 40 segundos, y el segundo cuando quedan 20 segundos.
- Ubicación: Los Power-Ups aparecen en posiciones específicas adyacentes a la casa de los fantasmas.

Cada partida presenta 2 Power-Ups que son seleccionados aleatoriamente de un conjunto de 5 posibles, lo que añade un elemento de sorpresa y variabilidad a cada partida.

Los Power-Ups desarrollados son los que se muestran a continuación:



Congela al oponente por 5 turnos, impidiéndole moverse.



Intercambia la posición de Pac-Man con la del oponente



Bonus de 15 puntos para el jugador que recolecte este Power-up



Transfiere 5 puntos del oponente a la puntuación del jugador que lo recolecta



Regenera todos los puntos del laberinto que han sido consumidos.

5.2.6.5 Puntuación

El sistema de puntuación está diseñado para recompensar las habilidades y estrategias del jugador de manera equilibrada.

A continuación, se muestra un resumen de los puntos asignados a cada elemento:

-  Pellet: 1pt
-  Power pellet: 2 pts
-  Frightened ghost: 5 pts
-  Chase/Scatter: -3 pts
-  PowerUps: 0-15 pts

Por cada 'pellet' consumido se obtiene 1 punto. Por otro lado, los 'Power Pellet' activan el modo 'frightened' de los fantasmas y dan 2 puntos. Si Pac-Man es capaz de atrapar a un fantasma cuando este está en modo 'frightened', se obtendrán 5 puntos, por otro lado, si es

el fantasma quien atrapa a Pac-Man en su modo ‘scatter’ o ‘chase’ se deducirán 3 puntos del jugador atrapado. Por último, los Power-Ups varían entre 0 y 15 puntos, dependiendo del tipo de Power-Up consumido.

5.2.6.6 Pacman

Pac-Man, es el personaje principal, sobre el cual se realizan las implementaciones de IA. Dichas implementaciones deben determinar cuál es el movimiento más conveniente para su personaje dentro de las cuatro posibles direcciones, arriba, abajo, izquierda o derecha. Además, opcionalmente se podrá optar por no moverse y quedarse en la misma posición.

Con el fin de poder identificar a cada uno de los jugadores, en la interfaz gráfica, se ha asignado el color clásico amarillo al jugador 1 y el color naranja al jugador 2.



A diferencia de versión original, en esta variación Pac-Man no cuenta con un determinado número de vidas. Sin embargo, si es atrapado por un fantasma se le deducen tres puntos y se congelan sus movimientos durante tres turnos.

La principal estrategia recomendada es la combinación entre algunos de los siguientes algoritmos:

- A*: El algoritmo A* es un algoritmo de búsqueda que trata de encontrar la ruta óptima desde un punto inicial hasta un punto final. Por ello, este algoritmo es el más recomendado. Este algoritmo utiliza la siguiente función de coste: $f(n) = g(n) + h(n)$, donde $g(n)$ es el coste desde el punto inicial hasta n y $h(n)$ es la heurística que estima el coste desde n hasta el punto final. El algoritmo cuenta con una estructura de datos de prioridad con los diferentes nodos en los que se ordenan según su coste $f(n)$. De esta forma el algoritmo expande el nodo con menor coste. Para la implementación de $h(n)$, es recomendable experimentar con diferentes heurísticas

que ayuden a encontrar el camino óptimo a los ‘pellet’ o Power-Ups. Algunas heurísticas interesantes para el caso de uso de este proyecto podrían ser la distancia euclídea o la distancia de Manhattan. Por otro lado, para determinar el coste $g(n)$, sería interesante ajustarlo en función de los puntos o Power-Ups recolectados.

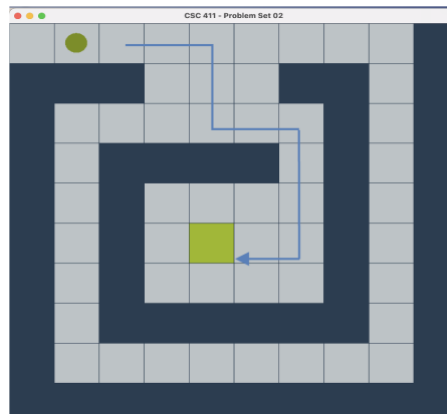


Figura 33. Ejemplo del algoritmo A*. Elaboración propia.

- Simulated Annealing: Este es un algoritmo probabilístico que puede usarse para encontrar una aproximación a la óptima global en un espacio de búsqueda grande. En el contexto de este proyecto, este algoritmo puede ser utilizado para desarrollar estrategias que decidan cuando es óptimo ir a por puntos, huir de los fantasmas o recolectar Power-Ups.

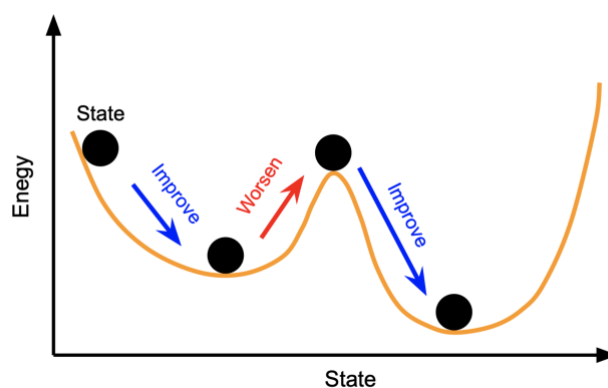


Figura 34. Gráfica del algoritmo Simulated Annealing. Obtenido de [49]

- Máquinas de estado finitas (FSM): Este es modelo de comportamiento que puede estar en uno de varios estados predefinidos y cambiar de estado en función de eventos

de la partida. Los estados podrían incluir recolectar puntos, huir, explorar o recolectar Power-Ups. El estado de operación variará en función de parámetros como la proximidad de fantasmas, disponibilidad de los Power-Ups o la cantidad de puntos restantes en el laberinto.

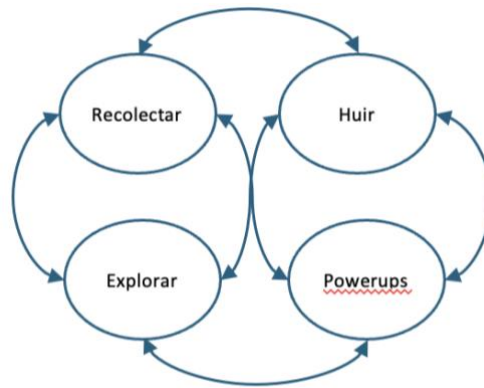


Figura 35. Ejemplo de máquinas de estado para Pac-Man. Elaboración propia.

- **Algoritmo MiniMax:** Este algoritmo es una técnica de decisión para obtener en el peor escenario el mejor resultado posible. En el juego de Pac-Man, este algoritmo podría ser utilizado para maximizar la puntuación del agente mientras que se minimiza la posibilidad de ser capturado por los fantasmas. Para ello, se construyen árboles de decisión que evalúan las posibles acciones del agente y de los fantasmas.

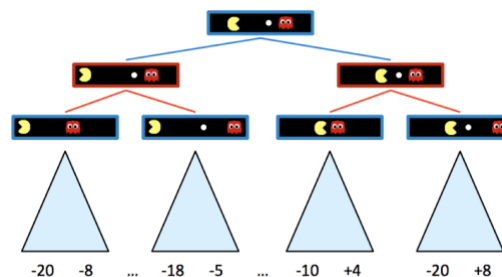


Figura 36. Ejemplo del algoritmo Minimax. Obtenido de [50]

- **Red Neuronal Profunda (Deep Q-Learning):** Este algoritmo es una técnica de aprendizaje por refuerzo que busca aprender una función Q, que puede estimar el valor de una acción en un estado dado. Para ello la red neuronal toma el estado del juego como entrada y produce un valor Q para cada posible acción. Por otro lado, la función Q se actualiza iterativamente en base a las recompensas recibidas y la

estimación de los valores futuros. Para el proceso de entrenamiento, el agente de usar un balance entre explorar nuevas acciones y usar las acciones que ya sabe que son buenas. Para ello, se utiliza una estrategia epsilon-greedy, donde el agente selecciona una acción aleatoria con una probabilidad de ϵ , y selecciona la acción con el mayor valor Q estimado con una probabilidad de $(1-\epsilon)$.

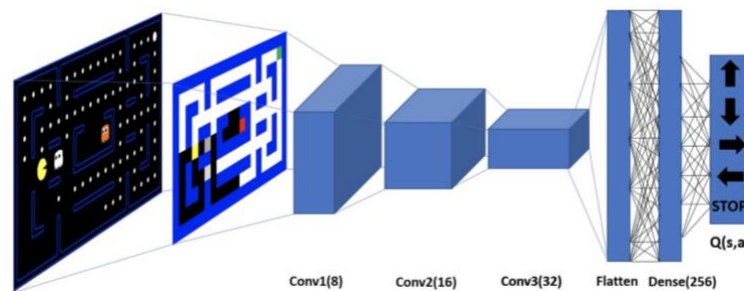


Figura 37. Ejemplo de red neuronal profunda (Deep Q-Learning) en Pac-Man. Obtenido de [51]

- **Sistemas Expertos:** Los sistemas expertos utilizan un conjunto de reglas predefinidas para tomar decisiones basadas en el conocimiento experto. En el juego del Pac-Man los sistemas expertos pueden ser usados para manejar situaciones específicas. Un ejemplo de ello podrías ser el siguiente: “Si un fantasma se encuentra a menos de tres celdas de mi posición y no tengo ningún punto cerca, entonces huir”

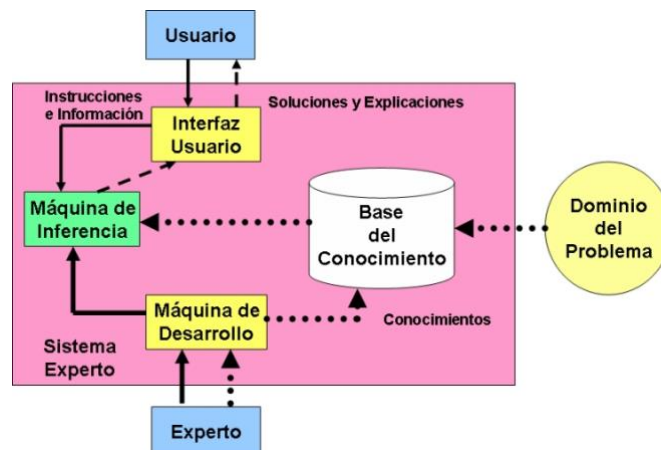


Figura 38. Ejemplo de estructura del sistema experto. Obtenido de [52]

- **Algoritmo D* Lite:** Este algoritmo es una variante del algoritmo A*, anteriormente explicada. D* Lite está diseñada para entornos dinámicos donde el mapa del entorno puede cambiar con el tiempo. Por ello, este algoritmo es especialmente útil en el

juego del Pac-Man, puesto que, los agentes se enfrentan continuamente a cambios en el laberinto. Estos cambios en el laberinto pueden ser la aparición de Power-Ups o el acercamiento de los fantasmas. Este algoritmo cuando detecta un cambio en el entorno, como la aparición de un fantasma, ajusta los costes y recalcula el camino óptimo desde la posición actual a la meta.

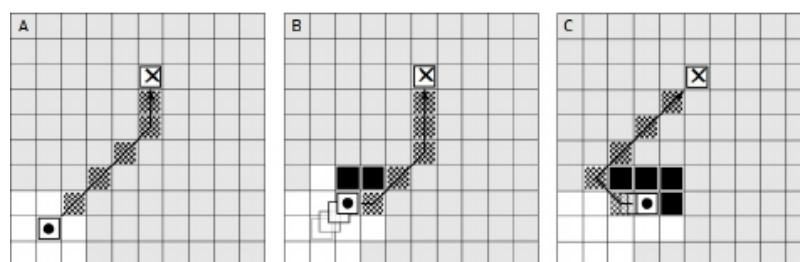


Figura 39. Ejemplo algoritmo D* Lite. Obtenido de [53]

5.2.6.7 Fantasmas

Por último, se va a explicar uno de los elementos cruciales de este juego, los fantasmas, debido a que desafían continuamente al jugador. Cada uno de estos fantasmas cuentan con una personalidad y comportamiento distinto, interactuando de forma única con Pac-Man. [30]

Los fantasmas alternan entre dos modos principales, dispersión ('scatter') y persecución ('chase'). Además, pueden entrar en un modo de pánico ('frightened') cuando un jugador consume un 'Power Pellet'.

Para explicar los cambios entre los distintos modos se ha construido una máquina de estados, Figura 40. Los cambios entre los modos de dispersión y persecución se producen en función de un temporizador. El fantasma está 10 segundos en modo dispersión y posteriormente 20 segundos en persecución. Esto se produce de forma indefinida hasta que uno de los jugadores consigue un 'Power Pellet', momento en el que los fantasmas pasan a su modo de pánico. En este modo permanecen durante 7 segundos, momento en el que pasan al modo de dispersión. Sin embargo, si durante el modo de pánico son atrapados por Pac-Man, dichos fantasmas volverán a su posicionamiento original dentro de la casa de los fantasmas. Debido

a esto, el temporizador es individual para cada uno de los fantasmas, puesto que, si un fantasma es comido y otro no, tendrán comportamientos y modos distintos.

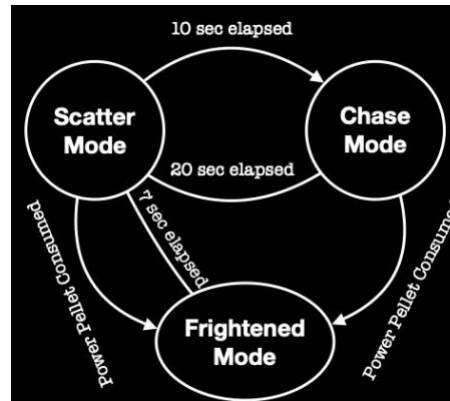


Figura 40. Máquina de estados de los modos de los fantasmas. Elaboración propia.

Antes de introducir a cada uno de los fantasmas, es necesario explicar dos reglas de movimiento de estos. Para explicar posteriormente estas reglas, se adjunta la siguiente imagen, Figura 41:

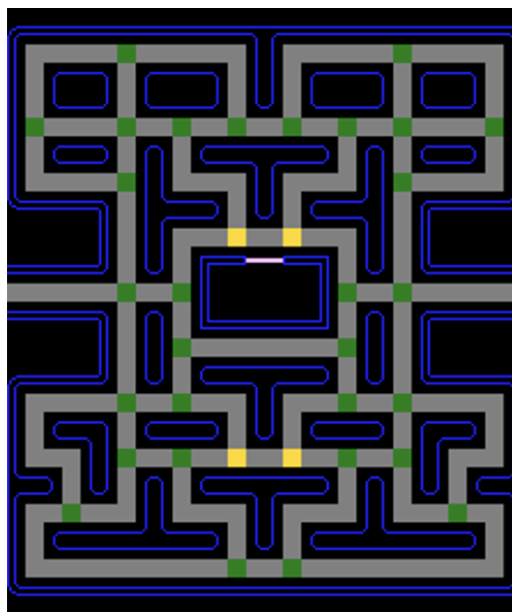


Figura 41. Reglas de movimiento de los fantasmas. Obtenido de [30]

La primera regla es, que un fantasma nunca podrá invertir su dirección, es decir, si va hacia la izquierda no puede darse la vuelta para ir a la derecha. De esta forma, las decisiones de

movimiento de estos fantasmas únicamente tendrán lugar en las intersecciones, casillas verdes.

En segundo lugar, las casillas amarillas presentan restricciones especiales. En dichas intersecciones los fantasmas no podrán ir hacia arriba, debiendo mantener el sentido de su movimiento.

Por otro lado, es importante mencionar como calculan cada uno de estos fantasmas sus objetivos en las intersecciones. Estos cálculos se realizan en línea recta desde las dos o tres celdas contiguas, tal y como se puede comprobar en la Figura 42. En caso de encontrarse a la misma distancia el orden de preferencia es el siguiente: arriba – izquierda – abajo, por tanto, los fantasmas nunca irán hacia la derecha en caso de igualdad.

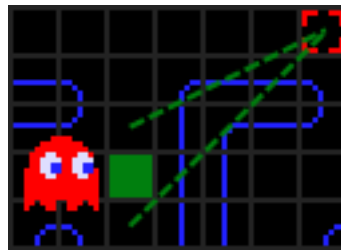


Figura 42. Celdas usadas para los cálculos de movimientos de fantasmas. Obtenido de [30]

Una de las principales desventajas de usar líneas rectas a los objetivos es, la posibilidad de escoger el camino más largo. Como se puede apreciar en la Figura 43, la celda de abajo a la izquierda del fantasma parece estar más cerca del objetivo, sin embargo, ese sentido conllevará un mayor número de celdas que recorrer.

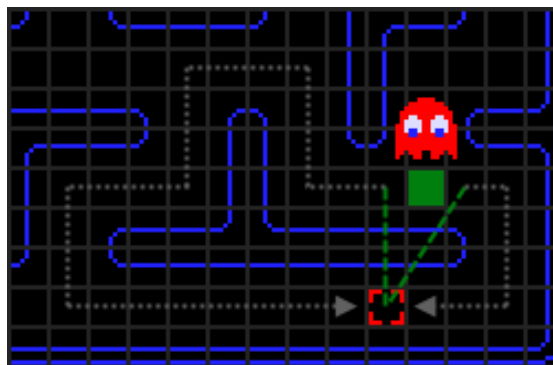


Figura 43. Ejemplo de cálculo no óptimo al objetivo de los fantasmas. Obtenido de [30]

Dentro del juego de Pac-Man se encuentran los siguientes cuatro fantasmas:



Como se ha mencionado anteriormente, cada uno de estos cuenta con una personalidad distinta. En primer lugar, se va a explicar las diferencias de estos fantasmas dentro del modo de dispersión ('scatter').

En el modo dispersión cada uno de los fantasmas tiene asignado una esquina del laberinto. Para ello, como se puede apreciar en la siguiente imagen, Figura 44, a los fantasmas se les pone como objetivo una celda inaccesible que esta fuera del mapa. Esto último combinado con la regla de movimiento de no invertir el sentido, hace que los fantasmas den vueltas alrededor de las estructuras de las esquinas, aunque en la mayoría de casos debido a la corta duración de este modo, los fantasmas no llegan a realizar ni siquiera una vuelta completa.

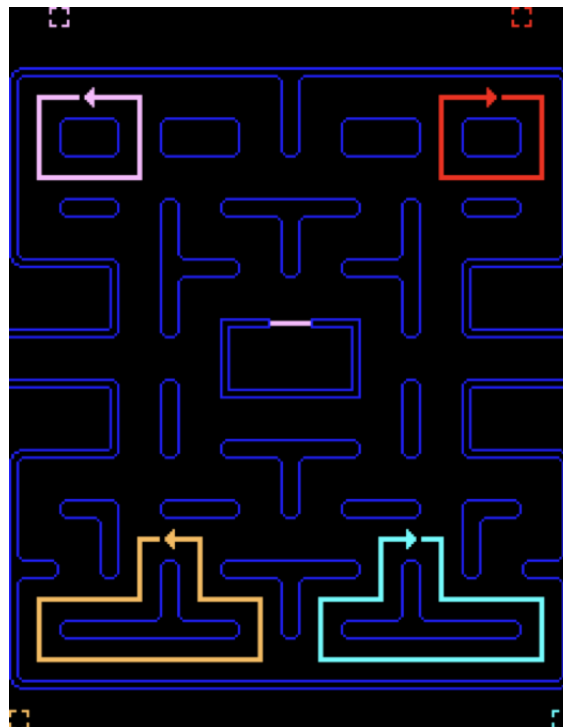


Figura 44. Modo dispersión. Obtenido de [30]

En segundo lugar, se va a explicar el modo persecución. En este modo, los fantasmas utilizan algoritmos diferentes para determinar la celda objetivo. Cabe destacar que, los fantasmas persiguen al jugador más próximo.

En primer lugar, el fantasma rojo, blinky, es el más agresivo. Este fantasma tiene como objetivo la posición actual del jugador más próximo, Figura 45.

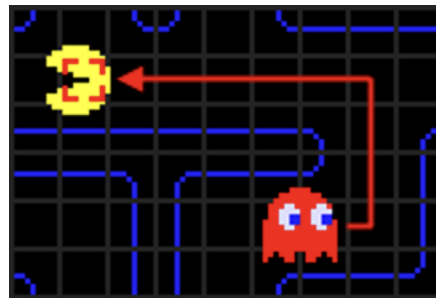


Figura 45. Modo persecución del fantasma rojo. Obtenido de [30]

En segundo lugar, el fantasma rosa, pinky, trata de emboscar a Pac-Man. Para esto escoge la celda situada 4 posiciones más adelantada del sentido de dirección del jugador más próximo, Figura 46. De esta forma pinky intenta anticiparse y colocarse en medio del camino del jugador.

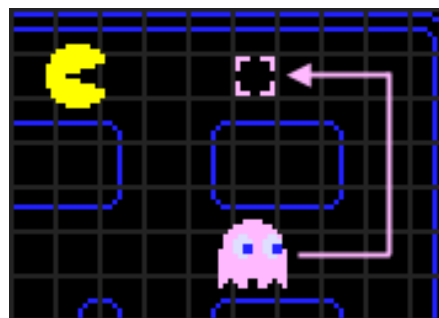


Figura 46. Modo persecución del fantasma rosa. Obtenido de [30]

En tercer lugar, el fantasma naranja, clyde, tiene dos comportamientos dependiendo de la distancia a la que se encuentre del jugador, Figura 47. Si clyde se encuentra a más de ocho celdas de distancia de Pac-Man se comporta como blinky, es decir, se mueve directamente a por el jugador. Sin embargo, si se encuentra a menos de ocho celdas se comporta como si estuviera en el modo de dispersión. Este fantasma tiene como objetivo confundir al jugador.

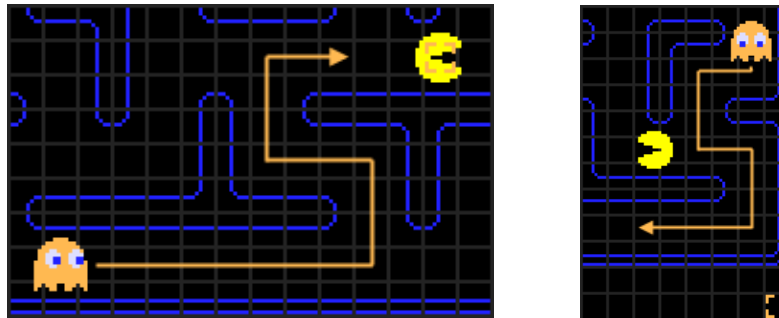


Figura 47. Modo persecución del fantasma naranja. Obtenido de [30]

En cuarto lugar, el fantasma azul, inky, tiene el comportamiento más complejo de todos. En primer lugar, este fantasma identifica la celda que está dos casillas por delante del sentido en el que se mueve Pac-Man. Una vez hecho esto, traza una línea imaginaria desde la posición del fantasma rojo, blinky, hasta la posición calculada anteriormente. Por último, este fantasma calcula una reflexión de dicha línea en la misma dirección. El cálculo de la celda objetivo se puede observar de manera gráfica en la siguiente imagen, Figura 48.

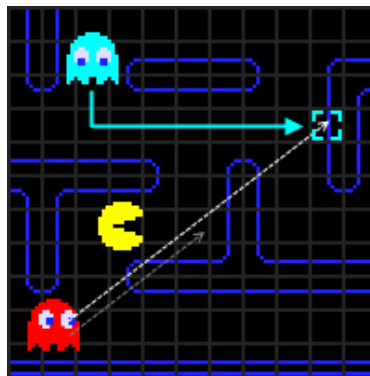


Figura 48. Modo persecución del fantasma azul. Obtenido de [30]

Por último, en el modo pánico, todos los fantasmas se comportan de la misma manera. Estos cambian de color al azul oscuro, Figura 49, y si son comidos por los jugadores regresan a su posición inicial. Por otro lado, si no son atrapados por los jugadores, estos parpadean durante los últimos dos segundos para indicar que se va a cambiar de modo a dispersión.



Figura 49. Diseño de fantasma en modo pánico. Obtenido de [54]

5.3 IMPLEMENTACIÓN

Una vez finalizada la etapa de diseño se va a comentar como se ha llevado a cabo la implementación de este. Hay que destacar, que, debido a la magnitud y complejidad del proyecto, la implementación del juego de Pac-Man multijugador con una arquitectura de cliente-servidor ha requerido una elaboración de código extenso y detallado, superando las 15.000 líneas de código. Debido a esto, en esta sección se presentará y explicará el código más relevante y representativo, para proporcionar una visión clara de los aspectos más críticos del sistema.

Como se ha explicado a lo largo de este documento, el objetivo principal es crear una arquitectura cliente-servidor que sea capaz de comunicar a dos jugadores y gestionar el entorno de partida en tiempo real, en este caso en el juego del Pac-Man. Para ello, como se ha mencionado en el diseño se han diseñado protocolos de comunicación personalizados, incluyendo serializadores de datos y, se ha llevado un estudio del juego de Pac-Man para su replicación y extensión, incorporando nuevas funcionalidades y ajustes que permitan una mayor competitividad. Además, como novedad, Pac-Man es controlado por una IA que debe ser diseñada por el jugador.

En las siguientes subsecciones se detallará la implementación de las principales funcionalidades para el correcto funcionamiento de este juego.

5.3.1 COMUNICACIÓN CLIENTE-SERVIDOR

La comunicación entre el cliente y servidor es la parte más crítica del sistema, puesto que, dos sistemas en diferentes entornos deben ser capaces de comunicarse y entenderse, en tiempo real. Como se ha explicado en el diseño, es fundamental asegurar una comunicación eficiente y fluida.

Con el fin de que el servidor pueda comunicarse con múltiples clientes de forma concurrente es esencial el uso de Threads de java. Por ello, el servidor utilizará un hilo para cada conexión con el cliente.

Además, la comunicación entre un cliente individual y el servidor se ha llevado con la API de sockets de java. Un socket es un punto final en una conexión bidireccional entre dos programas que se ejecutan en una red.

En java, el servidor crea un ‘ServerSocket’ que escucha en un puerto específico. Cuando un cliente envía una solicitud de conexión, este la acepta, y crea un ‘Socket’ para comunicarse con dicho cliente.

```
// Código de ejemplo para el servidor
ServerSocket serverSocket = new ServerSocket(12345);
while (true) {
    Socket clientSocket = serverSocket.accept();
    new Thread(new ClientHandler(clientSocket)).start();
}

// Código de ejemplo para el cliente
Socket socket = new Socket("localhost", 12345);
```

La transmisión de los datos se lleva a cabo a través de los flujos de entrada y de salida del socket.

```
// Enviar datos desde el cliente
OutputStream outputStream = socket.getOutputStream();
ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);
objectOutputStream.writeObject(data);

// Recibir datos en el servidor
InputStream inputStream = clientSocket.getInputStream();
ObjectInputStream objectInputStream = new ObjectInputStream(inputStream);
Object data = objectInputStream.readObject();
```

Por otro lado, como se explicado anteriormente, es necesario crear un protocolo personalizado para que el cliente y el servidor sepan entenderse. Este protocolo define el formato de los mensajes y los campos que deben incluirse para que ambos sistemas sepan interpretar el mensaje. La estructura del mensaje es la siguiente:

```
{
  "type": request/response,
  "action": [evento]
  "data": { campo : valor
    }
}
```

A continuación, se muestra un ejemplo de la estructura de mensaje cuando el servidor manda un petición al cliente con la actualización del entorno y, la respuesta del cliente.

```
{
  "type": request,
  "action": "getMove"
  "data": {
    "gameSessionId" : String
    "player1": {
      "position": {
        "row": int,
        "col": int
      }
      "move": Move,
      "points": int,
    },
    ...(same for player2)
    "ghost": {
      "red": {
        "move": Move,
        "mode": String,
        "position": {
          "row": int,
          "col": int
        }
      },
      ... (same for all the ghost)
    }
    "Map": [[string1,string2,...],[ string1,string2,...],[ string1,string2,...]]
    "Remaining Time": seconds
  }
}
```

Petición de movimiento enviada desde el servidor al cliente, con la actualización del entorno.

```
{
  "type": response,
  "action": "getMove"
  "data": {
    "gameSessionId" : String
    "NextMove" : Move
  }
}
```

Respuesta con el siguiente movimiento del cliente. (El servidor es capaz de identificar que jugador es)

Además, tanto en el cliente como en el servidor, en función de si el mensaje recibido es una petición o una respuesta, tienen funciones específicas para procesar cada tipo de mensaje y actuar conforme a ello.

```
if (type.equals("request"))
    processRequest(messageJson, out);
else if (type.equals("response"))
    processResponse(messageJson, out);
else
    processInvalidMessage(messageJson, out);

public void processResponse(JsonObject responseJson, PrintWriter out){

    String action = responseJson.get("action").getAsString();
    JsonObject data = responseJson.getAsJsonObject("data");
    switch(action) {
        case "getAction":
            Gson deserializer = new GsonBuilder()
                .registerTypeAdapter(Move.class, new MoveDeserializer())
                .create();
            Move nextMove = deserializer.fromJson(data.get("Next Move"),
                Move.class);
            String playerId = data.get("Player Id").getAsString();
            this.pacmanGame.setNextMove(nextMove, playerId);
            break;
        default:
            System.out.println("WARNING: Unknown command (" + action + ")");
            break;
    }
}

public void processRequest(JsonObject requestJson, PrintWriter out) throws
    IOException {

    String action = requestJson.get("action").getAsString();
    JsonObject data = requestJson.getAsJsonObject("data");

    if(action.equals("createSessionId")) {
        sendGameSessionID(out);
    }else if(action.equals("joinSessionId")) {
        String sessionId = data.get("Session Id").getAsString();
        sendJoinStatus(out, sessionId.toUpperCase());
    }else {
        System.out.println("WARNING: Not supported request");
    }
}
```

Funciones para procesar los mensaje en el lado del servidor.

Por último, debido a que los sockets no soportan directamente la transmisión de objetos complejos (clases personalizadas), ha sido necesario serializar los datos de estos objetos en un formato que pueda ser enviado y posteriormente deserializado por el receptor.

En este proyecto, se ha usado la biblioteca externa de GSON para serializar los datos a JSON.

```
Gson gson = new Gson();

// Serializar
String jsonData = gson.toJson(complexObject);

// Deserializar
ComplexObject complexObject = gson.fromJson(jsonData, ComplexObject.class);
```

Por otro lado, para manejar los objetos complejos como la posición de los fantasmas y Pac-Man, se han creado serializadores y deserializadores personalizados.

A continuación, se muestra, a modo de ejemplo, la serialización y deserialización de la clase posición.

```
// Serialización
public class PositionSerializer implements JsonSerializer<Position> {
    @Override
    public JsonElement serialize(Position src, Type typeOfSrc,
                                JsonSerializerContext context) {
        JsonObject positionObject = new JsonObject();
        // Position is sent in two different fields, row and col
        positionObject.addProperty("row", src.getRow());
        positionObject.addProperty("col", src.getCol());
        return positionObject;
    }
}

// Deserialización
public class PositionDeserializer implements JsonDeserializer<Position> {
    @Override
    public Position deserialize(JsonElement json, Type typeOfT,
                               JsonDeserializationContext context) throws JsonParseException {
        JsonObject jsonObject = json.getAsJsonObject();
        int row = jsonObject.get("row").getAsInt();
        int col = jsonObject.get("col").getAsInt();
        return new Position(row, col);
    }
}
```

5.3.2 DESARROLLO DEL JUEGO

El desarrollo del juego de Pac-Man ha sido un proceso complejo y detallado. El objetivo principal de este proyecto no ha sido únicamente recrear fielmente la jugabilidad y elementos visuales del juego, sino implementar también nuevas funcionalidades.

Cada componente del juego ha sido desarrollado de manera meticulosa, desde la construcción del mapa, pasando por la implementación de los personajes y de los fantasmas, hasta la gestión de puntuación y el diseño de la interfaz de usuario.

A continuación, se puede observar la distribución de las clases y paquetes que se ha llevado en el proyecto, Figura 50. La imagen de la izquierda representa los paquetes en el cliente. La imagen del medio representa como se han distribuido los recursos. Por último la imagen de la derecha muestra los paquetes y clases del servidor

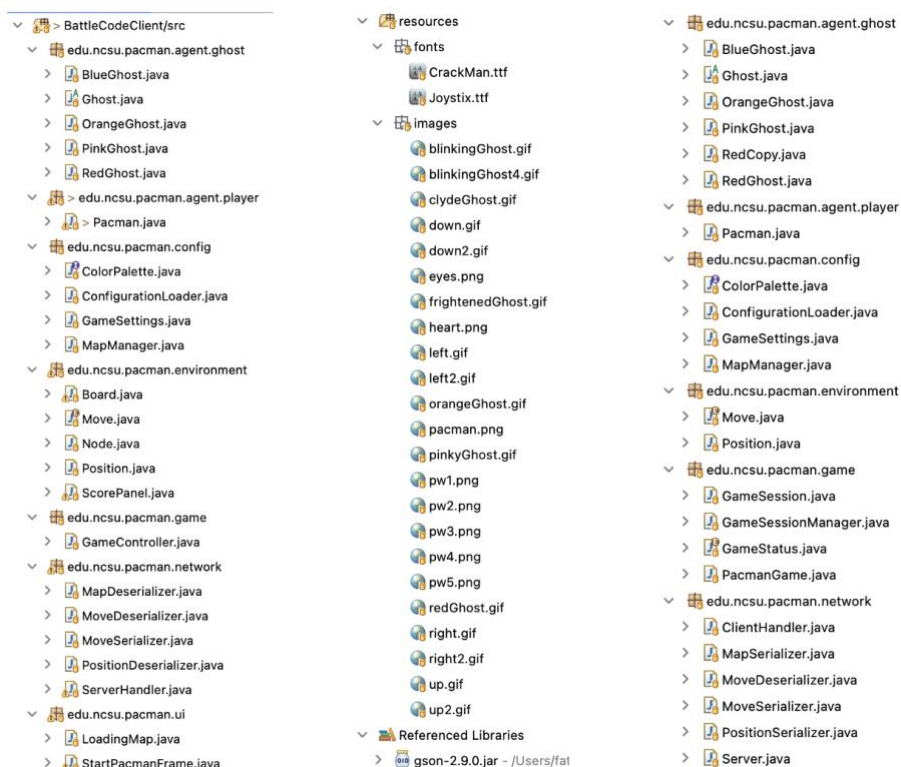


Figura 50. Diagrama de paquetes del proyecto. Elaboración propia.

En los siguientes apartados se explicará la implementación de la mayor parte de estos componentes. Aun así, se adjuntan los enlaces a los repositorios con el código del proyecto completo.

Servidor: <https://github.com/luca1015/BattleCodeServer/tree/master/BattleCodeServer>

Cliente: <https://github.com/luca1015/BattleCodeClient>

5.3.3 IMPLEMENTACIÓN DEL MAPA

La implementación del mapa ha sido llevada a cabo de manera fiel al juego original en lo referido a los elementos visuales. Sin embargo, se ha llevado a cabo una reestructuración de los elementos del mapa. Como se ha explicado anteriormente, el mapa está compuesto por un conjunto de mini celdas, asociadas cada una de ellas a un String. Esto permite flexibilidad para la elaboración de diferentes mapas. En este proyecto se ha llevado a cabo la siguiente configuración del mapa:

[illegible]

Figura 51. Implementación del mapa en formato String. Elaboración propia.

Cada una de estas cadenas tienen unas propiedades java distintas. Para la representación completa del laberinto, se va recorriendo y representado individualmente cada una de estas cadenas. A continuación, se muestra a modo de ejemplo, el proceso de representación del laberinto:

```
for (int i = 0; i < map.length; i++) {  
    for (int j = 0; j < map[i].length; j++) {
```

```
        if (map[i][j].equals("SP")) {
            g.setColor(GameSettings.POINT_COLOR);
            g.fillOval(CELL_SIZE * j + 12, CELL_SIZE * i + 12, 6, 6);
        }
        else if (map[i][j].equals("BP")) {
            g.setColor(GameSettings.POINT_COLOR);
            g.fillOval(CELL_SIZE * j + 10, CELL_SIZE * i + 10, 10, 10);
        }
        ..... (same for all the possible Strings)
    }
}
```

Hay que destacar que, siempre que se realizan cambios en el mapa, como actualizaciones de posicionamiento de los fantasmas o jugadores, estos deben de pasar una serie de pruebas de integración. En caso de no superar dichas pruebas no se aceptan los nuevos cambios.

5.3.4 ARCHIVO DE CONFIGURACIÓN

Por otro lado, se ha creado un archivo de configuración que es de vital importancia para poder gestionar diversos parámetros que afectan la jugabilidad y funcionalidad del juego. Este archivo permite ajustar y personalizar diferentes aspectos del juego de manera centralizada y flexible.

Algunos de estos ajustes incluyen: acceso a recursos, velocidad de juego, duración de la partida, tiempos de cambio de modo de los fantasmas, tamaño de las celdas del mapa o color de las estructuras del laberinto. A continuación, se muestra de forma parcial el archivo de configuración:

```
public class GameSettings {
    // Network configuration for game server.
    public static final int SERVER_PORT = 12345;

    // Graphical settings for rendering the game map.
    public static final int CELL_SIZE = 25; // Pixel size of map cells.
    public static final int ROWS = 30; // Number of rows in the game map.
    public static final int COLUMNS = 27; // Number of columns in the game map.

    // Color settings for various game elements.
    public static final Color BACKGROUND = Color.black; // Game background color.
    public static final Color SCORE_FONT = Color.white; // Color for score.
    ...
}
```

```
// Movement and timing settings.
public static final float MOVEMENT_INCREMENT = 0.25f; // Movement speed
public static final int DELAY = 75; // Update interval in milliseconds.

public static final String PW1 = "resources/images/pw1.png"; // PowerUp 1
public static final String PW2 = "resources/images/pw2.png"; // PowerUp 2
...

// Font settings for UI elements.
public static final String PUNTUATION_FONT = "/fonts/Joystix.ttf";
public static final String MESSAGE_FONT = "/fonts/CrackMan.ttf";

// Position of the PowerUps
public static final int PW1_ROW = 15;
public static final int PW1_COL = 11;
public static final int PW2_ROW = 15;
public static final int PW2_COL = 15;

...
}
```

5.3.5 ACTUALIZACIONES DE JUEGO

Las actualizaciones del juego que lleva a cabo el servidor es una fase crítica en el juego. Estas actualizaciones incluyen validación de movimientos o actualizaciones de puntuación y estado del juego.

El núcleo del sistema de actualización es el temporizador, puesto que, se encarga de realizar las comprobaciones mencionadas en intervalos regulares. Este intervalo es configurable en el archivo de configuración explicado en el apartado anterior. De forma predeterminada las actualizaciones se llevan a cabo cada 75ms.

Hay que destacar que, se ha hecho de Threads para poder gestionar múltiples partidas de forma concurrente sin que se produzca interferencias entre ellas, principalmente entre los temporizadores y recursos.

En el código de a continuación se muestra como se ha implementado el temporizador y las actualizaciones. En primer lugar, se comprueba si la partida ha finalizado. En caso de haber finalizado la partida se manda el resultado final del juego. En caso de seguir en curso, se

comprueba si se debe mandar una petición de movimiento a los jugadores o, si se debe mandar únicamente el estado del juego actualizado.

```
this.timer = new Timer(GameSettings.DELAY, new ActionListener() {

@Override
public void actionPerformed(ActionEvent e) {

    if(gameOver) {
        System.out.println("Thread of the game is ended");
        // If game over, the timer stops
        timer.stop();
        try {
            // Try to finish the game
            gameOver();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        return;
    }

    // Here, ghost mode are updated
    updateGhostMode();

    try {
        if(counter<4) {
            // Updating positions (smoother moves)
            if(checkGameStatus()) {
                sendUpdateEnvironment();
                counter++;
            }
        }else {
            // If the game hasn't finished, a new request message is sent
            to the client for getting their next move
            if(checkGameStatus()) {
                // Getting player's move
                sendGetAction();
                counter=1;
            }
        }
    } catch (IOException e1) {
        System.out.println("ERROR: Couldn't get next move of agents");
        e1.printStackTrace();
    }

}

});

// Timer initialization
timer.start();
```

Se han creado métodos específicos que comprueban que el siguiente movimiento deseado por parte de los fantasmas y jugadores es válido. En caso de ser válido se actualiza la posición y, en caso de ser inválido el movimiento se cancela y el fantasma o jugador se queda en la misma posición en la que se encuentra. Además, se han creado también métodos para actualizar la puntuación de los jugadores junto con el estado del mapa. Otro método independiente para actualizar el modo de comportamiento de los fantasmas. Por último, se ha creado también una función para comprobar el estado de la partida, es decir, si esta ha finalizado o sigue en curso.

Es importante destacar que, para actualizar el movimiento de los jugadores, este método tiene que ser sincronizado. Esto es necesario para que no se pierda información en caso de que ambos jugadores utilizar el mismo recurso al mismo tiempo.

```
// Sets the next move for a player, identified by their ClientHandler.  
public synchronized void setNextMove(Move nextMove, String playerId) {  
    validateMove(nextMove, playerId); // The move needs to be validated  
}
```

5.3.6 MÉTODOS DISPONIBLES PARA DESARROLLO DE IA

Para que los alumnos puedan implementar sus algoritmos de inteligencia artificial es importante que comprenda que funciones y recursos tienen a su disposición. Por ello, en este apartado se describe los datos y métodos que pueden ser usado para el desarrollo de diferentes estrategias.

Algunas consideraciones importantes son las siguientes:

- Los alumnos tienen 80 milisegundos para determinar y devolver su próximo movimiento. Si no se toma una decisión dentro de este límite, el juego interpretará la acción como DO_NOTHING.
- En el mapa del juego, x se refiere al índice de la columna e y se refiere al índice de la fila. Por tanto, para acceder a un elemento en el mapa `map[row][col] = map[y][x]`.

En la Figura 52, se muestra un ejemplo de los recursos disponibles para los alumnos. Cada elemento de la imagen está enumerado y corresponden a las siguientes funciones:

1. Información de los Power Ups:

- 1.1. `GameController.getPowerUps()`: Devuelve una lista de strings con los power-up disponibles en el mapa.
- 1.2. `GameController.getPwRow(String powerUp)`: Proporciona la fila en la que se encuentra un power-up específico
- 1.3. `GameController.getPwCol(String powerUp)`: Proporciona la columna en la que se encuentra un power-up específico

2. Información de los fantasmas:

- 2.1. `GameController.getGhosts()`: Devuelve un mapa con la información de cada uno de los fantasmas, donde cada clave es el color del fantasma y el valor es el objeto Ghost correspondiente.

3. Información del mapa:

- 3.1. `GameController.getIntMap()`: Se ha creado una función que convierte el mapa de Strings a un mapa de números enteros. Estos valores corresponden con el número de puntos que otorgan si un jugador se posiciona ahí. De esta manera, podemos encontrar los siguientes valores:

- 1 para paredes.
- 0 para caminos transitables sin recompensas.
- 1 para puntos individuales.
- 2 para Power Pellets.
- 3 para Power-Ups.

* Nota: En la imagen de referencia, las paredes se representan con un guion (-).

4. Información del oponente:

- 4.1. `GameController.getOpponent(this.getId())`: Obtiene la instancia de Pac-Man de tu oponente, proporcionando acceso a su puntuación y posición actual.

5. Tiempo Restante:

5.1. GameController.getRemainingTime(): Indica el tiempo restante para que finalice la partida

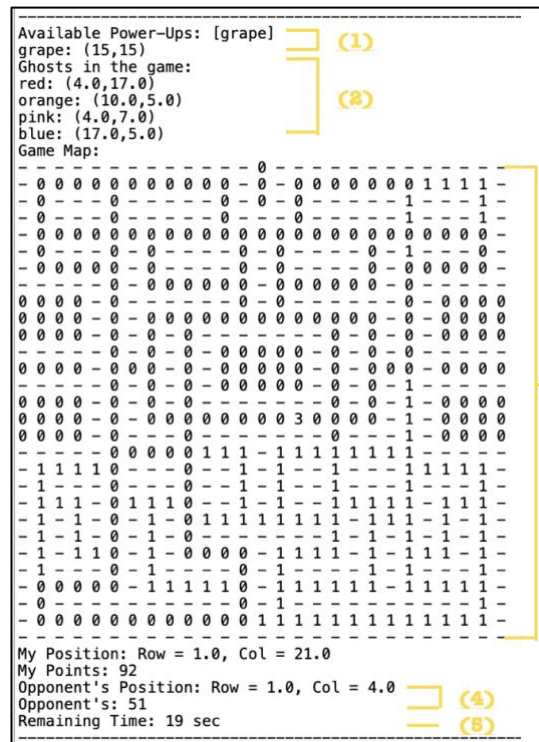


Figura 52. Recursos disponibles para las implementaciones de los alumnos. Elaboración propia.

Haciendo uso de los métodos y datos descritos, el jugador debe modificar el método `getMove()` de la clase Pac-Man para devolver su próximo movimiento.

5.3.7 DESARROLLO PROPIO DE ALGORITMOS PARA AGENTE IA

El desarrollo propio de IA ha sido un proceso largo, meticuloso y detallado, cuyo código se encuentra adjunto en el ANEXO II: Implementación propia de IA.

Mi estrategia se basa en una combinación de algoritmos y técnicas avanzadas para determinar el mejor movimiento en cada actualización de juego.

Entre los algoritmos usados destaca una adaptación de A*. En esta adaptación, la ruta se calcula de la misma forma que en A*, pero en cada celda por la que va avanzando se va

recalculando y actualizando la ruta. Además, a diferencia del algoritmo D* Lite, en esta estrategia el objetivo puede cambiar.

Por otro lado, se ha desarrollado también una lógica que evalúa y reacciona al movimiento de los fantasmas. De esta forma, si alguno de los fantasmas supone una amenaza inmediata se ajusta el objetivo para huir de estos. Por otro lado, si estos fantasmas se encuentran en modo pánico y Pac-Man está cerca de ellos, intentará capturarlos.

La estrategia desarrollada incorpora también un análisis continuo del estado del juego a través de la búsqueda de power-ups y el análisis del tiempo restante. De esta forma, si el tiempo de aparición de un power-up se acerca, mi agente se dirigirá hacia la zona en la que aparezca. Por otro lado, si no hay power-ups, se intenta priorizar los 'Power Pellet, siempre y cuando estos no se encuentren demasiado lejos.

5.3.8 CONTENERIZACIÓN DEL SERVIDOR

Para asegurar una implementación robusto y replicable se ha utilizado Docker para encapsular el código en un contenedor Docker permite empaquetar el servidor junto con todas sus dependencias en una imagen ligera y portable, lo que facilita su despliegue en cualquier entorno. Es importante destacar que, son los archivos compilados los que se han copiado al contenedor. A continuación, se adjunta el código del Dockerfile.

```
# Using a Java base image.
FROM openjdk:11-jre-slim

# Setting the working directory inside the container
WORKDIR /app

# Copying the .class files and the external library needed
COPY bin /app/bin
COPY gson-2.10.1.jar /app

# Exposes the port that the server uses
EXPOSE 12345

# Sets the working directory where the .class files are located
WORKDIR /app/bin

# Command to run the server
CMD ["java", "-cp", ".,/app/gson-2.10.1.jar", "edu.ncsu.pacman.network.Server"]
```

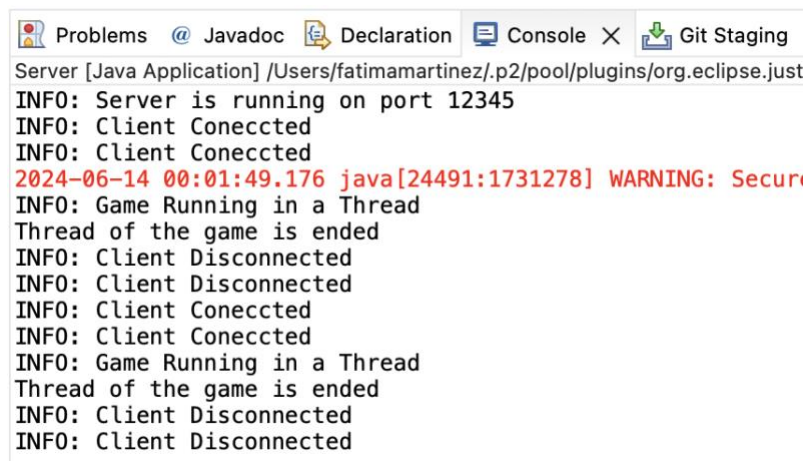
Para construir y ejecutar el contenedor Docker es necesario ir a la ubicación del DockerFile y construir la imagen. Una vez construida la imagen se ejecuta el contenedor.

```
docker build -t pacman-server .  
docker run -p 12345:12345 pacman-server
```

Capítulo 6. ANÁLISIS DE RESULTADOS

En este capítulo se van a presentar los resultados más relevantes obtenidos durante el desarrollo de este proyecto.

Uno de los resultados más importantes logrados con este proyecto es la creación de una infraestructura cliente-servidor eficiente. El sistema que se ha desarrollado permite una comunicación en tiempo real fluida, sincronizando correctamente las acciones de dos jugadores en una misma partida. Además, la infraestructura desarrollada es capaz de gestionar varias partidas de forma simultánea.



```
Problems @ Javadoc Declaration Console X Git Staging
Server [Java Application] /Users/fatimamartinez/.p2/pool/plugins/org.eclipse.just
INFO: Server is running on port 12345
INFO: Client Coneccted
INFO: Client Coneccted
2024-06-14 00:01:49.176 java[24491:1731278] WARNING: Securi
INFO: Game Running in a Thread
Thread of the game is ended
INFO: Client Disconnected
INFO: Client Disconnected
INFO: Client Coneccted
INFO: Client Coneccted
INFO: Game Running in a Thread
Thread of the game is ended
INFO: Client Disconnected
INFO: Client Disconnected
```

Figura 53. Gestión de clientes y partidas en el servidor. Elaboración propia.

Por otro lado, se puede comprobar cómo se ha conseguido llevar con éxito una pantalla de carga del juego, Figura 54. Esta pantalla sirve para llevar a cabo de forma paralela la conexión con el servidor.

Al inicio de esta pantalla de carga hay siete fantasmas en modo pánico situados en medio del laberinto. Pasados unos segundos aparece Pac-Man para comerse a estos fantasmas individualmente, mostrando en su lugar el mensaje de loading. Finalmente, una vez desaparecido Pac-Man, los fantasmas reaparecen en modo persecución, intentando atrapar a Pac-Man.

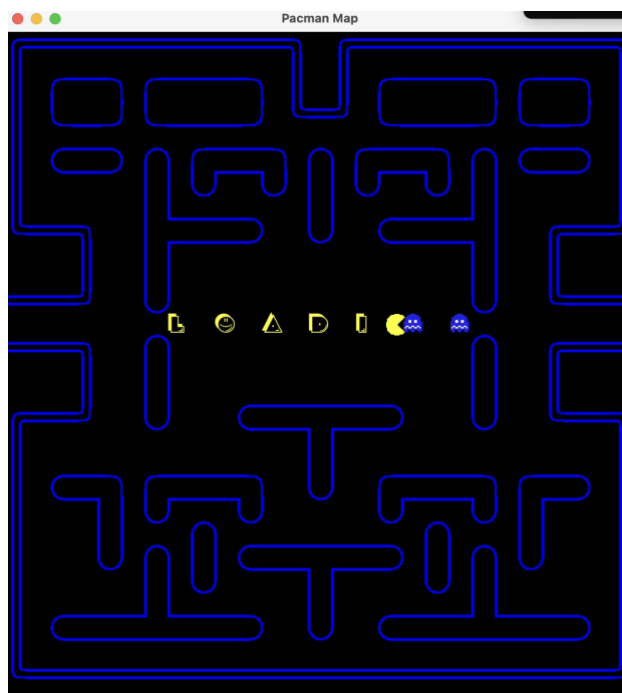


Figura 54. Pantalla de carga del juego. Elaboración propia.

Una vez realizada con éxito la conexión con el servidor, el usuario avanza a la pantalla de inicio, Figura 55. En esta pantalla, el jugador tiene la opción de crear una de partida o de unirse a una sesión existente.

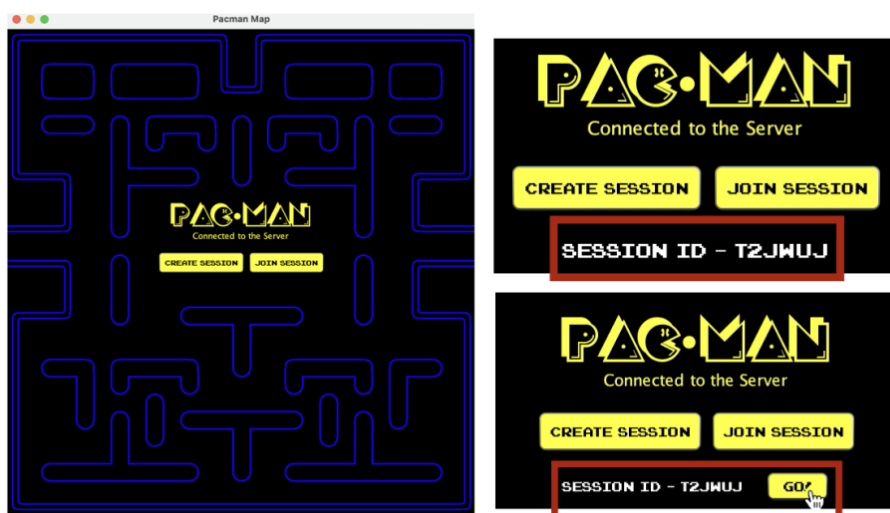


Figura 55. Pantalla de inicio del juego. Elaboración propia.

Una vez se haya unido un segundo jugador a la sesión, el juego se inicia, dando lugar a la siguiente pantalla de ‘gameplay’, Figura 56. Aquí se puede observar la estructura del mapa diseñado y construido a través de los diferentes Strings. Además, se muestra el posicionamiento inicial predeterminado tanto para los jugadores como para los fantasmas. Por otro lado, se puede comprobar que la duración de la partida es de 60 segundos. Por último, mencionar que el mapa contiene 204 ‘pellets’ y 4 ‘Power Pellets’.

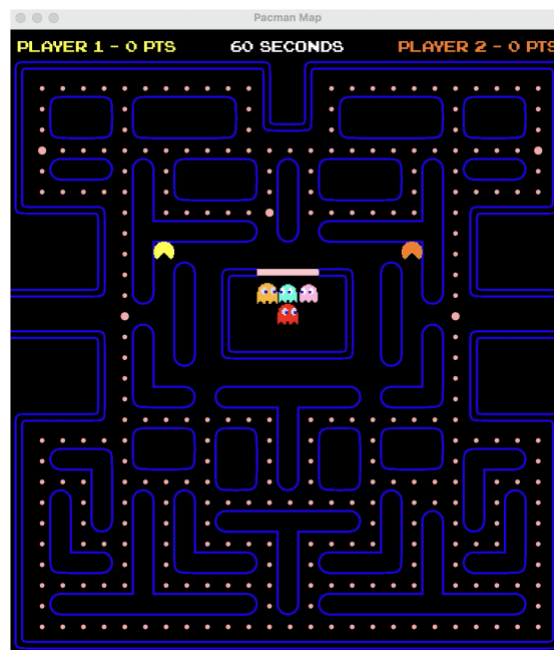


Figura 56. Pantalla del gameplay. Elaboración propia.

La siguiente imagen, Figura 57, contiene dos capturas independientes. La captura de la izquierda se ha adjuntado para ilustrar como cambia el diseño de los fantasmas al entrar en modo pánico y, como se ve cuando es atrapado por uno de los jugadores (dos ojos). Por otro lado, la captura de la derecha ilustra la ubicación del primer power-up, tras haber superado los 40 segundos de juego restante. Además, en esta captura se puede apreciar como mi agente, player2, va en busca de recolectar este potenciador.

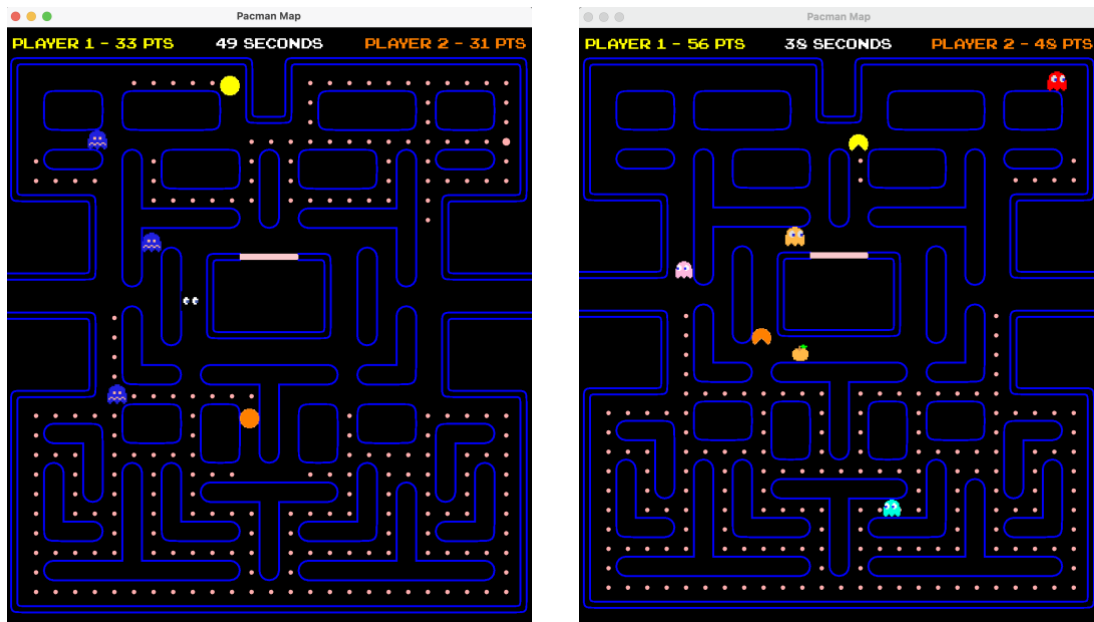


Figura 57. Fantasmas en modo pánico y ubicación de power-up en el laberinto. Elaboración propia.

Como última pantalla del juego del Pac-Man se adjunta la siguiente imagen, Figura 58. Tras finalizar la partida, a cada jugador se le manda su resultado final, en este caso el jugador 1 ha ganado la partida 116 puntos a 44 puntos.



Figura 58. Pantalla con resultado final. Elaboración propia.

Las siguientes dos figuras, Figura 59 y Figura 60, demuestran que el servidor ha podido ser encapsulado correctamente en Docker, creándose correctamente la imagen y, que se ha podido ejecutar correctamente en un entorno aislado. Esto permite que el servidor pueda funcionar en cualquier entorno de forma consistente e independiente de la configuración del sistema.

☐	Name	Tag	Status	Created	Size	Actions
☐	pacmanserver 72e757232acd	latest	In use	23 days ago	223.9 MB	▶ ⋮ 🗑️

Figura 59. Creación de la imagen del servidor en Docker. Elaboración propia.

```
~ — com.docker.cli ▶ docker run -p 12345:12345 pacman
Last login: Sun Jun  2 09:58:32 on console
[luca@MBP-de-Lucas ~ % docker run -p 12345:12345 pacmanserver
INFO: Server is running on port 12345
```

Figura 60. Ejecución del contenedor Docker. Elaboración propia.

Por otro lado, debido a las restricciones de tiempo, no se ha podido llevar a cabo una competición de IAs este curso académico. Sin embargo, para demostrar el espíritu competitivo y las capacidades de las implementaciones desarrolladas, se ha organizado un combate especial entre mi implementación, anteriormente explicada, y la implementación del director de este proyecto en Estados Unidos, el Doctor Adam Gaweda.

Para conmemorar este combate, se ha creado un cartel que, además de anunciar y destacar la batalla, también servirá como presentación del torneo que se va a realizar el próximo curso académico. En dicho torneo, se invitará a los nuevos participantes a intentar superar al ganador de esta batalla, incentivando así la innovación y desarrollo de nuevas estrategias.

A continuación, se adjunta el cartel diseñado:



Figura 61. Cartel de presentación del torneo. Elaboración de Adam Gaweda.

El combate llevado a cabo el 5 de junio entre ambas implementaciones dejó resultados muy interesantes. En primer lugar, el combate se realizó al mejor de 9 partidas. Mi implementación fue capaz de ganar la totalidad de estas partidas. Inicialmente, esto parecía indicar una clara superioridad de mi estrategia. Sin embargo, tras realizar un análisis más en profundidad, se descubrió la existencia de un sesgo significativo hacia el jugador 1, posicionado en la parte izquierda de la casa de los fantasmas al inicio de la partida.

En un segundo combate a 9 partidas, en el que mi implementación era el jugador 2, mi agente fue capaz de ganar únicamente 3 partidas. Esta ventaja se debía a la distribución del posicionamiento inicial de los fantasmas junto con el diseño de los algoritmos.

Al principio de las partidas los fantasmas se movían rápidamente hacia sus esquinas. Pero al modificar la distribución de los fantasmas, respecto a la versión original, se creó una desventaja. Como se puede comprobar en la siguiente imagen, Figura 62, los fantasmas no están distribuidos de forma simétrica respecto a sus esquinas. Esto hace que al principio algunos fantasmas se aproximen antes a un jugador.

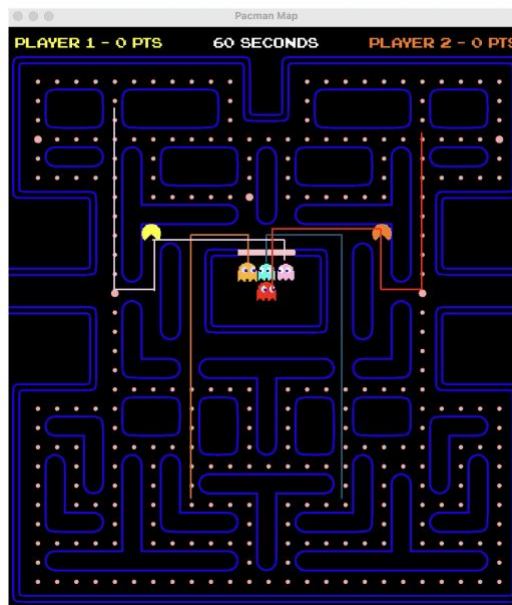


Figura 62. Sesgo en el comportamiento de los fantasmas. Elaboración propia.

Esto junto con la proximidad de los ‘Power Pellet’, situados a la mitad del laberinto, con respecto a los jugadores, propició una falta de equidad de condiciones entre los jugadores.

Por ello, en respuesta a este descubrimiento, se realizaron una serie de modificaciones. En primer lugar, los ‘power pellet’ situados en la mitad del laberinto fueron desplazados a la parte inferior del mapa. En segundo lugar, los fantasmas fueron redistribuidos de forma simétrica respecto a sus esquinas. Estos cambios se muestran a continuación, en la Figura 63.

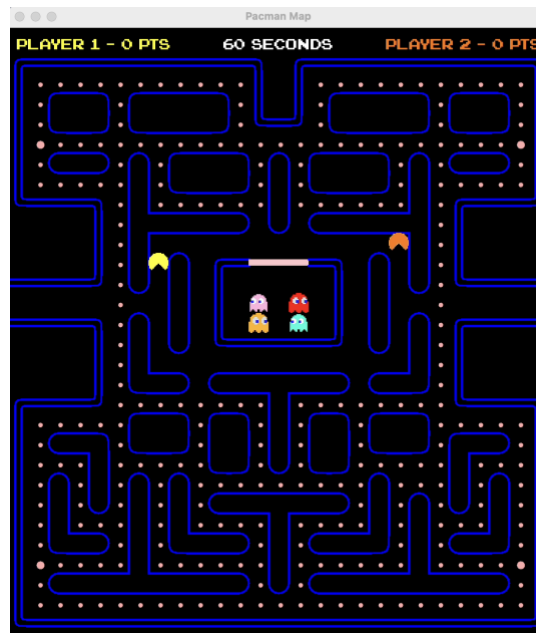


Figura 63. Corrección de posición de los fantasmas y 'Power Pellets'. Elaboración propia.

Por último, se configuró un temporizador a los fantasmas, para que estos empiecen a avanzar pasados los dos primeros segundos de partida. En las siguientes figuras se pueden apreciar como los fantasmas empiezan a avanzar pasados los 58 segundos y, como estos se mueven de forma simétrica hacia ambos lados.

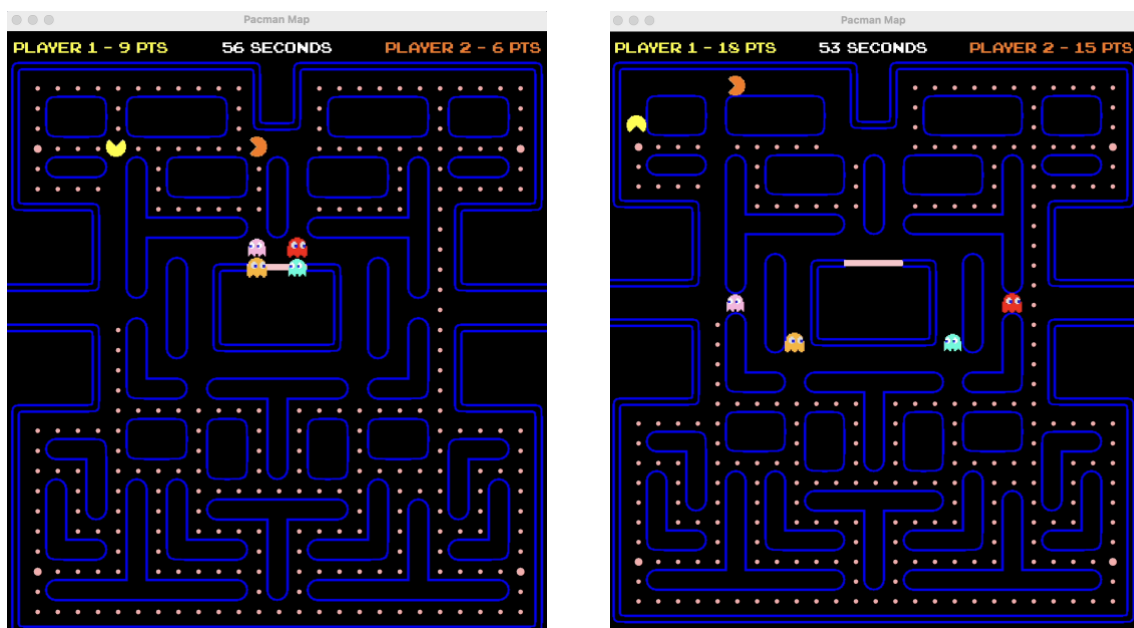


Figura 64. Comprobación de corrección de errores. Elaboración propia.

Al probar el sistema, con dichas modificaciones, se descubrió un nuevo pequeño bug, el primer jugador empezaba sus movimientos uno antes que el segundo jugador. Por ello, para corregir este fallo, cuando se inicia la partida no se cuenta el primer movimiento, de esta forma ambos jugadores comenzaran a moverse al mismo tiempo. Para ilustrar este fallo se adjuntan las siguientes dos capturas. La primera de ellas (imagen de la izquierda), hace referencia al problema que existía antes de eliminar el primer movimiento de los jugadores. La segunda de ellas muestra como gracias a la corrección, los jugadores empiezan a moverse al mismo tiempo, teniendo así igualdad de condiciones en la partida.

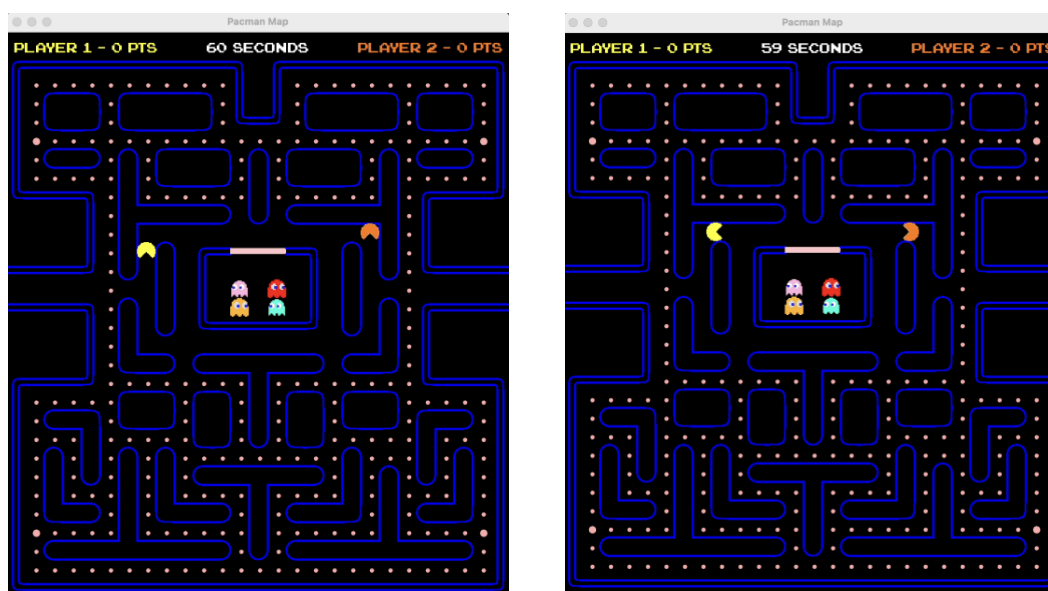


Figura 65. Corrección fallo de movimiento en el jugador 1. Elaboración propia.

Una vez solucionados los fallos y asegurada la igualdad de condiciones, se llevó a cabo un último combate final. Antes de mostrar y analizar los resultados obtenidos, es necesario explicar de forma breve la implementación del doctor Adam Gaweda.

La implementación llevada a cabo por el Adam se centra en optimizar los movimientos de su agente a través del algoritmo A*, determinando el camino más corto hacia el objetivo seleccionado. Para determinar el objetivo, Adam diseñó un algoritmo de búsqueda con radio variable para localizar los puntos y power-ups. En la siguiente imagen, Figura 66, se muestra una visualización de su estrategia, donde se muestran en rojo, las circunferencias que realizaría su algoritmo.

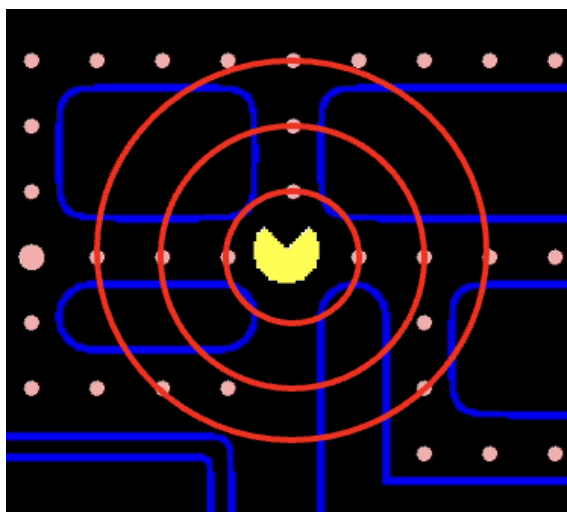


Figura 66. Implementación del doctor Adam Gaweda. Elaboración propia.

Por último, en la siguiente tabla se muestran los resultados obtenidos:

	Player 1	Player 2	Ganador	Resultado Final
1	Lucas	Adam	Lucas	141-106
2	Lucas	Adam	Lucas	147-96
3	Lucas	Adam	Adam	113-132
4	Lucas	Adam	Lucas	135-102
5	Lucas	Adam	Adam	120-137
6	Adam	Lucas	Adam	140-125
7	Adam	Lucas	Lucas	114-138
8	Adam	Lucas	Lucas	102-145
9	Adam	Lucas	Lucas	113-135
10	Adam	Lucas	Adam	135-123

Tabla 3. Resultados del combate

En la tabla proporcionada se muestra para cada partida, el jugador asignado a cada implementación, el ganador del juego y la puntuación final.

Como se puede comprobar el combate fue ajustado, puesto que, mi implementación consiguió ganar 6 partidas frente a las 4 ganadas por el agente de Adam. La diferencia de resultados en las partidas se debe a varios factores, como la aparición de power-ups o el movimiento pseudo aleatorio de los fantasmas en el modo pánico.

La principal diferencia entre las dos implementaciones que han participado en el combate, y que ha podido ayudar a mi agente a ganar un número mayor de partidas, es la forma en que se elige el objetivo. Mientras que la estrategia de Adam únicamente tiene en cuenta los puntos del laberinto, mi implementación evalúa las amenazas que suponen los fantasmas y además prioriza los power-ups cuando estos aparecen y están cerca.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

En este proyecto Fin de Máster realizado durante el curso académico 23/24, se ha conseguido desarrollar una infraestructura basada en el modelo cliente-servidor para gestión de partidas concurrentes en el juego versionado de Pac-Man. Además, se ha tenido un enfoque especial en la implementación para que los agentes sean controlados por algoritmos de inteligencia artificial.

El proyecto ha logrado cumplir con todos los objetivos establecidos y con las expectativas generadas. Obteniendo como resultado final una plataforma robusta y escalable que permite la participación de múltiples estudiantes en el desarrollo de algoritmos de IA en un entorno controlado.

En este proyecto se ha conseguido implementar con éxito el diseño del juego de Pac-Man, incluyendo las modificaciones necesarias para aumentar la competitividad y variabilidad de las estrategias. Se ha desarrollado y probado minuciosamente cada una de las diferentes fases del juego desarrolladas, como la construcción del mapa, el comportamiento de los fantasmas o los ‘power-ups’ desarrollados. Además, se ha creado un archivo de configuración flexible que permite ajustar con facilidad los parámetros del juego.

El mayor logro de este proyecto ha sido la exitosa implementación de la comunicación entre el cliente y servidor para la gestión de juegos multijugador en tiempo real. Esto se ha llevado a cabo a través de los sockets de Java utilizando un protocolo de comunicación personalizado basado en JSON. Además, ha sido fundamental la creación de serializadores y deserializadores personalizados para la transmisión de objetos complejos.

Durante el desarrollo de este proyecto han surgido distintos desafíos, como la sincronización de partidas en tiempo real. Sin embargo, todos estos desafíos han sido superados con éxito, implementando soluciones como el uso de Threads para la gestión de múltiples clientes y de partidas concurrentes. Además, se ha probado el sistema para comprobar su correcto

funcionamiento y asegurar que todos los jugadores tengan igualdad de condiciones, eliminando cualquier sesgo existente.

Los resultados obtenidos del combate final llevado a cabo entre mi implementación y la del coordinador de este proyecto, el doctor Adam Gaweda, han demostrado el correcto funcionamiento del sistema y la efectividad de los algoritmos desarrollados.

Aunque el proyecto ha sido realizado con éxito se plantean algunas líneas de trabajo futuras. Estas propuestas no solo buscan mejorar la funcionalidad actual sino también ampliar su alcance. A continuación, se describen los pasos futuros de este proyecto:

- Creación de nuevos laberintos. Sería muy beneficioso para este proyecto diseñar y desarrollar una mayor variedad de laberintos con diferentes estructuras y tamaños. Esto permitirá probar de forma más precisa la efectividad de los algoritmos en diferentes escenarios. Para esto, una gran ventaja es, que la implementación de los mapas de este proyecto ha sido llevada a cabo de tal forma que únicamente hay que seleccionar las mini celdas deseadas.
- Ajuste en el número y distribución de los ‘Pellet’. Reducir el número y distribución de los ‘pellets’ en el laberinto podría ser útil para ver cómo se comportan los algoritmos. Esto ayudaría a entender mejor la flexibilidad y adaptabilidad de las estrategias implementadas.
- Análisis de diversos algoritmos. Sería interesante analizar diferentes algoritmos actuales de IA, para evaluar su rendimiento en diferentes laberintos.
- Expansión del mapa. Explorar la posibilidad de aumentar el tamaño del mapa e incluso hacerlo dinámico. Un mapa más grande introduciría nuevos desafíos y más oportunidades a los jugadores de desarrollar estrategias más complejas.
- Aumentar el número de jugadores por partida. Si se realiza una expansión del mapa, sería muy interesante permitir la participación de un mayor número de jugadores por partida. Esta ampliación incrementaría en gran medida la complejidad del juego.
- Introducción de WildCards. Añadir elementos extra como cambios en la puntuación de los jugadores o la aparición de power-ups dinámicos podrían incrementar la

complejidad de los algoritmos y requerir que los jugadores adapten sus estrategias en tiempo real.

- Organizar una competición académica. Este paso se va a llevar a cabo durante el próximo curso académico 24/25. La realización de esta competición académica oficial añade un gran valor a este proyecto y valida el sistema desarrollado. Será muy interesante realizar un análisis de las estrategias empleadas por los alumnos.
- Nuevo modo de juego, Pac- Man Color Clash. Como último paso a futuro se pretende desarrollar un nuevo modo de juego en el que los ‘pellets’ en vez de desaparecer al ser comidos, cambian de color. Este nuevo modo de juego cambia por completo el desarrollo de estrategias con respecto a modo clásico.

Capítulo 8. BIBLIOGRAFÍA

- [1] «Iberdrola,» [En línea]. Available: <https://www.iberdrola.com/innovacion/historia-inteligencia-artificial>. [Último acceso: 2024].
- [2] «The Good Gamer,» [En línea]. Available: <https://thegoodgamer.es/el-papel-de-la-inteligencia-artificial-en-los-videojuegos/>.
- [3] «Santander SmartBank,» [En línea]. Available: <https://santandersmartbank.es/empleo/lenguajes-de-programacion/>.
- [4] Wikipedia, «Wikipedia,» [En línea]. Available: [https://en.wikipedia.org/w/index.php?title=Eclipse_\(software\)&oldid=1221248005..](https://en.wikipedia.org/w/index.php?title=Eclipse_(software)&oldid=1221248005..)
- [5] «Oracle,» [En línea]. Available: <https://www.oracle.com/es/database/what-is-json/>.
- [6] «Programación con Java,» [En línea]. Available: http://profesores.fi-b.unam.mx/carlos/java/java_basico9_2.html.
- [7] V. Rojas, «Curiosidades de la 'Inteligencia Artificial' que quizá no conocías,» Telefonía, 27 12 2021. [En línea]. Available: <https://www.telefonica.com/es/sala-comunicacion/blog/curiosidades-de-la-inteligencia-artificial-que-quiza-no-conocias/>.
- [8] «Test de Turing,» Wikipedia, 26 5 2024. [En línea]. Available: https://en.wikipedia.org/w/index.php?title=Turing_test&oldid=1225753131.

-
- [9] Daniel, «Inteligencia artificial : definición, historia, usos, peligros,» DataScientest, 10 8 2022. [En línea]. Available: <https://datascientest.com/es/inteligencia-artificial-definicion>.
- [10] «Inteligencia artificial, ventajas y desventajas,» VIU España, 05 03 2024. [En línea]. Available: <https://www.universidadviu.com/es/actualidad/nuestros-expertos/inteligencia-artificial-ventajas-y-desventajas>.
- [11] «¿Qué es la inteligencia artificial y cómo nos ayuda?,» Repsol, [En línea]. Available: <https://www.repsol.com/es/energia-futuro/tecnologia-innovacion/inteligencia-artificial/index.cshtml>.
- [12] B. Marr, «Estos son los 15 mayores riesgos de la IA,» Forbes España, 28 07 2023. [En línea]. Available: <http://forbes.es/tecnologia/316482/estos-son-los-15-mayores-riesgos-de-la-ia/>.
- [13] G. Gow, «Estos son los cinco principales peligros en la era de la IA y ChatGPT,» Forbes España, 10 04 2023. [En línea]. Available: <http://forbes.es/tecnologia/260702/estos-son-los-cinco-principales-peligros-en-la-era-de-la-ia-y-chatgpt/>.
- [14] «¿Qué es NPC?,» Pc Components, [En línea]. Available: <https://www.pccomponentes.com/npc-significado>.
- [15] «La Evolución de los Videojuegos: Del Pong a la Nube,» Cultura Tecnológica, [En línea]. Available: <https://culturatecnologica.com/videojuegos/evolucion-videojuegos-pong-nube/>.

- [16] «Space Invaders: Los “marcianitos” cumplen 40 años,» La Vanguardia, [En línea].
Available:
<https://www.lavanguardia.com/tecnologia/20180725/451089770519/space-invaders-40-aniversario-marcianitos-arcade.html>.
- [17] «LA INTELIGENCIA ARTIFICIAL EN LOS VIDEOJUEGOS,» Huawei Enterprise Support Community, [En línea]. Available:
<https://forum.huawei.com/enterprise/es/la-inteligencia-artificial-en-los-videojuegos/thread/667224519217135616-667212895009779712>.
- [18] «Pac-Man,» Pac-Man Wiki, [En línea]. Available:
[https://pacman.fandom.com/es/wiki/Pac-Man_\(videojuego\)](https://pacman.fandom.com/es/wiki/Pac-Man_(videojuego)).
- [19] «Donkey Kong Original Edition,» Nintendo, [En línea]. Available:
<https://www.nintendo.com/es-es/Juegos/Arcade/Donkey-Kong-Original-Edition-918598.html>.
- [20] «Reportajes,» MeriStation, [En línea]. Available:
https://as.com/meristation/2010/01/25/reportajes/1264403100_037611.html.
- [21] iat20, «Inteligencia artificial en videojuegos: origen y evolución,» IAT, 19 03 2020. [En línea]. Available: <https://iat.es/tecnologias/inteligencia-artificial/videojuegos/>.
- [22] «BattleCode,» BattleCode, [En línea]. Available: <https://battlecode.org/>.
- [23] I. Geffner, «A Guide to Battlecode,» BattleCode, [En línea]. Available: <https://battlecode.org/assets/files/battlecode-guide-xsquare.pdf>.

- [24] «Terms and conditions,» CodinGame, [En línea]. Available: <https://www.codingame.com/legal/sponsored-ai-contest-rules>.
- [25] «Legends of Code & Magic — a Codingame contest,» Medium, [En línea]. Available: <https://paulgrn.medium.com/legends-of-code-magic-a-codingame-contest-8b78dd30a859>.
- [26] «Objective,» RoboCup, [En línea]. Available: <https://www.robotcup.org/objective>.
- [27] «RoboCup-3D-Soccer-Field,» Wikipedia, [En línea]. Available: <https://en.wikipedia.org/wiki/File:RoboCup-3D-Soccer-Field.jpg>.
- [28] «Pac-Man,» Wikipedia, 01 06 2024. [En línea]. Available: <https://es.wikipedia.org/w/index.php?title=Pac-Man&oldid=160479460>.
- [29] J. Gomez, «Historia del Pac-Man: el imparable Comecocos,» Tokio School, 03 12 2021. [En línea]. Available: <https://www.tokioschool.com/noticias/historia-del-pac-man-el-imparable-comecocos/>.
- [30] C. Birch, «Understanding Pac-Man Ghost Behavior,» GameInternals, 2010 12 2020. [En línea]. Available: <http://gameinternals.com/understanding-pac-man-ghost-behavior>.
- [31] «Map 256 Glitch,» Pac-Man Wiki, [En línea]. Available: https://pacman.fandom.com/wiki/Map_256_Glitch.
- [32] «Kill Screen (Pac-Man),» Glitchpedia, [En línea]. Available: [https://glitchpedia.fandom.com/es/wiki/Kill_Screen_\(Pac-Man\)](https://glitchpedia.fandom.com/es/wiki/Kill_Screen_(Pac-Man)).

- [33] «Ms. Pac-Man,» Wikipedia, 22 01 2024. [En línea]. Available:
https://es.wikipedia.org/w/index.php?title=Ms._Pac-Man&oldid=157514882.
- [34] «Ms. Pac Man,» WikiNintendo, [En línea]. Available:
https://wikines.fandom.com/es/wiki/Ms._Pac_Man.
- [35] «PAC-MAN Championship Edition ya está disponibl,» PlayStation, 01 12 2010. [En línea]. Available: <https://blog.es.playstation.com/2010/12/01/pac-man-championship-edition-ya-est-disponible/>.
- [36] «Pac-Man 256,» MeriStation, [En línea]. Available:
https://as.com/meristation/2015/08/15/analisis/1439589600_147926.html.
- [37] «Pac-Man Battle Royale Deluxe,» Elite Home amerooms, [En línea]. Available:
<https://elitehomegamerooms.com/product/pac-man-battle-royale-deluxe/>.
- [38] «¿Cuál es la metodología más adecuada para tu proyecto?,» Deloitte Spain, [En línea]. Available:
<https://www2.deloitte.com/es/es/pages/technology/articles/waterfall-vs-agile.html>.
- [39] «Manifiesto por el Desarrollo Ágil de Software,» Agile Manifiesto, [En línea]. Available: <https://agilemanifesto.org/iso/es/manifesto>.
- [40] n. C. Benavides, «¿Transformación digital o transformación personal?,» Muy Segura, 12 09 2019. [En línea]. Available:

<https://www.muysegura.com/transformacion-digital-o-transformacion-personal/>.

[41] S. Ahmad, «Implementation Cycle of D365 F&O,» Microsoft, 11 02 2024. [En línea].

Available: <https://community.dynamics.com/blogs/post/?postid=c01785a6-d3c9-ee11-92bd-000d3adfe3bb>.

[42] «Calcular sueldo medio por profesión y provincia | InfoJobs Salarios,» InfoJobs, [En línea]. Available: <https://salarios.infojobs.net/>.

[43] «Conversor de salario por año / por mes / por semana / por hora,» Talent com, [En línea]. Available: <https://es.talent.com/convert?salary>.

[44] A. Ken, «Requisitos no funcionales: ¿Por qué son importantes?,» Gluo, 17 08 2023. [En línea]. Available: <https://www.gluo.mx/blog/requisitos-no-funcionales-por-que-son-importantes>.

[45] «UML,» Microsoft, [En línea]. Available: <https://www.microsoft.com/es-es/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling#:~:text=Tipos%20de%20diagramas%20de%20UML,usan%20diferentes%20tipos%20de%20personas..>

[46] «Tutorial de diagramas de casos de uso,» Creately, 26 06 2023. [En línea]. Available: <https://creately.com/blog/es/diagramas/tutorial-diagrama-caso-de-uso/>.

[47] «Java Socket Programming,» Java Point, [En línea]. Available: <https://www.javatpoint.com/socket-programming>.

- [48] «Java Socket Programming,» Javatpoint, [En línea]. Available: <https://www.javatpoint.com/socket-programming>.
- [49] «Simulated Annealing,» JijZept, [En línea]. Available: <https://www.documentation.jijzept.com/docs/jijzept/tutorials/sa/>.
- [50] Y. W. W. Ikezogwo, «Adversarial Search,» UW, [En línea]. Available: <https://courses.cs.washington.edu/courses/cse473/23au/notes/Adversarial-Expectimax-Search-notes.pdf>.
- [51] J. F. F. J. A. Abeynaya Gnanasekaran, «Reinforcement Learning in Pacman,» Stanford, [En línea]. Available: <https://cs229.stanford.edu/proj2017/final-reports/5241109.pdf>.
- [52] m. Maldonado, «ARQUITECTURA DE UN SISTEMA EXPERTO,» Wordpress, 15 07 2012. [En línea]. Available: <https://maricelamaldonado.wordpress.com/2011/03/23/arquitectura-de-un-sistema-experto/>.
- [53] Research Gate, [En línea]. Available: https://www.researchgate.net/figure/Figura-1-Funcionamiento-do-D-D-Lite-aO-robo-sente-o-ambiente-e-calcula-o-caminho_fig6_266888773.
- [54] «Vulnerable Ghost,» Pac-Man Fandom, [En línea]. Available: https://pacman.fandom.com/wiki/Vulnerable_Ghost.
- [55] «De los ODM a los ODS,» Sustainable Development Goals Fund, 21 01 2015. [En línea]. Available: <https://www.sdgfund.org/es/de-los-odm-los-ods>.

- [56] J. Cotey, «La paradoja de la educación para todos,» Onda Global, 18 09 2016. [En línea]. Available: <https://sites.middlebury.edu/ondaglobal/2016/09/18/la-paradoja-de-la-educacion-para-todos/>.
- [57] «Objetivos de Desarrollo del Milenio: se cumplen 3 metas antes de finalizar 2015 | UNICEF,» UNICEF, [En línea]. Available: <https://www.unicef.es/noticia/objetivos-de-desarrollo-del-milenio-se-cumplen-3-metas-antes-de-finalizar-2015#>.
- [58] «La Asamblea General adopta la Agenda 2030 para el Desarrollo Sostenible,» Naciones Unidas, 25 09 2015. [En línea]. Available: <https://www.un.org/sustainabledevelopment/es/2015/09/la-asamblea-general-adopta-la-agenda-2030-para-el-desarrollo-sostenible/>.
- [59] «Conoce 5 diferencias entre los ODM y los ODS,» Unicef, [En línea]. Available: <https://www.unicef.es/noticia/5-diferencias-entre-los-objetivos-de-desarrollo-del-milenio-y-los-objetivos-de-desarrollo>.
- [60] «Objetivos de Desarrollo Sostenible,» A&B, 18 10 2017. [En línea]. Available: <https://ab-innovativesolutions.com/es/actualidad-es/objetivos-de-desarrollo-sostenible/>.
- [61] «Los 17 ODS - La Ruta de la Sostenibilidad hacia 2030,» ODS, [En línea]. Available: <http://los17ods.org/>.

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

En septiembre del año 2000, se firma un documento histórico en el que líderes de 189 países distintos se ponen de acuerdo para fijar 8 objetivos a cumplir antes del año 2015. La reunión se llevó a cabo en la sede central de la ONU. [55]

A continuación, se muestran los ocho objetivos fijados:



Figura 67. Objetivos de Desarrollo del Milenio (ODM). Elaboración [56].

Aunque los ODM lograron grandes avances en diversas áreas, se comprobó que su alcance era limitado y que ciertos problemas fundamentales seguían estando presentes en la sociedad. Por ejemplo, aunque con los ODM se consiguió reducir de forma significativa la pobreza extrema, millones de personas en el mundo seguían en condiciones inhumanas. Otro ejemplo es con los indicadores de salud y educación, aunque estos mejoraron en gran medida, seguían existiendo desigualdades tanto entre diferentes países como dentro de ellos. [57]

Por ello, en 2015 surgen los Objetivos de Desarrollo Sostenible (ODS) como una expansión y mejora de los ODM. A diferencia de los ODM, los ODS cuentan con un enfoque más integral y ambicioso, abarcando no solo aspectos sociales y económicos, sino también ambientales [58]. Los ODS resaltan la necesidad de una colaboración global.

Los ODS, se componen de 17 objetivos interconectados, Figura 68, y 169 metas específicas que buscan promover un desarrollo inclusivo y sostenible a nivel mundial para el año 2030.

Cada objetivo está diseñado para enfrentar problemas específicos. Además, tal y como se recoge en el artículo de Unicef los 5 principales cambios en los ODS respecto a los ODM son los siguientes [59]:

- A diferencia de los ODM, los ODS ponen la sostenibilidad en el centro de la agenda.
- Los ODS, adoptan un enfoque más inclusivo, asegurando que se consideren las necesidades de todos.
- Mientras que los ODM se enfocaban en los países en desarrollo, los ODS tienen un alcance universal.
- A diferencia de los ODM, los ODS exigen un compromiso de todos los países del mundo.
- Los ODS expanden el número de objetivos de 8 a 17, con 169 metas específicas.



Figura 68. Objetivos de Desarrollo Sostenible. Obtenido de [60]

Este proyecto colabora a la consecución de estos objetivos en los siguientes apartados [61]:

- N.º 4: Educación de calidad. Meta 4.4: “Aumentar sustancialmente el número de jóvenes y adultos que tienen las competencias necesarias, en particular técnicas y profesionales, para acceder al empleo, el trabajo decente y el emprendimiento.”
 - Este proyecto proporciona una plataforma educativa que facilita el aprendizaje de competencias técnicas avanzadas. Al crear una infraestructura donde los participantes pueden implementar y probar algoritmos de Inteligencia Artificial, se está promoviendo habilidades que son altamente demandadas en el mercado laboral.
- N.º 9: Industria, Innovación e Infraestructura. Meta 9.5: “Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales de todos los países ...”
 - La infraestructura cliente-servidor desarrollada no solo es una herramienta educativa, sino que también es una plataforma para la innovación e investigación tecnológica. De este modo, el proyecto promueve la creación de nuevas soluciones tecnológicas a través de la optimización de los algoritmos, contribuyendo al fortalecimiento de los sectores industriales.
- N.º 10: Reducción de las desigualdades. Meta 10.2: “Potenciar y promover la inclusión social, económica y política de todas las personas, independientemente de su edad, sexo, discapacidad, raza, etnia, origen, religión o situación económica u otra condición.”
 - Una de las principales características de este proyecto es su accesibilidad. Esta plataforma ha sido diseñada para ser utilizada para todos los públicos, independientemente de su nivel de experiencia. Con esto se busca proporcionar una herramienta educativa accesible que genere igualdad de oportunidades.
- N.º 17: Alianzas para lograr los objetivos. Meta 17.6: “Mejorar la cooperación regional e internacional Norte-Sur, Sur-Sur y triangular en materia de ciencia, tecnología e innovación y el acceso a ellas y aumentar el intercambio de conocimientos en condiciones mutuamente convenidas...”

- Este proyecto fomenta la colaboración entre estudiantes, desarrolladores y académicos de diferentes partes del mundo. Al proporcionar una plataforma común para la competición se está creando oportunidades para el intercambio de conocimientos, fortaleciendo alianzas globales en la ciencia.

ANEXO II: IMPLEMENTACIÓN PROPIA DE IA

En este Anexo se adjunta el código java con la implementación propia desarrollada para el juego del Pac-Man.

```
public Move getMove(String player) {
    return findBestMove();
}
public Move findBestMove() {

    Move move = Move.DO_NOTHING;

    // Returns the final decision of the AI implementation

    int[][] originalMap = GameController.getIntMap();

    int remainingTime = GameController.getRemainingTime();

    Position current = new Position((int) x, (int) y);

    Pacman opponent = GameController.getOpponent(this.getId());
    Position opponentPos = new Position((int) opponent.x, (int) opponent.y);

    Position target = closestPoint(current, 1, originalMap, opponentPos);

    Map<String, Ghost> ghosts = GameController.getGhosts();

    double ghostDistance = Double.POSITIVE_INFINITY;
    Position ghostTargetPos = new Position(0, 0);

    for(Map.Entry<String, Ghost> entry : ghosts.entrySet()) {

        String key = entry.getKey();
        Ghost ghost = entry.getValue();

        int x = (int) ghost.getX();
        int y = (int) ghost.getY();

        Position ghostPos = new Position(x, y);

        String ghostMode = GameController.getGhostMode(key);

        if(!ghostMode.equals("scatter") && !ghostMode.equals("chase") &&
!ghostMode.equals("eaten")) {
            double distance = calculateEuclideanDistance(current, ghostPos);
            if(distance < ghostDistance) {
                ghostDistance = distance;
                ghostTargetPos = ghostPos;
            }
        }
    }

    return move;
}
```

```

    }

    }

    //System.out.println("Current: " + current.toString());
    //System.out.println("Target: " + target.toString());

    if(ghostTargetPos.getRow()==0 || ghostDistance>4) {
        if(remainingTime<45 && remainingTime>=40) {
            target = new
Position(GameSettings.PW1_COL,GameSettings.PW1_ROW);
        } else if(remainingTime<=40 && GameController.pw1Status){
            target = new
Position(GameSettings.PW1_COL,GameSettings.PW1_ROW);
        } else if(remainingTime<25 && remainingTime>=20) {
            target = new
Position(GameSettings.PW2_COL,GameSettings.PW2_ROW);
        } else if(remainingTime<=20 && GameController.pw2Status){
            target = new
Position(GameSettings.PW2_COL,GameSettings.PW2_ROW);
        } else {
            Position powerPellet =
closestBigPoint(current,2,originalMap);
            if(powerPellet.getCol()!=0) {
                double distance =
calculateEuclideanDistance(current,powerPellet);
                if(distance<5) {
                    target = powerPellet;
                }
            }
        }
    }else {
        target = ghostTargetPos;
    }

    move = aStarSearch(current,target,originalMap);

    switch(move) {
        case RIGHT:
            move = checkGhostMove(move,new
Position(current.getRow()+1,current.getCol()));
        case LEFT:
            move = checkGhostMove(move,new Position(current.getRow()-
1,current.getCol()));
        case UP:
            move = checkGhostMove(move,new
Position(current.getRow(),current.getCol()-1));
        case DOWN:
            move = checkGhostMove(move,new
Position(current.getRow(),current.getCol()+1));
        case DO_NOTHING:
    }

```

```
        move = checkValidMove(move);
        return move;
    }

    public Move checkGhostMove(Move move, Position pacman) {

        Move checkMove = move;
        Map<String,Ghost> ghosts = GameController.getGhosts();

        boolean error = false;

        for(Map.Entry<String, Ghost> entry : ghosts.entrySet()) {

            String key = entry.getKey();
            Ghost ghost = entry.getValue();

            String ghostMode = GameController.getGhostMode(key);

            if(ghostMode.equals("scatter") || ghostMode.equals("chase")) {

                int x = (int) ghost.getX();
                int y = (int) ghost.getY();

                int x_dif = pacman.getRow()-x;
                int y_dif = pacman.getCol()-y;

                if(x_dif==0 && y_dif==0) {
                    error = true;
                }

                x_dif = pacman.getRow()-x-1;
                y_dif = pacman.getCol()-y;

                if(x_dif==0 && y_dif==0) {
                    error = true;
                }

                x_dif = pacman.getRow()-x+1;
                y_dif = pacman.getCol()-y;

                if(x_dif==0 && y_dif==0) {
                    error = true;
                }

                x_dif = pacman.getRow()-x;
                y_dif = pacman.getCol()-y-1;

                if(x_dif==0 && y_dif==0) {
                    error = true;
                }

                x_dif = pacman.getRow()-x;
```

```
        y_dif = pacman.getCol()-y+1;

        if(x_dif==0 && y_dif==0) {
            error = true;
        }
    }

    if(error) {
        System.out.println("If moving there Pacman will be caught by a
ghost");
        ArrayList<Move> possibleMoves = Board.getPossibleMoves(this);

        if(possibleMoves.contains(move)) {
            possibleMoves.remove(move);
            Random random = new Random();
            checkMove =
possibleMoves.get(random.nextInt(possibleMoves.size()));
        }
    }

    return checkMove;
}

public Move checkValidMove(Move move) {

    Move checkMove = move;

    ArrayList<Move> possibleMoves = Board.getPossibleMoves(this);

    if (!possibleMoves.contains(move)) {
        System.out.println("Move Selected is not possible");
        Random random = new Random();
        checkMove =
possibleMoves.get(random.nextInt(possibleMoves.size()));
    }

    return checkMove;
}

public Position closestBigPoint(Position pacman, int points, int[][] map) {

    int rows = map.length;
    int cols = map[0].length;

    Position pos = new Position(0,0);
    double distance = Double.POSITIVE_INFINITY;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if(map[i][j]>=points) {
```

```
        Position local = new Position(j,i);
        double dis = calculateEuclideanDistance(local,pacman);
        if(dis<distance) {
            distance = dis;
            pos = local;
        }
    }
}

return pos;
}

public Position closestPoint(Position pacman, int points, int[][] map, Position
opponentPos) {

    int rows = map.length;
    int cols = map[0].length;

    Position posOpponent = new Position(0,0);
    Position pos = new Position(0,0);
    double distanceOpponent = Double.POSITIVE_INFINITY;
    double distance = Double.POSITIVE_INFINITY;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if(map[i][j]>=points) {
                Position local = new Position(j,i);
                double dis = calculateEuclideanDistance(local,opponentPos);
                if(dis<distanceOpponent) {
                    distanceOpponent = dis;
                    posOpponent = local;
                }
            }
        }
    }

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if(map[i][j]>=points && !posOpponent.equals(new Position(j,i))) {
                Position local = new Position(j,i);
                double dis = calculateEuclideanDistance(local,pacman);
                if(dis<distance) {
                    distance = dis;
                    pos = local;
                }
            }
        }
    }

    return pos;
}
```

```
// This list represents all the nodes that has been removed from the priority
queue and added as a potential path
public List<Node> potentialPathNodes = new ArrayList<Node>();
// This list of positions stores the path from the robot position to the target
public List<Position> path = new ArrayList<Position>();

public Move aStarSearch( Position initialPos, Position targetPos, int[][] gameMap
) {

    // For the first node, g value is 0 since the robot hasn't moved yet
    int g = 0;
    // I calculate the Manhattan distance from the robot position to the
target (walls ignored)
    int h = heuristic(initialPos,targetPos);
    // f is calculated as g+h
    int f = g+h;

    // Initialization of the priority queue
    PriorityQueue<Node> queue = new PriorityQueue<Node>();

    // First Node is added to the queue
    queue.offer(new Node(initialPos,null,f,g,h));

    boolean algorithmFinished = true;

    long startTime = System.currentTimeMillis();

    // While queue is not empty the process is repeated
    while(!queue.isEmpty()) {

        //System.out.println("Current time: "+System.currentTimeMillis());

        if (System.currentTimeMillis() - startTime > 60) {
            System.out.println("Forced End of the A* Algorithm");
            algorithmFinished = false;
            break; // Salir si excede el límite de tiempo
        }

        // I get the first node in the queue that it's the one with lowest f
value!
        Node node = queue.poll();
        // Since it has been removed from the queue it's added as a
potential path node!
        potentialPathNodes.add(node);
        // I get the child that it's now the new parent
        Position posNewParent = node.getChild();

        // If the new position of the parent is the target, the loop is
broken.
        if(posNewParent.equals(targetPos)==true) {
            break;
        }
    }
}
```

```
Position neighbor;
neighbor = new Position(posNewParent.getRow(),posNewParent.getCol()-
1);

// If it is not null
if(neighbor!=null) {
    // Getting the tile status
    int status = getIntItem(neighbor.getRow(),
neighbor.getCol(),gameMap);
    // Here I check that the tile is not a wall and, that the
neighbor is not the parent of the tile evaluated
    if(checkNodesList(posNewParent,neighbor)==false && status!=-
1) {
        // Here g,h and f values are calculated and the node is
added to the queue.

        g = node.getG()+1;
        h = heuristic(neighbor,targetPos);
        f = g+h;
        queue.offer(new Node(neighbor,posNewParent,f,g,h));
    }
}

neighbor = new
Position(posNewParent.getRow(),posNewParent.getCol()+1);
// If it is not null
if(neighbor!=null) {
    // Getting the tile status
    int status = getIntItem(neighbor.getRow(),
neighbor.getCol(),gameMap);
    // Here I check that the tile is not a wall and, that the
neighbor is not the parent of the tile evaluated
    if(checkNodesList(posNewParent,neighbor)==false && status!=-
1) {
        // Here g,h and f values are calculated and the node is
added to the queue.

        g = node.getG()+1;
        h = heuristic(neighbor,targetPos);
        f = g+h;
        queue.offer(new Node(neighbor,posNewParent,f,g,h));
    }
}

neighbor = new
Position(posNewParent.getRow()+1,posNewParent.getCol());
// If it is not null
if(neighbor!=null) {
    // Getting the tile status
    int status = getIntItem(neighbor.getRow(),
neighbor.getCol(),gameMap);
    // Here I check that the tile is not a wall and, that the
neighbor is not the parent of the tile evaluated
    if(checkNodesList(posNewParent,neighbor)==false && status!=-
1) {
```

```

// Here g,h and f values are calculated and the node is
added to the queue.
        g = node.getG()+1;
        h = heuristic(neighbor,targetPos);
        f = g+h;
        queue.offer(new Node(neighbor,posNewParent,f,g,h));
    }
}

neighbor = new Position(posNewParent.getRow()-
1,posNewParent.getCol());
// If it is not null
if(neighbor!=null) {
    // Getting the tile status
    int status = getIntItem(neighbor.getRow(),
neighbor.getCol(),gameMap);
    // Here I check that the tile is not a wall and, that the
neighbor is not the parent of the tile evaluated
    if(checkNodesList(posNewParent,neighbor)==false && status!=-
1) {
        // Here g,h and f values are calculated and the node is
added to the queue.
        g = node.getG()+1;
        h = heuristic(neighbor,targetPos);
        f = g+h;
        queue.offer(new Node(neighbor,posNewParent,f,g,h));
    }
}

if(algorithmFinished) {

    // Once the queue is empty or the target is reached, the best path
is calculated and stored in a list (the positions)

    boolean completed = false;

    Position pathPosition=targetPos;

    path.add(pathPosition);

    while(completed==false) {

        if(pathPosition.equals(initialPos)) {
            break;
        }
        for(Node node : potentialPathNodes) {
            if(node.getChild().equals(pathPosition)) {
                pathPosition = node.getParent();
                path.add(pathPosition);
                break;
            }
        }
    }
}

```

```
    }

    path.remove(path.size() - 1);

    return getRouteStep(initialPos);

} else {
    ArrayList<Move> possibleMoves = Board.getPossibleMoves(this);
    return possibleMoves.get(random.nextInt(possibleMoves.size()));
}
}

public int getIntItem(int x, int y, int[][] gameMap) {

    try {
        return gameMap[y][x];
    } catch (ArrayIndexOutOfBoundsException ex) {
        return -1;
    }
}

public Move getRouteStep(Position pacmanPosition) {

    // I get the last tile of the path since is the next position the robot
    should move
    int lastTile = path.size() - 1;

    if(lastTile<0) {
        return Move.DO_NOTHING;
    } else {

        Position pos = path.get(lastTile);

        // What action should the robot do based on the positions
        if(pos.getRow()==pacmanPosition.getRow() &&
pos.getCol()>pacmanPosition.getCol()) {return Move.DOWN;}
        else if(pos.getRow()==pacmanPosition.getRow() &&
pos.getCol()<pacmanPosition.getCol()) {return Move.UP;}
        else if(pos.getRow()<pacmanPosition.getRow() &&
pos.getCol()==pacmanPosition.getCol()) {return Move.LEFT;}
        else if(pos.getRow()>pacmanPosition.getRow() &&
pos.getCol()==pacmanPosition.getCol()) {return Move.RIGHT;}
        else if(pos.getRow()==pacmanPosition.getRow() &&
pos.getCol()==pacmanPosition.getCol()) {
            // This means the robot has found the target!.
            return Move.DO_NOTHING;
        } else {
            // Shouldn't happen
            return Move.DO_NOTHING;
        }
    }
}
```

```
}

// This function check if a Node(parent,child,...) has been stored before as a
Node(child,parent,...)
public boolean checkNodesList(Position parent, Position child) {

    boolean exist = false;

    for(Node node : potentialPathNodes) {
        if(parent.equals(node.getChild()) && child.equals(node.getParent()))
        {
            exist = true;
        }
        if(parent.equals(node.getParent()) && child.equals(node.getChild()))
        {
            exist = true;
        }
    }

    return exist;
}

// Manhattan Distance (Ignoring walls)
public int heuristic(Position current, Position target) {
    return Math.abs(current.getRow() - target.getRow()) +
    Math.abs(current.getCol() - target.getCol());
}

public static double calculateEuclideanDistance(Position current, Position
target) {
    return Math.sqrt(Math.pow(current.getRow() - target.getRow(), 2) +
    Math.pow(current.getCol() - target.getCol(), 2));
}

public class Node implements Comparable<Node>{

    public Position parent;
    public Position child;
    public int f, g, h;

    Node()
    {
        parent = new Position(-1, -1);
        f = -1;
        g = -1;
        h = -1;
    }

    public Node(Position child, Position parent, int f, int g, int h) {
        this.child = child;
        this.parent = parent;
    }
}
```

```
        this.f = f;
        this.g = g;
        this.h = h;
    }

    // Getter functions
    public Position getChild() { return child;}
    public Position getParent() { return parent; }
    public int getF() { return f; }
    public int getG() {return g;}
    public int getH() { return h; }

    // I make them comparable by their value of f
    @Override
    public int compareTo(Node node) {
        return Integer.compare(this.f, node.f);
    }
}
```

ANEXO III: GUÍA DE USUARIO

En este Anexo se adjunta la guía de usuario desarrollada para los alumnos de la asignatura “CSC 411-Aartificial Intellegence”.

En esta guía se detallan los objetivos y reglas del juego, instrucciones de como programar tu propio agente junto con otros requisitos del sistema, necesarios para poder jugar con la mejor experiencia y visualización posible. El nombre de la competición acordada con el doctor Adam Gaweda es PacMania.

Además, se puede encontrar la guía de usuario en formato .pdf para su descarga y visualización en el siguiente enlace:

https://drive.google.com/file/d/11dmwSY8AQg6trOJ_Ehae5gS4i697CXeF/view?usp=sharing

A continuación, se adjunta la guía de usuario.

PAC-MANIA

USER GUIDE

Objective

The primary objective of this project is to **design a sophisticated AI agent** in Java that can outperform its opponent in a competitive multiplayer Pac-Man game. Each match lasts **60 seconds**, during which the AI agents must employ effective strategies to maximize their scores. The successful AI will not only need to navigate the game environment skillfully but also make strategic decisions at crucial moments. **The ultimate goal is for the AI to consistently score higher than its opponent** within the limited timeframe of each match.

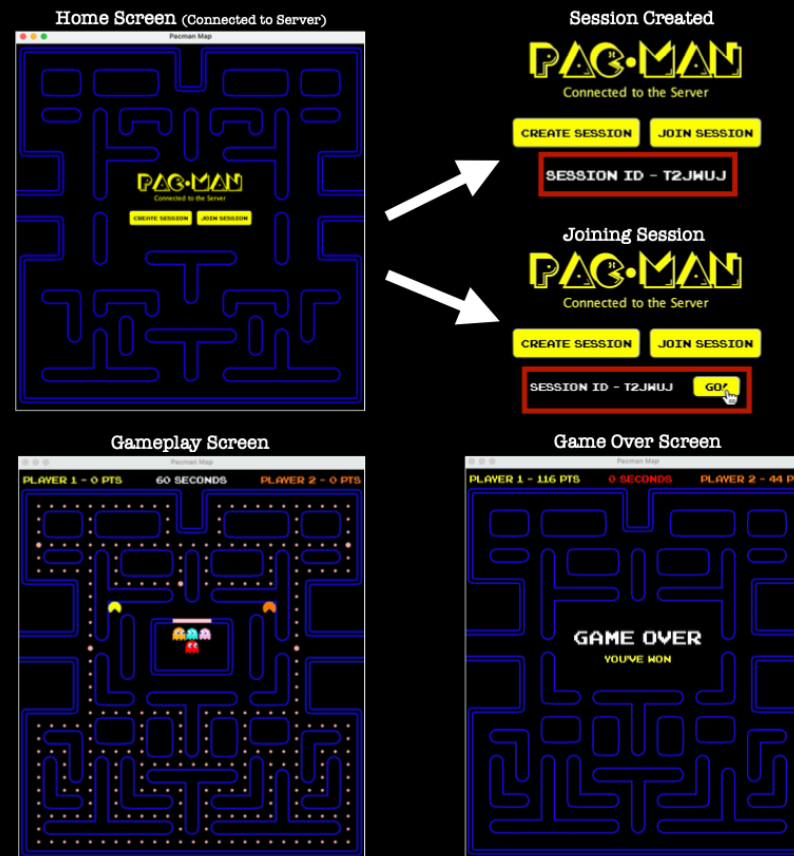
System Requirements

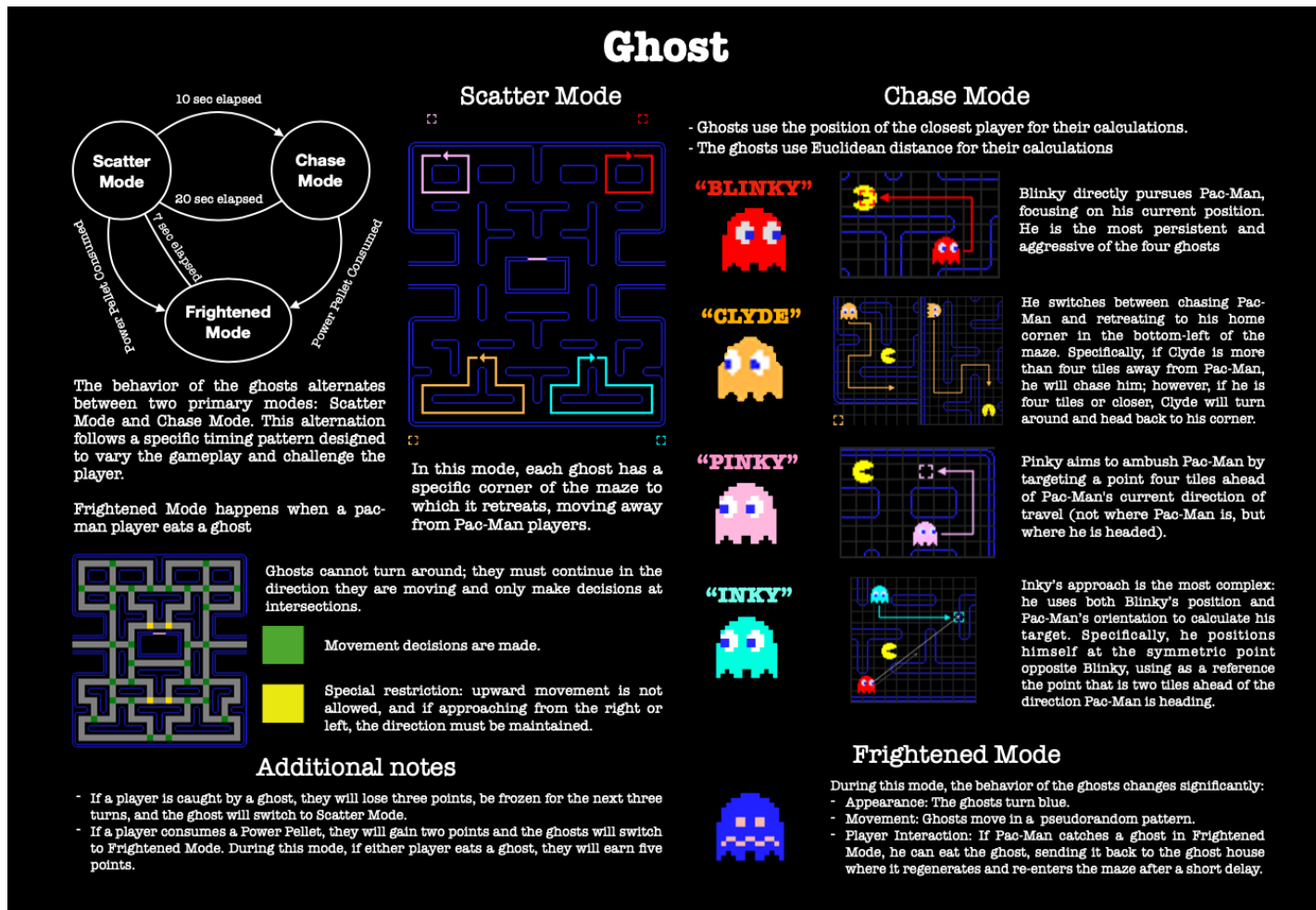
- JDK version 9 or higher.
- Eclipse IDE
- Min Screen Resolution: 2560 x 1600 pixels

Setup

- Clone the repository using the command:
`git clone https://github.com/luca1015/BattleCodeClient`
- Modify the `Pacman.java` file located in the `edu.ncsu.pacman.agent.player` package, to implement your AI strategy.
- Run `StartPacmanFrame.java` located in the `edu.ncsu.pacman.ui` package, to play against others

Game Screens





- Pellet: 1pt
- Power pellet: 2 pts
- 👹 Frightened ghost: 5 pts
- 👹 Chase/Scatter: -3 pts
- 🔧 PowerUps: 0-15 pts

PowerUps

Each game randomly features two out of five possible power-ups.

- **Appearance Schedule:** The first power-up appears with 40 seconds remaining in the game, and the second appears at the 20-second mark.
- **Location:** Both power-ups appear at specific positions adjacent to the ghost house—one on each side.

- | | |
|---|--|
|  | Freezes the opponent for 5 turns, preventing them from moving. |
|  | Swaps your position with your opponent's position on the maze. |
|  | Grants the player who collects it a bonus of 10 points. |
|  | Transfers 5 points from your opponent's score to yours. |
|  | Regenerates all dots on the maze that have been eaten. |

Coding your agent

In this section, you will learn how to code the decision-making process for your Pac-Man agent. You have **80 milliseconds** to determine and return your next move in each game tick, making efficient coding crucial.

Modifying the getMove() Method: The getMove() method in Pacman.java is the core function where you will implement your agent's strategy. You need to modify this method to analyze the game state and return the best possible move from the following options: RIGHT, UP, DOWN, LEFT. DO NOTHING.

Available Functions and Data: You have access to several functions and data structures that provide information about the game state:

[illegible]

- (1) Power-Ups Information:**

- **GameController.getPowerUps()**: returns a list of strings, each representing a power-up available on the map.
- **GameController.getPwRow(String powerUp)**: provides the row in the grid where a specific power-up is located.
- **GameController.getPwCol(String powerUp)**: provides the column in the grid where a specific power-up is located.

(2) Ghost Information:

- **GameController.getGhosts()**: gives a map with ghost details, where each key is a ghost's color and the value is the corresponding Ghost object.

— **(3) Map Data:**

- **GameController.getIntMap()**: retrieves the game map as an integer matrix. Values in this matrix represent different elements: -1 for walls, 0 for passable paths without rewards, 1 for single points, 2 for power pellets, and 3 for power-ups.

(4) Opponent Information:

- **GameController.getOpponent(this.getId())**: fetches the Pac-Man instance of your opponent, providing access to their current score and position for tactical planning.

(5) Remaining Time:

- **GameController.getRemainingTime()**: indicates the time left in the current game, which is crucial for time-sensitive strategies.

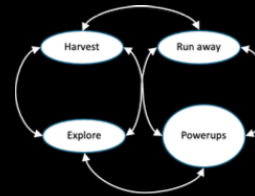
Important Considerations: In the game map, x refers to the column index, and y refers to the row index. To access an element in the map, use `map[row][col] = map[y][x]`.

Recommend Strategies

A* Algorithm

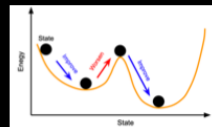


- Using Different Heuristics: Experiment with different heuristics to find the optimal path to goals (points or exits).
- Cost Function ($g(n)$): Define $g(n)$ as the sum of the costs of moving through each square to reach the position n . Adjust this cost by the points or power-ups collected along the way. For instance, each move could have a base cost of 1, and collecting a point or power-up could subtract a specific value from this cost, making the path more attractive.



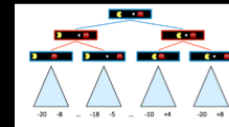
Finite State Machine (FSM)

- Harvest: Collect dots; switch to Run Away if ghosts approach, or to Power-Ups for advantages.
- Run Away: Avoid ghosts; revert to Harvest when safe, or switch to Explore or Power-Ups as needed.
- Explore: Seek new dots; transition to Harvest with dots, Run Away with ghosts, or Power-Ups if found.
- Power-Ups: Grab and use power-ups; post-use, move to Harvest for benefits, Run Away from risks, or Explore further.



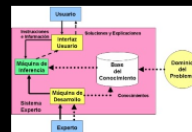
Simulated Annealing

Use this algorithm to develop strategies like deciding when it's optimal to go for points versus power-ups, or how to switch strategies under certain game conditions.



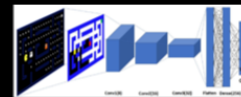
Minimax Algorithm

Seek to maximize the points collected while minimizing the chances of being captured by ghosts. This involves choosing paths that maximize point and power-up collection while effectively avoiding ghosts controlled by AI or another player.



Expert System

Build a knowledge base containing rules for specific game situations, such as "if a ghost is less than 3 squares away and no power-ups are nearby, then evade" or "if a power-up is available and no ghosts are near, then pursue". The system makes logical decisions based on these predefined rules.



Deep Q-Learning

Apply Deep Q-Learning by training a Deep Q-Network (DQN) to make decisions based on the current state of the game, such as the positions of Pac-Man, the ghosts, and available points/power-ups. The network learns to predict which action (moving in a specific direction) will maximize future rewards based on accumulated experience during training.

ANEXO IV: MANUAL DE INSTALACIÓN

En este Anexo, se va a desarrollar un manual de instalación del juego desarrollado, con el objetivo que todo el mundo que lo desee pueda probar y desafiar a otros jugadores.

Por ello, en primer lugar, es necesario descargarse el proyecto. Esto puede realizarse de dos formas distintas:

1. Descargar el proyecto entero de GitHub
2. Clonar el proyecto

El enlace de descarga es el siguiente: <https://github.com/luca1015/BattleCodeClient>

Por otro lado, se va a utilizar Eclipse, como IDE de ejemplo para explicar cómo clonar el proyecto.

Lo primero se debe tener una cuenta de GitHub, en caso de no disponer de una, puede registrarse en el siguiente enlace: <https://github.com>

Posteriormente, debe instalar Git, en caso de no tenerlo instalado, a través del siguiente enlace: <https://www.git-scm.com/downloads>.

Al instalar Git se debe marcar la opción que permita utilizarlo con aplicaciones de terceros, tal y como se indica en la siguiente figura.

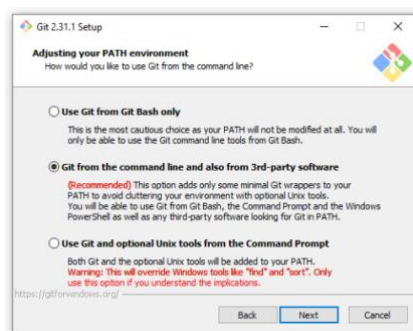


Figura 69. Instalación de Git. Elaboración propia.

Después dispondrá de dos opciones de realizar el clonado.

1. A través de línea de comandos con el siguiente comando:

`git clone https://github.com/luca1015/BattleCodeClient <ubicación>`

2. Con Eclipse. File → Import → Git → Project from Git (smart import) → Next

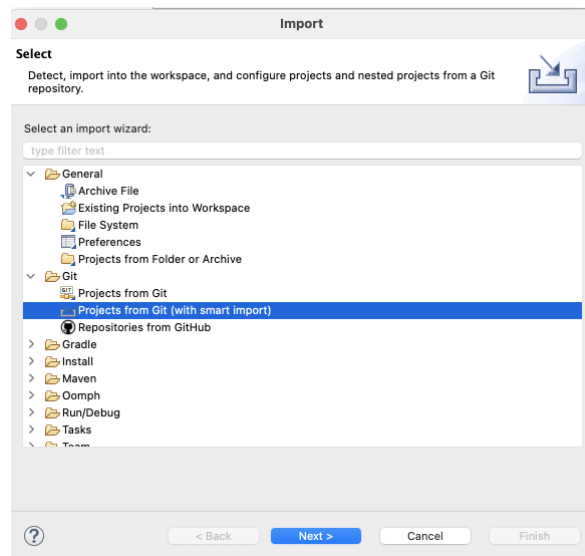


Figura 70. Importar proyecto Git. Elaboración propia.

En la siguiente ventana se debe seleccionar ‘Clone URI’ → Next

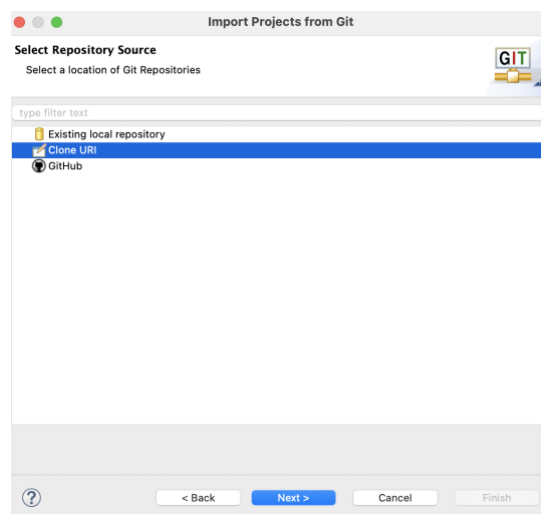


Figura 71. Clone URI. Elaboración propia.

Posteriormente se debe pegar este enlace en el campo de URI. Con esto se deberían de rellena automáticamente los campos ‘Host’ y ‘Repository path’. Por otro lado, al ser este un repositorio público se puede dejar sin rellenar los campos de autenticación.

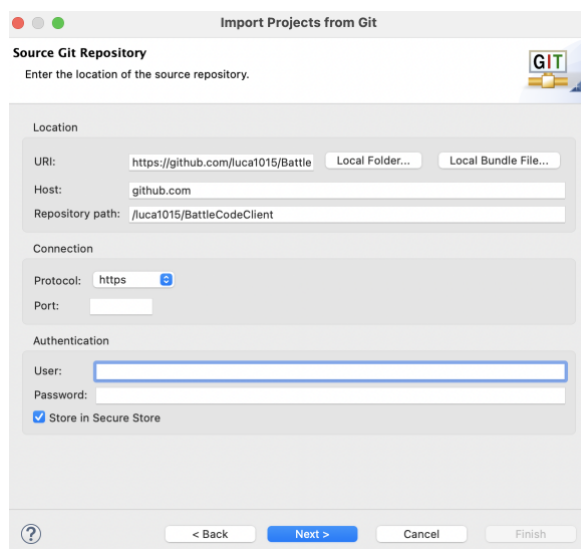


Figura 72. Campos clonación en Git. Elaboración propia.

En la siguiente pantalla la rama que se debe seleccionar para clonar es la principal, ‘master’.

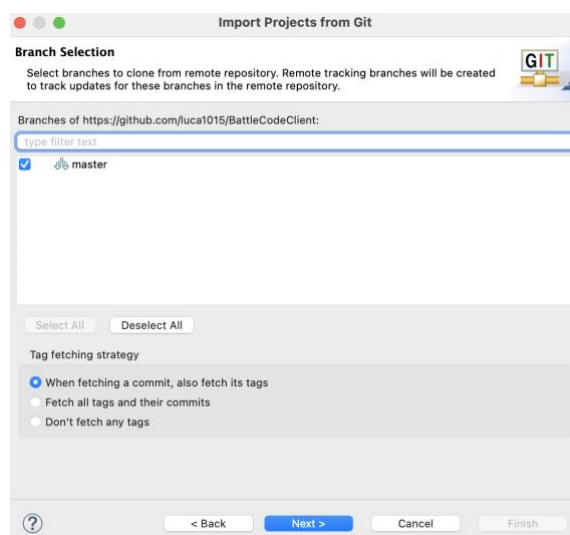


Figura 73. Branch Selection. Elaboración propia.

En la siguiente ventana se debe especificar la ruta y carpeta en la que se desea almacenar el proyecto.

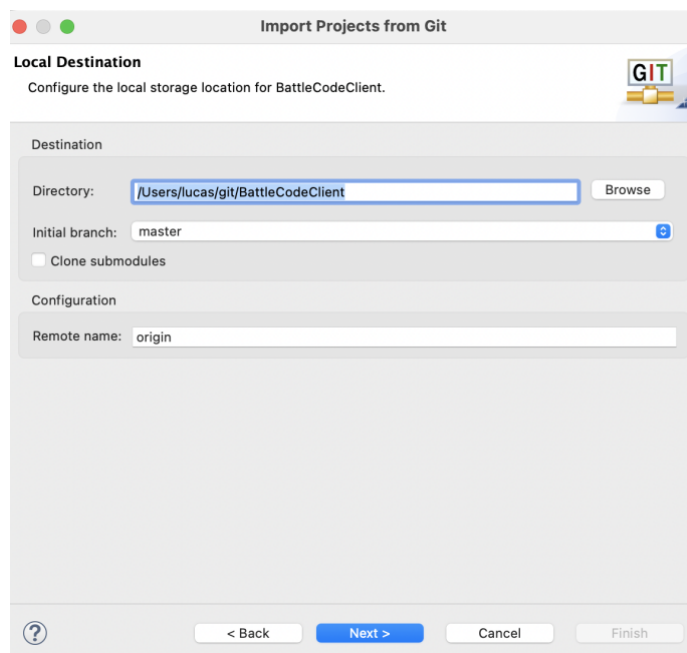


Figura 74. Ruta en la que se almacena el proyecto. Elaboración propia.

Por último, en caso de fallar la compilación del proyecto, se debe comprobar que la versión Java es igual o superior a la requerida, en este caso Java 11. Esto se puede verificar haciendo click derecho en el proyecto y pulsando la opción de propiedades. En esta ventana debe seleccionar Java Build Path e irse a la opción de Libraries.

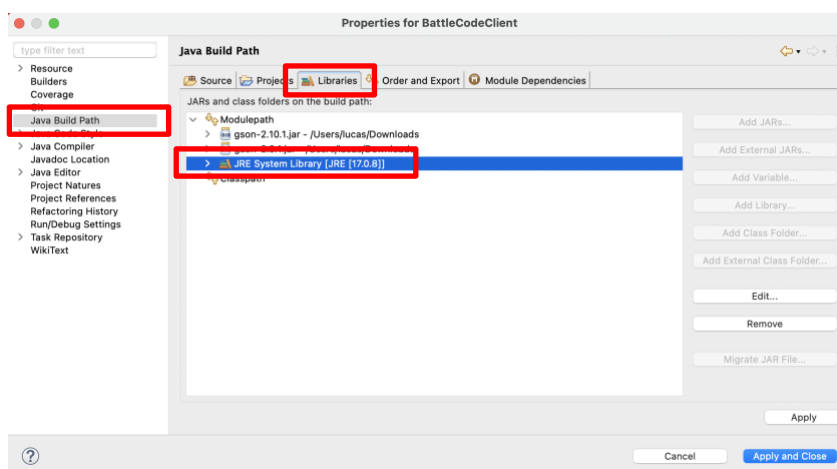


Figura 75. Verificar la versión java del proyecto. Elaboración propia.

En caso de no contar con la versión Java requerida puede descargar la versión 11 en el siguiente enlace: <https://www.oracle.com/es/java/technologies/javase/jdk11-archive-downloads.html>