

UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Inteligencia artificial para jugar al Juego de las Amazonas

AUTOR: Gonzalo Rodríguez Rodríguez

DIRECTOR: Cristian González García

Agradecimientos

No hubiese llegado hasta aquí sin la ayuda de mucha gente, mucha más de la que puedo mencionar aquí. De todas formas, me gustaría agradecer:

En primer lugar, a mi tutor Cristian por su ayuda y atención.

A mi familia, por apoyarme y permitirme estudiar este grado.

A todos mis amigos, los que he hecho durante la carrera y los que me han acompañado toda mi vida. En especial a mi mejor amigo Abel por empujarme siempre a ser una mejor persona.

Por último, a toda la gente que no he mencionado aquí pero que, para lo bueno y para lo malo, me han hecho ser quien soy.

Resumen

La inteligencia artificial es uno de los campos de la informática que más ha crecido y que más interés ha generado en los últimos años. Uno de los motivos más importantes por los que la inteligencia artificial interesa al público es que emula el pensamiento y razonamiento humano. La inteligencia artificial llega a superar al humano en campos concretos. Uno de los campos en los que la inteligencia artificial está probando su eficacia es el de los juegos de mesa.

El entorno de los juegos de mesa es perfecto para comparar el desempeño de una inteligencia artificial frente al de un humano. Hace pocos años parecía imposible que un ordenador superase a un humano experimentado en un juego de mesa. Sin embargo, en 1997, la supercomputadora Deep Blue superó en un encuentro a seis partidas al campeón del mundo de ajedrez Garry Kasparov. En 2016, en un evento similar, AlphaGo ganó al jugador profesional de go Lee Sedol 4 a 1.

El objetivo de este trabajo es desarrollar una serie de inteligencias artificiales para jugar al Juego de las Amazonas, un juego de mesa que comparte características del ajedrez y del go. Además, se entrenarán las inteligencias artificiales y se comparará su rendimiento en partidas entre ellas. Por último, se presentará una interfaz gráfica sencilla para permitir a un jugador humano jugar contra las inteligencias artificiales desarrolladas.

Las inteligencias artificiales que se desarrollarán serán: un algoritmo voraz, un algoritmo Minimax y un árbol de búsqueda de Monte Carlo.

Palabras Clave

Juego de las amazonas, inteligencia artificial, búsqueda en árboles, algoritmo voraz, MCTS, Minimax.

Abstract

Artificial intelligence is one of the fields of computer science that has grown the most and generated the most interest in recent years. One of the most important reasons why the public is so interested in artificial intelligence is that it emulates human thought and reasoning. Artificial intelligence is even capable of surpassing the human in specific fields. One of the fields where artificial intelligence is proving its efficacy is in board games.

The board game environment is perfect to compare the performance of an artificial intelligence against that of a human. A few years ago, it seemed impossible for a computer to surpass an experienced human in a board game. However, in 1997, the supercomputer Deep Blue defeated the world chess champion Garry Kasparov in a six-game match. In 2016, in a similar event, AlphaGo beat the professional Go player Lee Sedol 4 to 1.

The objective of this project is to develop a series of artificial intelligences to play the Game of Amazons, a board game that shares characteristics with chess and Go. Furthermore, the artificial intelligences will be trained and their performance will be compared in matches against each other. Finally, a simple graphical interface will be presented to allow a human player to play against the developed artificial intelligences.

The artificial intelligences that will be developed are a greedy algorithm, a Minimax algorithm and a Monte Carlo tree search algorithm.

Keywords

Game of the Amazons, artificial intelligence, tree search, greedy algorithm, MCTS, Minimax.

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO.....	16
1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO	16
1.2 RESUMEN DE TODOS LOS ASPECTOS.....	17
CAPÍTULO 2. INTRODUCCIÓN.....	18
2.1 JUSTIFICACIÓN DEL PROYECTO	18
2.2 OBJETIVOS DEL PROYECTO	18
2.3 ESTUDIO DE LA SITUACIÓN ACTUAL	18
2.3.1 <i>Evaluación de Alternativas</i>	19
CAPÍTULO 3. ASPECTOS TEÓRICOS.....	26
3.1 JUEGO DE LAS AMAZONAS.....	26
3.2 ALGORITMO VORAZ.....	27
3.3 MINIMAX.....	28
3.4 MONTE CARLO TREE SEARCH.....	29
3.5 ALPHA ZERO	30
CAPÍTULO 4. PLANIFICACIÓN DEL PROYECTO Y PRESUPUESTOS INICIALES.....	32
4.1 PLANIFICACIÓN INICIAL.....	32
4.1.1 <i>Planificación temporal</i>	32
4.2 PRESUPUESTO INICIAL	35
4.2.1 <i>Definición de empresa</i>	35
4.2.2 <i>Desarrollo de Presupuesto Detallado (Empresa)</i>	37
4.2.3 <i>Desarrollo de Presupuesto Simplificado (Cliente)</i>	41
CAPÍTULO 5. ANÁLISIS.....	43
5.1 DEFINICIÓN DEL SISTEMA	43
5.1.1 <i>Determinación del Alcance del Sistema</i>	43
5.2 REQUISITOS DEL SISTEMA.....	44
5.2.1 <i>Obtención de los Requisitos del Sistema</i>	44
5.2.2 <i>Identificación de Actores del Sistema</i>	46
5.2.3 <i>Especificación de Casos de Uso</i>	46
5.3 IDENTIFICACIÓN DE LOS SUBSISTEMAS EN LA FASE DE ANÁLISIS.....	47
5.3.1 <i>Descripción de los Subsistemas</i>	48
5.3.2 <i>Descripción de los Interfaces entre Subsistemas</i>	48
5.4 DIAGRAMA DE CLASES PRELIMINAR DEL ANÁLISIS	48
5.4.1 <i>Diagrama de Clases</i>	48
5.4.2 <i>Descripción de las Clases</i>	49
5.5 ANÁLISIS DE CASOS DE USO Y ESCENARIOS	52
5.5.1 <i>Elegir jugadores</i>	52
5.5.2 <i>Iniciar partida</i>	53
5.5.3 <i>Realizar movimiento</i>	53
5.5.4 <i>Entrenar algoritmos</i>	54
5.6 RELACIÓN ESCENARIOS – CASOS DE USO – REQUISITOS	54
5.7 ANÁLISIS DE INTERFACES DE USUARIO.....	55
5.7.1 <i>Descripción de la Interfaz</i>	55
5.7.2 <i>Descripción del Comportamiento de la Interfaz</i>	56
5.8 ESPECIFICACIÓN DEL PLAN DE PRUEBAS	57

5.8.1	<i>Estudio experimental</i>	57
5.8.2	<i>Pruebas unitarias</i>	57
5.8.3	<i>Pruebas del sistema</i>	57
5.8.4	<i>Pruebas de rendimiento</i>	58
5.8.5	<i>Pruebas de usabilidad</i>	58
5.8.6	<i>Pruebas de accesibilidad</i>	58
CAPÍTULO 6. DISEÑO DEL SISTEMA.....		59
6.1	ARQUITECTURA DEL SISTEMA	59
6.1.1	<i>Diagramas de Paquetes</i>	59
6.1.2	<i>Diagramas de Componentes y de Despliegue</i>	62
6.2	DISEÑO DE CLASES.....	63
6.2.1	<i>Consideraciones sobre el Diseño</i>	65
6.3	DIAGRAMAS DE SECUENCIA.....	65
6.3.1	<i>Realizar Movimiento</i>	66
6.3.2	<i>Entrenar Inteligencias Artificiales</i>	67
6.4	DISEÑO DE PERSISTENCIA.....	67
6.4.1	<i>Descripción del Sistema de Persistencia Usado</i>	67
6.4.2	<i>Integración del Sistema de Persistencia</i>	68
6.5	DISEÑO DE LA INTERFAZ.....	68
6.6	ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS.....	74
6.6.1	<i>Pruebas Unitarias</i>	74
6.6.2	<i>Pruebas de Integración y del Sistema</i>	78
6.6.3	<i>Pruebas de Usabilidad y Accesibilidad</i>	79
6.6.4	<i>Pruebas de Rendimiento</i>	82
CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA		83
7.1	ESTÁNDARES Y NORMAS SEGUIDOS	83
7.2	LENGUAJES DE PROGRAMACIÓN.....	83
7.2.1	<i>Módulos y Librerías</i>	84
7.3	HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO.....	86
7.3.1	<i>PyCharm Community Edition</i>	86
7.3.2	<i>draw.io</i>	87
7.3.3	<i>Visual paradigm</i>	87
7.3.4	<i>Git</i>	88
7.3.5	<i>GitHub</i>	88
7.3.6	<i>Mendeley Reference Manager</i>	88
7.4	CREACIÓN DEL SISTEMA	89
7.4.1	<i>Problemas Encontrados</i>	89
7.4.2	<i>Descripción Detallada de las Clases</i>	92
CAPÍTULO 8. DESARROLLO DE LAS PRUEBAS		93
8.1	PRUEBAS UNITARIAS.....	93
8.2	PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA	94
8.3	PRUEBAS DE USABILIDAD Y ACCESIBILIDAD	94
8.3.1	<i>Pruebas de Usabilidad</i>	95
8.3.2	<i>Pruebas de Accesibilidad</i>	99
8.4	PRUEBAS DE RENDIMIENTO	100
CAPÍTULO 9. ESTUDIO TEÓRICO		103
9.1	A KNOWLEDGE-BASED APPROACH OF THE GAME OF AMAZONS (HENSGENS P, UITERWIJK J ET AL. 2001).....	103
9.2	AN EVALUATION FUNCTION FOR THE GAME OF AMAZONS, LIEBERUM J. 2005.....	106

9.3	COMPARATIVE STUDY OF MONTE-CARLO TREE SEARCH AND ALPHA-BETA PRUNING IN AMAZONS, KATO ET AL. 2015.....	107
9.4	MONTE-CARLO TREE SEARCH WITH EPSILON-GREEDY FOR GAME OF AMAZONS, TIAN C. 2023	107
CAPÍTULO 10. ESTUDIO EXPERIMENTAL		109
10.1	ALEATORIO.....	109
10.2	VORAZ	109
10.3	MINIMAX.....	110
10.4	MCTS	112
10.5	MINIMAX VS MCTS	114
10.6	COMPARACIÓN FINAL.....	115
CAPÍTULO 11. MANUALES DEL SISTEMA		116
11.1	MANUAL DE INSTALACIÓN	116
11.1.1	<i>Instalación para Usuario.....</i>	<i>116</i>
11.1.2	<i>Instalación para Desarrollador</i>	<i>116</i>
11.2	MANUAL DE EJECUCIÓN	117
11.2.1	<i>Ejecución para Usuario</i>	<i>117</i>
11.2.2	<i>Ejecución para Desarrollador</i>	<i>119</i>
11.3	MANUAL DE USUARIO.....	119
11.4	MANUAL DEL PROGRAMADOR	124
11.4.1	<i>Paquetes</i>	<i>124</i>
11.4.2	<i>Algoritmos Implementados.....</i>	<i>124</i>
11.4.3	<i>Realizar Modificaciones.....</i>	<i>126</i>
11.4.4	<i>Formato Ficheros de Configuración</i>	<i>126</i>
CAPÍTULO 12. CONCLUSIONES Y AMPLIACIONES		129
12.1	CONCLUSIONES	129
12.2	AMPLIACIONES.....	130
12.2.1	<i>Interfaz Gráfica Mejorada.....</i>	<i>130</i>
12.2.2	<i>Internacionalización de la Interfaz Gráfica</i>	<i>131</i>
12.2.3	<i>Añadir Nuevas Inteligencias Artificiales</i>	<i>131</i>
12.2.4	<i>Persistencia de Partidas.....</i>	<i>131</i>
CAPÍTULO 13. PLANIFICACIÓN DEL PROYECTO Y PRESUPUESTO FINALES.....		132
13.1	PLANIFICACIÓN FINAL.....	132
13.1.1	<i>Planificación temporal</i>	<i>132</i>
13.2	PRESUPUESTO FINAL.....	136
13.2.1	<i>Desarrollo de Presupuesto Detallado (Empresa).....</i>	<i>136</i>
CAPÍTULO 14. REFERENCIAS BIBLIOGRÁFICAS.....		141
14.1	REFERENCIAS CONSULTADAS.....	141
CAPÍTULO 15. APÉNDICES.....		147
15.1	GLOSARIO Y DICCIONARIO DE DATOS.....	147
15.2	CONTENIDO ENTREGADO EN EL ARCHIVO ADJUNTO.....	147
15.2.1	<i>Contenidos</i>	<i>147</i>
15.2.2	<i>Código Ejecutable e Instalación</i>	<i>150</i>
15.2.3	<i>Ficheros de Configuración</i>	<i>150</i>
15.3	CÓDIGO FUENTE.....	150
15.4	ACTAS DE REUNIONES.....	150
15.4.1	<i>Acta Reunión 1</i>	<i>151</i>

15.4.2	<i>Acta Reunión 2</i>	151
15.4.3	<i>Acta Reunión 3</i>	151
15.4.4	<i>Acta Reunión 4</i>	151
15.4.5	<i>Acta Reunión 5</i>	152
15.4.6	<i>Acta Reunión 6</i>	152
15.4.7	<i>Acta Reunión 7</i>	152
15.4.8	<i>Acta Reunión 8</i>	153
15.4.9	<i>Acta Reunión 9</i>	153
15.4.10	<i>Acta Reunión 10</i>	153
15.4.11	<i>Acta Reunión 11</i>	153
15.4.12	<i>Acta Reunión 12</i>	153
15.4.13	<i>Acta Reunión 13</i>	154
15.4.14	<i>Acta Reunión 14</i>	154

Índice de Figuras

Ilustración 1 MindSports, Amazons AI.....	20
Ilustración 2 The Game of the Amazons	21
Ilustración 3 Game of the Amazons	22
Ilustración 4 Microsoft, Game of the Amazons, en modo oscuro	23
Ilustración 5 Posición inicial del Juego de las Amazonas [11]	26
Ilustración 6 Posible primer movimiento en el Juego de las Amazonas [11]	27
Ilustración 7 Ejemplo de árbol generado por el algoritmo minimax [17]	29
Ilustración 8 Fases del algoritmo MCTS [22].....	30
Ilustración 9 Diagrama de Gantt con la planificación inicial	34
Ilustración 10 Diagrama de casos de uso.....	46
Ilustración 11 Diagrama de clases preliminar	49
Ilustración 12 Mockup de la interfaz gráfica del sistema	56
Ilustración 13 Diagrama de paquetes general.....	59
Ilustración 14 Diagrama de paquetes algoritmos	60
Ilustración 15 Diagrama de paquetes activos	60
Ilustración 16 Diagrama de paquetes pruebas.....	61
Ilustración 17 Diagrama de componentes y despliegue	63
Ilustración 18 Diagrama de clases.....	64
Ilustración 19 Diagrama de secuencia, realizar movimiento	66
Ilustración 20 Diagrama de secuencia, entrenar inteligencias artificiales.....	67
Ilustración 21 Interfaz gráfica del sistema por partes	69
Ilustración 22 Interfaz gráfica, menú negras desplegado	69
Ilustración 23 Interfaz gráfica, menú blancas desplegado	70
Ilustración 24 Interfaz gráfica, botón iniciar partida en estado inicial.....	71
Ilustración 25 Interfaz gráfica, botón iniciar partida oculto	71
Ilustración 26 Interfaz gráfica, botón iniciar partida tras terminar partida	72
Ilustración 27 Interfaz gráfica, amazona seleccionada	72
Ilustración 28 Interfaz gráfica, amazona movida	73
Ilustración 29 Intefaz gráfica, disparo realizado	73
Ilustración 30 Logo de PyCharm Community Edition	87
Ilustración 31 Logo de draw.io.....	87
Ilustración 32 Logo de Visual Paradigm Online	87
Ilustración 33 Logo de Git	88
Ilustración 34 Logo de GitHub	88
Ilustración 35 Logo de Mendeley.....	89
Ilustración 36 Código para creación de nodos a partir de acciones	90
Ilustración 37 Profiling del sistema.....	101
Ilustración 38 Profiling del sistema previo	102
Ilustración 39 Directorio en el explorador de archivos de Windows	117
Ilustración 40 Abrir una consola desde el explorador de archivos de Windows	118
Ilustración 41 Consola abierta desde el explorador de archivos de Windows.....	118
Ilustración 42 Ejecución gráfica en cmd con fichero de configuración por defecto	119
Ilustración 43 Ejecución entrenamiento en powershell con fichero de configuración personalizado	119
Ilustración 44 Jugador para blancas seleccionado y menú de elección para negras desplegado.....	120
Ilustración 45 Amazona seleccionada	121
Ilustración 46 Amazona movida.....	122

Ilustración 47 Casilla bloqueada.....	122
Ilustración 48 Partida terminada.....	123
Ilustración 49 Ilustración de los componentes de la ventana y el botón de cerrar	123

Índice de Tablas

Tabla 1 Tareas principales del proyecto	32
Tabla 2 Desglose de las tareas del proyecto.....	33
Tabla 3 Equipo de desarrollo	35
Tabla 4 Personal (I)	36
Tabla 5 Personal (II)	36
Tabla 6 Resumen de costes	36
Tabla 7 Costes indirectos.....	37
Tabla 8 Costes de producción	37
Tabla 9 Partida I, planificación del proyecto	38
Tabla 10 Partida II, análisis.....	38
Tabla 11 Partida III diseño	39
Tabla 12 Partida IV desarrollo.....	39
Tabla 13 Partida V pruebas	40
Tabla 14 Partida VI documentación.....	41
Tabla 15 Resumen del presupuesto	41
Tabla 16 Presupuesto del cliente.....	42
Tabla 17 Requisitos de alto nivel.....	44
Tabla 18 Caso de uso elegir jugadores.....	47
Tabla 19 Caso de uso iniciar partida.....	47
Tabla 20 Caso de uso realizar movimiento.....	47
Tabla 21 Caso de uso entrenar algoritmos.....	47
Tabla 22 Clase RandomAlgorithm	49
Tabla 23 Clase GreedyAlgorithm	50
Tabla 24 Clase MinimaxAlgorithm	50
Tabla 25 Clase MCTSAgorithm	50
Tabla 26 Clase AIPlayer	50
Tabla 27 Clase HumanPlayer	51
Tabla 28 Clase Board	51
Tabla 29 Clase GameGUI	52
Tabla 30 Análisis caso de uso elegir jugadores	53
Tabla 31 Análisis caso de uso iniciar partida.....	53
Tabla 32 Análisis caso de uso realizar movimiento	54
Tabla 33 Análisis caso de uso entrenar algoritmos	54
Tabla 34 Relación casos de uso - escenarios	55
Tabla 35 Clases de equivalencia __init__	74
Tabla 36 Pruebas __init__	75
Tabla 37 Clases de equivalencia is_valid_move (movimiento).....	75
Tabla 38 Clases de equivalencia is_valid_move (jugador).....	75
Tabla 39 Pruebas is_valid_move	76
Tabla 40 Clases de equivalencia __eq__	76
Tabla 41 Pruebas __eq__.....	76
Tabla 42 Clases de equivalencia get_legal_moves	77
Tabla 43 Pruebas get_legal_moves.....	77
Tabla 44 Clases de equivalencia is_win	77
Tabla 45 Pruebas is_win.....	78
Tabla 46 Casos de prueba Algorithms.....	78
Tabla 47 Pruebas Algorithms.....	78
Tabla 48 Procedimiento pruebas de integración del sistema	78

Tabla 49 Checklist pruebas de integración del sistema	79
Tabla 50 Preguntas de carácter general	80
Tabla 51 Preguntas cortas sobre la aplicación	81
Tabla 52 Observaciones	81
Tabla 53 Cuestionario para el responsable de las pruebas	81
Tabla 54 Cuestionario accesibilidad	82
Tabla 55 Comparación copias de tablero	90
Tabla 56 Resultado de los tests unitarios	94
Tabla 57 Resultados pruebas de integración y del sistema	94
Tabla 58 Preguntas de carácter general usuario 1	95
Tabla 59 Preguntas cortas sobre la aplicación usuario 1	96
Tabla 60 Cuestionario para el responsable de las pruebas sobre el usuario 1	96
Tabla 61 Preguntas de carácter general usuario 2	97
Tabla 62 Preguntas cortas sobre la aplicación usuario 2	97
Tabla 63 Observaciones usuario 2	97
Tabla 64 Cuestionario para el responsable de las pruebas sobre el usuario 2	98
Tabla 65 Preguntas de carácter general usuario 3	98
Tabla 66 Preguntas cortas sobre la aplicación usuario 3	99
Tabla 67 Observaciones usuario 3	99
Tabla 68 Cuestionario para el responsable de las pruebas sobre el usuario 3	99
Tabla 69 Resultado pruebas de accesibilidad	100
Tabla 70 Comparación voraz y aleatorio	110
Tabla 71 Comparación voraz territorial y voraz movilidad	110
Tabla 72 Comparación Minimax y aleatorio	111
Tabla 73 Comparación Minimax y voraz	111
Tabla 74 Comparación evaluaciones Minimax	112
Tabla 75 Comparación con y sin tabla histórica para Minimax	112
Tabla 76 Comparación MCTS y aleatorio	113
Tabla 77 Comparación MCTS y voraz	113
Tabla 78 Comparación implementaciones MCTS	114
Tabla 79 Comparación Minimax y MCTS	114
Tabla 80 Comparativa final	115
Tabla 81 Algoritmo aleatorio	124
Tabla 82 Algoritmo voraz	125
Tabla 83 Algoritmo Minimax	125
Tabla 84 Algoritmo MCTS	126
Tabla 85 Guía de nombrado de los algoritmos	128
Tabla 86 Tareas principales del proyecto	132
Tabla 87 Desglose de las tareas del proyecto	134
Tabla 88 Diagrama de Gantt con la planificación final	135
Tabla 89 Partida I, planificación del proyecto	136
Tabla 90 Partida II, análisis	137
Tabla 91 Partida III, diseño	137
Tabla 92 Partida IV, desarrollo	138
Tabla 93 Partida V, pruebas	138
Tabla 94 Partida VI, documentación	140
Tabla 95 Resumen del presupuesto	140
Tabla 96 Contenido del entregable	149

Índice de Ecuaciones

Ecuación 1 UCB1 score [21].....	30
Ecuación 2 Cálculo de la evaluación territorial en [34].....	91
Ecuación 3 Función para el balanceo de w y α	91
Ecuación 4 Cálculo retro propagación de Q modificado	92
Ecuación 5 Función UCB utilizada en [35]	107
Ecuación 6 Actualización del valor de una acción según [36]	108

Capítulo 1. Memoria del Proyecto

En este capítulo se presentará un resumen general del proyecto. Esto incluye la motivación, objetivos y alcance, así como una pequeña descripción de todos los capítulos.

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Los juegos de estrategia de 2 jugadores han formado parte del campo de estudio de la inteligencia artificial desde los comienzos de la informática. En este proyecto se utiliza el Juego de las Amazonas (ver 3.1) como base para el desarrollo de inteligencias artificiales capaces de jugar a este juego.

El proyecto incluye varias partes. Primero se realiza una investigación sobre artículos científicos relacionados con algoritmos inteligentes, en especial aquellos adaptados al Juego de las Amazonas. Después se diseñan y desarrollan varios algoritmos inteligentes. Tras su desarrollo se realiza un estudio comparativo para extraer información y poder compararlos entre ellos. Además, se presenta una interfaz gráfica sencilla para visualizar partidas.

El número de inteligencias artificiales desarrolladas queda limitado a 3 principales algoritmos. Algoritmos voraces (ver 3.2), Minimax (ver 3.3) y MCTS (ver 3.4). También se ha desarrollado un algoritmo que realiza movimientos de forma aleatoria con el fin de utilizarlo para evaluar las diferentes inteligencias artificiales. Para cada uno de ellos, a excepción del aleatorio, se han estudiado varios parámetros y se han desarrollado varias implementaciones diferentes. De esta forma se pueden comparar diferentes implementaciones de un algoritmo que siguen una misma estructura.

Para comparar los algoritmos y mejorarlos se simulan partidas entre ellos. Esto permite mejorar los algoritmos durante el desarrollo al observar su desempeño contra otras inteligencias artificiales y permitir al desarrollador probar distintas implementaciones o parámetros.

El proyecto incluye el desarrollo de una interfaz gráfica que permite que un usuario juegue partidas contra sí mismo, contra los algoritmos desarrollados o que elija 2 algoritmos para que jueguen una partida entre ellos. Sin embargo, este no es el objetivo principal del proyecto y da pie a que se desarrolle una nueva interfaz gráfica o que se mejore la existente en un futuro.

1.2 Resumen de Todos los Aspectos

En este apartado se resume cada uno de los contenidos de los capítulos de la documentación.

Capítulo 1: resumen y motivación del proyecto junto con los objetivos, el alcance y un resumen de cada capítulo.

Capítulo 2: introducción al proyecto que incluye una justificación, una enumeración de los objetivos y un estudio de la situación actual.

Capítulo 3: resumen de aspectos teóricos relacionados con el proyecto y que son de interés para que el lector entienda alguno de los conceptos fundamentales del proyecto.

Capítulo 4: planificación temporal y presupuesto inicial del proyecto.

Capítulo 5: análisis del sistema que incluye el alcance, requisitos, casos de uso, subsistemas, diagramas preliminares y pruebas.

Capítulo 6: diseño del sistema que incluye su arquitectura y diagramas de clases, de secuencia y el diseño de persistencia interfaz y plan de pruebas.

Capítulo 7: resumen de la implementación del sistema que incluye información sobre normas, lenguajes y herramientas utilizados en el proyecto, así como una explicación de problemas encontrados durante el desarrollo.

Capítulo 8: resultados y desarrollo de las diferentes pruebas que se realizan al sistema.

Capítulo 9: estudio teórico de artículos científicos consultados para la realización del proyecto.

Capítulo 10: estudio experimental en el que se compara el desempeño de las diferentes inteligencias artificiales.

Capítulo 11: manuales del sistema que incluyen la instalación, ejecución, y uso tanto para un usuario como para un desarrollador.

Capítulo 12: conclusiones del proyecto y posibles ampliaciones que se pueden realizar.

Capítulo 13: planificación temporal y presupuesto final del proyecto.

Capítulo 14: referencias bibliográficas consultadas durante el proyecto.

Capítulo 15: apéndices que contienen un glosario, un resumen del contenido del proyecto entregado y las actas de las reuniones.

Capítulo 2. Introducción

En esta sección se presentará el proyecto de forma breve. Se explicará cuál es el objetivo del proyecto y la finalidad de este, también se expondrán algunos proyectos similares para compararlos con el que se desarrollará.

2.1 Justificación del Proyecto

La idea detrás de este proyecto es desarrollar y comparar diferentes inteligencias artificiales en el juego de mesa de las Amazonas. El Juego de las Amazonas es interesante debido a su complejidad computacional que además varía según transcurre la partida. Debido a estas características es más complejo encontrar algoritmos de juego que logren tener un buen rendimiento tanto al principio como al final de la partida. Al desarrollar varias inteligencias artificiales se puede comparar las cualidades de estas para encontrar la que mejor funcione para este juego. Sin embargo, al comparar las inteligencias artificiales, también es posible encontrar características de cada una que puedan ser interesantes y que permita desarrollar mejores algoritmos en un futuro.

2.2 Objetivos del Proyecto

Los principales objetivos del proyecto son los siguientes:

1. Investigar sobre diferentes inteligencias artificiales desarrolladas o propuestas para el Juego de las Amazonas.
2. Desarrollar varias inteligencias artificiales capaces de jugar al Juego de las Amazonas
3. Entrenar a las inteligencias artificiales de forma que jueguen lo mejor posible.
4. Comparar las inteligencias artificiales haciendo que jueguen entre sí.
5. Examinar los resultados de las partidas para saber cuál o cuáles juegan mejor.
6. Sacar conclusiones en base a los resultados para identificar técnicas útiles en el campo de la inteligencia artificial, en especial en el Juego de las Amazonas y juegos de características similares.
7. Desarrollar una interfaz gráfica sencilla que permita jugar al Juego de las Amazonas con los algoritmos desarrollados.

2.3 Estudio de la Situación Actual

El proyecto presentado en este documento difiere en dos aspectos principales a las alternativas presentadas en los siguientes apartados. La primera diferencia es la capacidad de poder jugar contra varias inteligencias artificiales distintas, no solo una y además poder dejar que jueguen de forma autónoma dos inteligencias artificiales. La segunda es que no se centra en la interfaz gráfica ni en perfeccionar la jugabilidad. En otros sistemas se prioriza el juego por medio de una interfaz gráfica de varios jugadores de forma remota. Sin embargo, este proyecto se centra en comparar varias inteligencias

artificiales usando el Juego de las Amazonas y extraer información de estas comparaciones.

2.3.1 Evaluación de Alternativas

A continuación, se presentan varias alternativas consideradas en el proyecto. En el primer apartado se describen varios sistemas similares al que se desarrolla en este proyecto. Para cada uno de los sistemas se valorarán similitudes y diferencias con el sistema desarrollado en este proyecto y se enumerarán sus ventajas y desventajas. En el segundo apartado se comparan diferentes lenguajes de programación considerados para el proyecto, destacando ventajas y desventajas de cada uno de ellos.

2.3.1.1 Alternativas para el Juego de las Amazonas

2.3.1.1.1 MindSports, Amazons AI

La página web MindSports [1] contiene una versión digital de varios juegos de mesa, sobre todo juegos de mesa de dos jugadores y menos conocidos por el público general. Para los juegos de mesa presenta una interfaz gráfica sencilla y permite el juego offline entre dos personas o jugar en contra de una inteligencia artificial. Entre estos juegos de mesa se encuentra el Juego de las Amazonas.

Algunas de las similitudes con el proyecto son el poder jugar con una inteligencia artificial y también permitir juego entre humanos. Sin embargo, no presenta ni varias inteligencias artificiales ni se puede hacer que dos jugadores no-humanos jueguen entre sí.

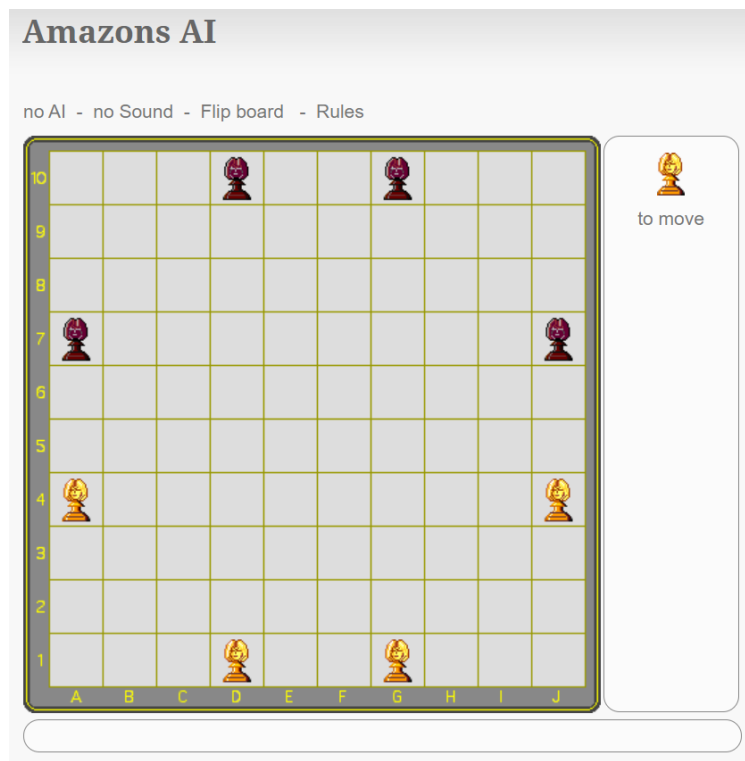


Ilustración 1 MindSports, Amazons AI

2.3.1.1.1.1 Ventajas

- **Interfaz gráfica sencilla y fácil de usar:** la interfaz gráfica, pese a ser sencilla, es bastante fácil de entender, también tiene las normas del juego a la derecha del tablero, haciendo que sea fácil aprender a jugar.
- **Juego humano-humano y humano-máquina:** como ya se comentó en la descripción de la alternativa, se puede jugar por turnos entre humanos y jugar contra una inteligencia artificial.

2.3.1.1.1.2 Desventajas

- **Solo se permite el juego con blancas:** no es posible jugar contra una inteligencia artificial y que empiece ella, solo se puede jugar si es el humano el que hace el primer movimiento.
- **Dificultad de la inteligencia artificial:** esta página web no permite variar la dificultad de la inteligencia artificial en ninguno de los juegos que presenta, incluido el Juego de las Amazonas. Además, las inteligencias artificiales de los juegos de la página web no superan el nivel humano. Esto se puede entender en juegos como el Juego de las Amazonas, un juego computacionalmente complejo en el que no hay muchas inteligencias artificiales y no es muy conocido. Sin embargo, resulta extraño que, en juegos como el ajedrez, para el que existen inteligencias artificiales muy superiores al humano, no se presente en esta página ninguna de este nivel.

2.3.1.1.2 The Game of the Amazons

En la siguiente referencia se puede encontrar una implementación del Juego de las Amazonas, [2]. En esta página se presenta una interfaz gráfica en la que se puede seleccionar el tipo de jugador que juega con blancas y el que juega con negras. En este aspecto es muy similar al proyecto que se describe en este documento. Se permite seleccionar 3 opciones para la inteligencia artificial. 'Smart' para un algoritmo inteligente (aunque no se menciona cuál), 'Random' para un algoritmo que realiza movimientos de forma aleatoria y 'None' para indicar que no será una inteligencia artificial sino un jugador humano. El sistema también tiene la similitud de permitir que jueguen dos inteligencias artificiales sin intervención humana.

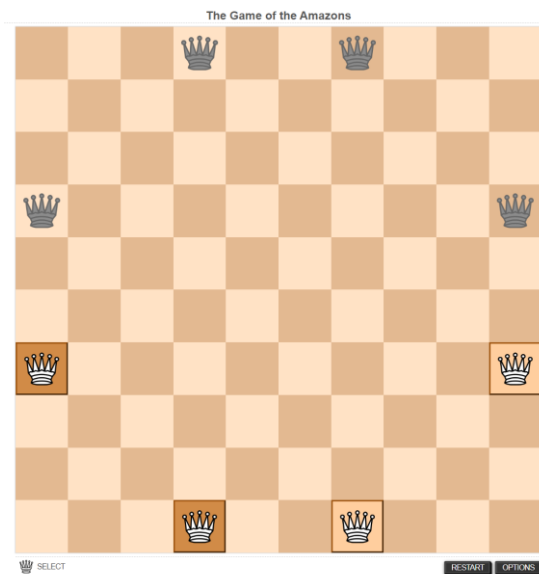


Ilustración 2 *The Game of the Amazons*

2.3.1.1.2.1 Ventajas

- **Modificabilidad:** las opciones de personalización son bastante completas. Desde modificar el tiempo que tarda la inteligencia artificial en hacer un movimiento hasta dos niveles de dificultad.
- **Aspecto interfaz gráfica:** el aspecto de la interfaz gráfica está muy logrado, tiene iconos para la fase de movimiento de la pieza y para la de disparo de flecha.

2.3.1.1.2.2 Desventajas

- **Bug con negras:** si al jugar con negras se selecciona una pieza, no se podrá seleccionar otra pieza para mover. Esto es frustrante para el usuario que puede querer mover con otra pieza. Esto es posiblemente un bug, ya que con las piezas blancas no pasa.

2.3.1.1.3 Game of the Amazons

Está disponible desde la Microsoft Store [3] y permite el juego entre humanos o entre humano e IA. A diferencia del sistema de este proyecto, solo presenta una inteligencia artificial y no se puede hacer que esta juegue contra otra. Se puede variar el tamaño del tablero entre el tradicional 10 x 10 y un tablero 8 x 8.

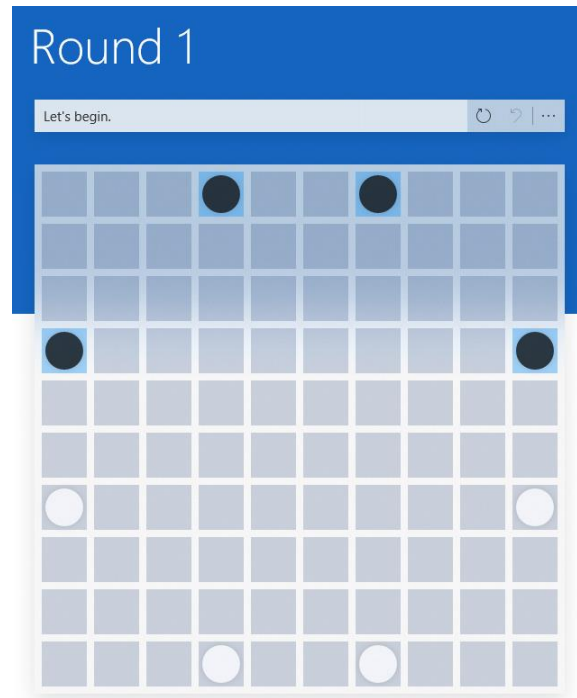


Ilustración 3 *Game of the Amazons*

2.3.1.1.3.1 *Ventajas*

- **Inteligencia artificial:** aunque solo haya una IA, esta es muy fuerte y es mucho mejor que las de las otras alternativas.
- **Funcionalidad extra:** permite el guardado y cargado de partidas del Juego de las Amazonas.

2.3.1.1.3.2 *Desventajas*

- **Problemas de color:** al utilizar el modo oscuro de Windows se deja de ver bien la mitad del tablero y con ello las piezas blancas. Esto solo pasa en el modo oscuro, en el modo claro se ve de forma normal. En la Ilustración 4 se puede ver una imagen con Windows en modo oscuro.

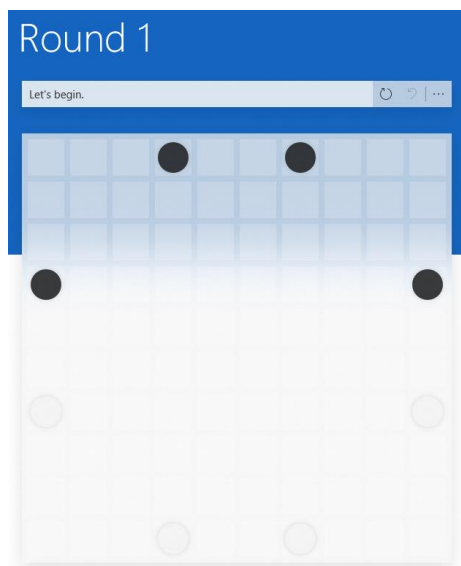


Ilustración 4 Microsoft, *Game of the Amazons*, en modo oscuro

- **Situación inicial no tradicional:** aunque no afecta mucho a cómo se juega, en esta implementación empiezan las fichas negras, no las blancas. Esto es bastante atípico ya que en la gran mayoría de implementaciones del Juego de las Amazonas es de forma inversa.

2.3.1.2 Alternativas para los lenguajes de programación

2.3.1.2.1 C / C++

El lenguaje de programación C y su extensión, C++, son lenguajes de programación con muchos años de historia y en muchos sentidos muy similares. El lenguaje C es uno de los lenguajes de programación de propósito general más usados en la historia de la programación. C++ surge como una evolución natural del lenguaje C y que busca mejorar algunos aspectos de este, por ejemplo, incluyendo la utilización de objetos. Como son similares en sintaxis y en utilización serán comparados de forma conjunta.

2.3.1.2.1.1 Ventajas

Velocidad: al ser lenguajes que se compilan directamente a ensamblador y usan características de bajo nivel como gestión manual de memoria o punteros, son mucho más rápidos que casi cualquier otro lenguaje de programación [4].

Uso de memoria: el uso de memoria puede llegar a ser muy eficiente ya que C y C++ proporcionan herramientas al desarrollador para gestionarla. De esta forma, se puede evitar usar más memoria de la necesaria. Sin embargo, esto puede ser también una desventaja puesto que el desarrollador tiene que aprender e invertir tiempo de desarrollo en la gestión de memoria que en otros lenguajes es automática [5].

2.3.1.2.1.2 Desventajas

Dificultad de uso: son lenguajes antiguos y difíciles de utilizar. El desarrollo suele ser más lento debido a que no incluyen características de lenguajes más modernos, como la gestión automática de memoria, recolectores de basura, etc.

Seguridad: el desarrollador debe prestar especial atención a la hora de utilizar direcciones de memoria, punteros y algunas estructuras de datos. Los lenguajes C y C++ no tienen mecanismos que eviten que se acceda a posiciones de memoria a las que no se debería tener acceso. Por ejemplo, si al iterar sobre un array no se escribe bien el código, es posible que se acceda por error a posiciones en memoria fuera del espacio reservado para el array. Esto puede causar que se modifique información que no tenga que ver con el programa y se produzca un error en el sistema.

2.3.1.2.2 Java

El lenguaje de programación Java es uno de los lenguajes de programación orientados a objetos más utilizados. Su amplia historia y su uso de la máquina virtual de Java (JVM) lo hacen una opción a considerar.

2.3.1.2.2.1 Ventajas

Orientación a objetos y tipado estático: la posibilidad de utilizar objetos da muchas posibilidades a la hora de programar, también facilita la programación el tipado estático que ayuda al programador evitando errores innecesarios.

JVM: la Java Virtual Machine [6] es un entorno que permite ejecutar código Java en prácticamente cualquier plataforma, independientemente de la plataforma en la que fue desarrollado. Esto hace que Java se pueda utilizar en proyectos muy diversos y en entornos muy diferentes con una misma plataforma de desarrollo.

2.3.1.2.2.2 Desventajas

Implementaciones de IA: pese a ser un lenguaje de programación bastante común, no existen muchas implementaciones de inteligencias artificiales en Java, sobre todo de redes neuronales y librerías de inteligencia artificial.

Verbosidad: al ser un lenguaje con un fuerte tipado estático y orientación a objetos es muy común que el código sea de difícil lectura y comprensión. De hecho, este fue uno de los factores (entre muchos otros) por los que JetBrains crea Kotlin en 2010 para el desarrollo móvil [7], [8].

2.3.1.2.3 Python

Python es un lenguaje de programación interpretado y de tipado dinámico que se ha vuelto muy popular en los últimos años, pese a ser más antiguo que otros. Es muy utilizado para ciencia de datos e inteligencia artificial.

2.3.1.2.3.1 Ventajas

Documentación sobre IA: existen muchas librerías y mucha documentación sobre inteligencia artificial en Python.

Fácil legibilidad: en cuanto a lenguajes de programación se refiere, Python es uno de los más sencillos en su sintaxis. Se parece mucho al lenguaje natural y esto hace que el código sea muy claro y se entienda de forma más rápida.

2.3.1.2.3.2 Desventajas

Velocidad: al ser un lenguaje de programación interpretado es varios órdenes de magnitud más lento que otros lenguajes de programación compilados.

GIL: el Global Interpreter Lock es una característica propia de Python que solo permite que un hilo ejecute código a la vez. Esta característica no es arbitraria, tiene un motivo por el cual existe (ver [9]). Sin embargo, hace más difícil utilizar programas multihilo, aunque existen formas de lidiar con ello.

Capítulo 3. Aspectos Teóricos

En este capítulo se presentarán algunos conceptos importantes y que se consideran necesarios para la comprensión del proyecto.

3.1 Juego de las Amazonas

El Juego de las Amazonas [10] o ‘Game of the Amazons’ en inglés es un juego de mesa para dos personas. Es un juego de estrategia en el que ambos jugadores tienen toda la información de la partida y en el que no existen elementos que dependan de la suerte. El Juego de las Amazonas fue inventado por el argentino Walter Zamkuskas en 1988.

El juego de las Amazonas se juega en un tablero similar al que se puede usar en ajedrez, pero de 10 x 10, al menos en su versión original. No es necesario que las casillas sean de dos colores como el tablero de ajedrez. En este tablero se colocan ocho piezas enfrentadas llamadas ‘amazonas’ de forma que a cada jugador le pertenecen cuatro de ellas. Generalmente las amazonas del jugador que realiza el primer movimiento son representadas de color blanco y las de su rival de color negro.

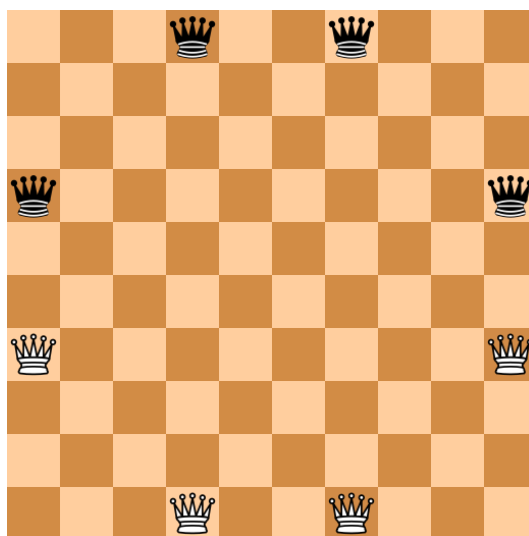


Ilustración 5 Posición inicial del Juego de las Amazonas [11]

Un movimiento en el Juego de las Amazonas consta de dos partes. Primero se mueve una amazona de su casilla a otra diferente como una reina del ajedrez. Esto es en línea recta diagonal u ortogonalmente. Además, no se pueden atravesar u ocupar casillas ocupadas por otra amazona, ya sea del propio jugador o de su rival. Luego con esa amazona se ‘dispara una flecha’ a otra casilla. Para esto se debe seleccionar una casilla libre que esté en una línea recta diagonal u ortogonal a partir de la casilla a la que se ha movido la amazona. La flecha no puede atravesar ni ocupar casillas ocupadas por una amazona. La casilla a la que se dispara la flecha queda marcada en el tablero y a partir de ese movimiento esta casilla no se podrá atravesar por flechas o amazonas de cualquier jugador. Conforme avanza la partida se van ocupando casillas del tablero con

flechas y se va reduciendo el número de casillas libres a las que poder mover las amazonas. El juego termina cuando un jugador ya no puede realizar ningún movimiento legal. El último jugador que haya movido es el ganador.

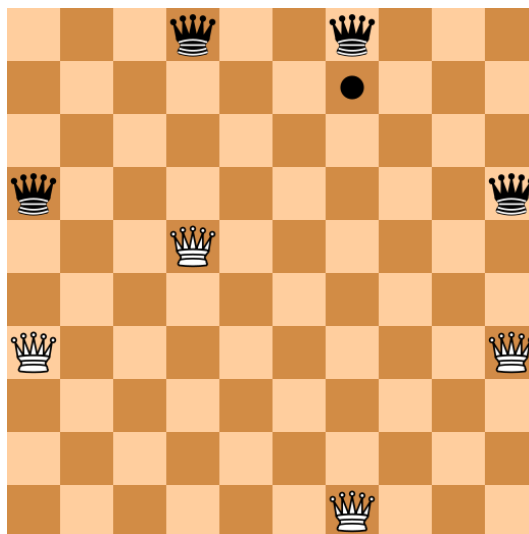


Ilustración 6 Posible primer movimiento en el Juego de las Amazonas [11]

De forma similar a otros juegos de mesa, una partida del Juego de las Amazonas se puede dividir en fases. Lo más normal es dividirlo en fase inicial (opening), mitad de partida (middle game) y fin de partida (endgame). Es difícil poner límites entre unas fases y otras ya que no hay una forma exacta de determinar cuándo una partida cambia de fase. Sin embargo, existe una parte de las partidas del Juego de las Amazonas que es sencillo identificar. Esta fase es llamada fase de relleno o 'filling phase' en inglés. Esta fase comprende la parte final de la partida en la que cada casilla libre solo puede ser alcanzada por un jugador. De esta forma, si cada jugador ejecuta una secuencia de movimientos óptima, se puede saber *a priori* quién es el ganador.

3.2 Algoritmo Voraz

Un algoritmo voraz [12] (en inglés greedy y conocido también en la literatura como ávido o devorador) es un tipo de algoritmo que se basa en la selección de la mejor opción local basándose en un heurístico. Una de las principales características de los algoritmos voraces es que únicamente consideran el estado de un problema en un instante determinado. De otra forma, las decisiones que se hayan tomado previamente en la resolución de un problema no afectan a la siguiente decisión que se tome.

Los algoritmos voraces utilizan un heurístico (ver apartado 15.1) que guía la búsqueda. Al depender del heurístico, la calidad de este afecta a la eficacia del algoritmo. Hay que tener en cuenta que los algoritmos voraces encuentran una solución óptima local utilizando un heurístico para dirigir la búsqueda. Esto puede hacer que se ignoren caminos que lleven a una solución óptima global y que esta nunca se encuentre.

Los algoritmos voraces suelen ser relativamente sencillos. Esto hace que generalmente sean más rápidos y simples que otros tipos de algoritmos. Son muy útiles en contextos

en los que se requiere una solución rápida y aproximada para un problema de gran complejidad (por ejemplo, TSP [13]). Si se conoce un buen heurístico para el problema a resolver puede ser interesante utilizar un algoritmo voraz frente a otro tipo de algoritmos.

Dependiendo de la naturaleza del problema puede ser que no exista un buen heurístico. En estos casos los algoritmos voraces pueden alejarse mucho de la solución óptima [14]. Esto es debido a que los algoritmos voraces encuentran una solución óptima local que puede no coincidir con la solución óptima global del problema en cuestión.

3.3 Minimax

El algoritmo minimax [15] es un algoritmo de decisión usado en gran medida para juegos de dos adversarios con información perfecta de suma cero [16]. El principal objetivo del algoritmo es tomar la mejor decisión en una posición para un determinado jugador.

Para el algoritmo, un jugador trata de maximizar el valor de la partida, llamado jugador maximizador, mientras que el otro jugador trata de minimizarlo, jugador minimizador. Si el valor de la partida en una posición es muy alto, el jugador maximizador va ganando, mientras que, si el valor de la partida es muy pequeño, va ganando el jugador minimizador.

Para tomar una decisión en una posición de la partida, el algoritmo minimax crea un árbol a partir de dicha posición. El nodo raíz sería la posición inicial y el resto de los nodos que parten de él son los estados a los que se llega con cada posible movimiento. Para cada nodo se repite el proceso de la misma forma.

Como se trata de un algoritmo para juegos de dos jugadores adversarios, cada nivel se alcanza con los movimientos de un jugador. Por ejemplo, si en una posición determinada donde pertenece el turno a las negras se ejecuta el algoritmo, el nodo raíz pertenece a las negras. El siguiente nivel pertenecería a las blancas, el siguiente a las negras y así sucesivamente hasta una profundidad determinada. En lenguaje formal un jugador es el maximizador y el otro el minimizador.

Para saber cuál de los posibles movimientos a partir de la posición de partida es el mejor es necesario darles una evaluación. Cuando se llega a los nodos hoja o terminales (aquellos que representan una partida finalizada) se les da un valor. Infinito positivo si ha ganado el jugador maximizador en ese estado, infinito negativo si ha ganado el jugador minimizador y cero en caso de empate. Como en muchos casos no se llega a desarrollar el árbol hasta nodos terminales, los nodos que estén a una cierta profundidad se evaluarán con una función. La función de evaluación debería dar valores positivos en caso de que vaya ganando el jugador maximizador y valores negativos en el caso contrario. Utilizando estas evaluaciones se va pasando el valor hacia arriba hasta llegar a la raíz del árbol. Para pasar el valor de nodos hijo a nodos padre hay que tener en cuenta que un nodo que pertenezca al jugador maximizador deberá tener la evaluación más alta de sus nodos hijo. Lo contrario también se cumple, la evaluación de un nodo del jugador minimizador será la menor de las evaluaciones de sus nodos hijo. De esta

forma el jugador maximizador intenta que la evaluación sea la mayor posible y el jugador minimizador que sea la menor posible.

Una vez que se haya transmitido la información hasta el nodo raíz se tendrá dos piezas de información muy valiosas. La primera es qué movimiento a partir de la posición de partida es el mejor (el mayor valor de evaluación en caso de que en la posición inicial le tocara al jugador maximizador y el menor valor en caso de que le tocara al minimizador). La segunda es una estimación de cómo va la partida, cuanto mayor sea el valor del nodo raíz mayor ventaja tendrá el jugador maximizador y viceversa.

Se puede ver un ejemplo de un árbol de minimax en la Ilustración 7.

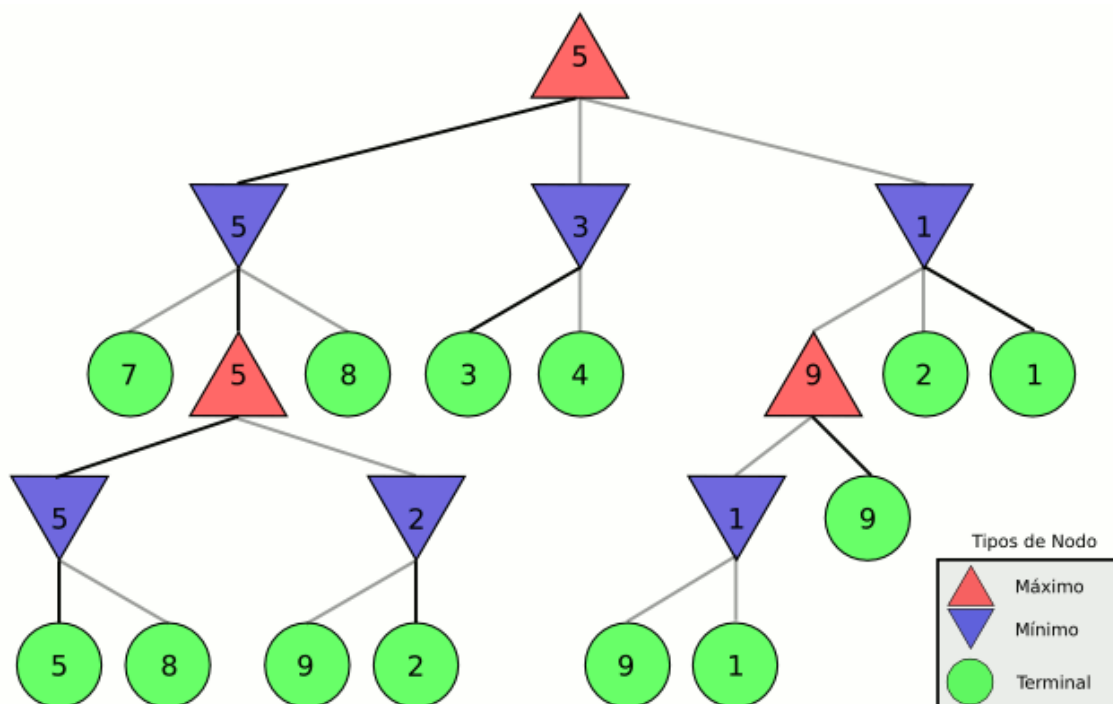


Ilustración 7 Ejemplo de árbol generado por el algoritmo minimax [17]

En ocasiones, para reducir el tamaño del árbol de búsqueda se emplean técnicas como Alpha-Beta [18]. Esta técnica consiste en emplear dos valores numéricos (Alpha y Beta) que se utilizan para podar partes del árbol de búsqueda y así reducir su tamaño.

3.4 Monte Carlo Tree Search

El algoritmo de búsqueda en árbol de Monte Carlo [19], en inglés, Monte Carlo Tree Search, conocido por sus siglas MCTS, es un algoritmo de búsqueda basado en simulación. El método de Monte Carlo [20] tiene muchas aplicaciones en informática y se basa en la simulación aleatoria de sucesos bajo la premisa de que la salida puede ser incorrecta con cierta probabilidad. En este caso se aplica el mismo concepto para búsqueda en árboles.

La idea detrás del algoritmo es crear un árbol de búsqueda que balancee la explotación de nodos con alto potencial y la exploración de nuevos nodos que puedan ser mejores.

El algoritmo se divide en cuatro fases: selección, expansión, simulación y retropropagación. Selection, expansion, simulation and backpropagation en inglés.

En la fase de selección se calcula la puntuación (hay varias fórmulas distintas, una muy común es UCB, véase Ecuación 1) de cada uno de los nodos del árbol. Aquel nodo con mayor puntuación que haya sido visitado por lo menos una vez será el que se expanda en la iteración actual. Si no ha sido visitado previamente se pasaría directamente a la fase de simulación.

$$UCB1(s_i) = \frac{w_i}{s_i} + c \sqrt{\frac{\ln s_p}{s_i}}$$

Ecuación 1 UCB1 score [21]

w_i : número de simulaciones del nodo resultantes en victoria

s_i : número de simulaciones totales del nodo

s_p : número total de simulaciones del nodo padre

En la fase de expansión, se obtienen nuevos nodos a partir del nodo a expandir. Uno de estos nodos será desde el que se realizará la siguiente fase, la fase de simulación.

En la fase de simulación se parte de un nodo y se toman movimientos aleatorios hasta llegar a un estado final. Al resultado se le otorga un valor en función del ganador de la partida. Una vez se obtiene ese valor es necesario transmitirlo hacia atrás hasta la raíz del árbol durante la fase de retropropagación.

En la fase de retropropagación se modifica el número de visitas y la puntuación total de cada uno de los nodos entre la raíz y el nodo del que parte la simulación ambos *inclusive*.

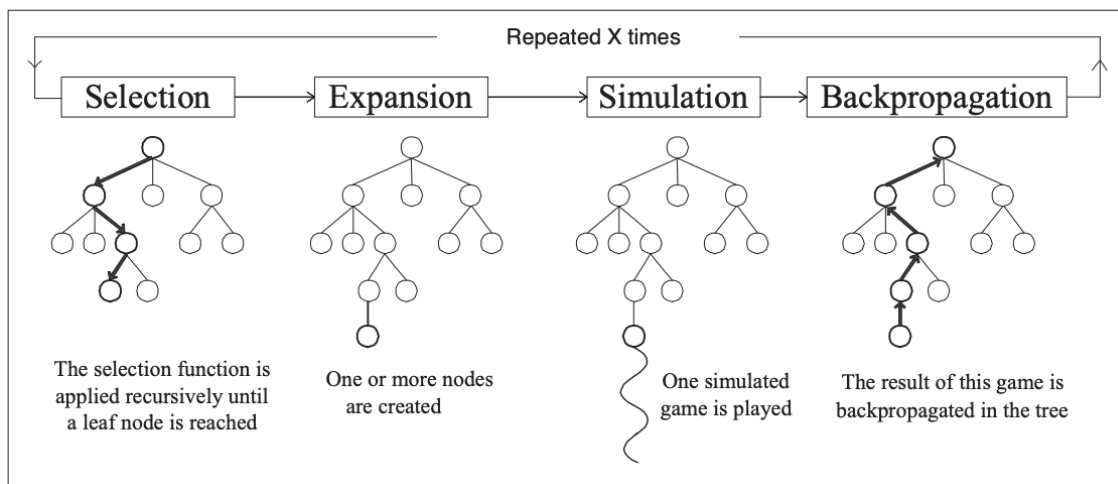


Ilustración 8 Fases del algoritmo MCTS [22]

3.5 Alpha Zero

En octubre de 2015 se celebró una serie de partidas del juego de mesa Go [23] entre una inteligencia artificial desarrollada por Google DeepMind y el jugador profesional Fan Hui. La inteligencia artificial, llamada AlphaGo [24], ganó 5-0, siendo la primera inteligencia artificial capaz de superar a un jugador profesional de Go. En octubre de

2017, el equipo detrás de AlphaGo publica un artículo con una versión mejorada llamada AlphaGo Zero [25]. Esta versión no requiere de conocimiento humano y ganó 100-0 al anteriormente mencionado AlphaGo. En diciembre del mismo año se presenta una versión generalizada de AlphaGo Zero, AlphaZero [26]. Esta versión fue capaz de, tras tan solo 24 horas, superar el nivel y derrotar a las inteligencias artificiales dominantes en ajedrez, shogi y Go (Stockfish [27], Elmo [28] y AlphaGo Zero [25] respectivamente).

Cada una de las versiones tratadas en el paso anterior y más concretamente AlphaZero, se basan en un árbol de búsqueda de Monte Carlo (ver Monte Carlo Tree Search) y dos redes neuronales. La idea es evitar el tiempo que tarda el árbol de búsqueda en generar información sobre los nodos más prometedores mediante simulaciones utilizando redes neuronales. Una de las redes neuronales, llamada '*value network*', se encarga de dar una estimación del valor de un nodo antes de realizar una simulación. De esta forma no es necesario simular partidas ya que se aproxima el valor mediante una red neuronal. La otra red, llamada '*policy network*', se encarga de ayudar al algoritmo MCTS a decidir qué nodos son los más prometedores. Esto lo hace dándole una estimación de lo prometedor que es cada nodo.

Capítulo 4. Planificación del Proyecto y Presupuestos Iniciales

En este capítulo se detalla la planificación y el presupuesto iniciales del proyecto. Tanto la planificación como el presupuesto son estimaciones por lo que no reflejarán de forma exacta el desarrollo del proyecto. La idea es que sirvan de base y que puedan ser comparadas con la planificación y presupuesto final en el [Capítulo 13 Planificación del Proyecto](#).

4.1 Planificación Inicial

La fecha que se ha establecido como el inicio del proyecto es el 23/11/2023. Se estima como fecha de finalización el 10/05/2024. Tanto la planificación como el presupuesto se ha realizado desde dos enfoques distintos. Por una parte, el alumno ha desarrollado las tareas del proyecto por sí mismo con una jornada de 2 horas al día, contando fines de semana y festivos, 14 horas semanales. Por otra parte, se ha realizado una planificación simulando una situación real de una pequeña empresa de desarrollo de software. En esta empresa se utilizaría un horario laboral regular de 8 horas al día los días laborales y se repartirían las tareas entre los diferentes trabajadores encargados del proyecto.

4.1.1 Planificación temporal

En la Tabla 1 se resume la estimación temporal para las partes principales del proyecto. En la Tabla 2 se detalla cada parte, dividiéndolas en tareas más pequeñas.

Número de esquema	Nombre de tarea	Duración
1	TFG	314 horas
1.1	Planificación del proyecto	21 horas
1.2	Análisis	40 horas
1.3	Diseño	32 horas
1.4	Desarrollo	112 horas
1.5	Pruebas	44 horas
1.6	Documentación	65 horas

Tabla 1 Tareas principales del proyecto

Número de esquema	Nombre de tarea	Duración
1	TFG	330 horas
1.1	Planificación del proyecto	21 horas
1.1.1	Elección de recursos humanos	5 horas
1.1.2	Estimación de tareas	8 horas
1.1.3	Estimación de costes	8 horas

1.2	Análisis	40 horas
1.2.1	Estudio de alternativas y viabilidad	8 horas
1.2.2	Estudio de artículos científicos	24 horas
1.2.3	Obtención de requisitos	8 horas
1.3	Diseño	32 horas
1.3.1	Diseño de la arquitectura del proyecto	16 horas
1.3.2	Diseño de las clases del proyecto	16 horas
1.4	Desarrollo	112 horas
1.4.1	Desarrollo de la lógica de juego	16 horas
1.4.2	Desarrollo de la interfaz gráfica	24 horas
1.4.3	Desarrollo del algoritmo voraz	8 horas
1.4.4	Desarrollo del algoritmo Minimax	32 horas
1.4.5	Desarrollo del algoritmo de MCTS	32 horas
1.5	Pruebas	44 horas
1.5.1	Diseño de pruebas	10 horas
1.5.2	Desarrollo de pruebas	10 horas
1.5.3	Comparación entre los diferentes algoritmos	24 horas
1.6	Documentación	65 horas
1.6.1	Introducción	2 horas
1.6.2	Aspectos teóricos	6 horas
1.6.3	Análisis	8 horas
1.6.4	Diseño del sistema	8 horas
1.6.5	Implementación del sistema	8 horas
1.6.6	Desarrollo de las pruebas	6 horas
1.6.7	Manuales del sistema	3 horas
1.6.8	Conclusiones y ampliaciones	6 horas
1.6.9	Planificación del proyecto y presupuestos finales	8 horas
1.6.10	Referencias bibliográficas	2 horas
1.6.11	Apéndices	8 horas

Tabla 2 Desglose de las tareas del proyecto

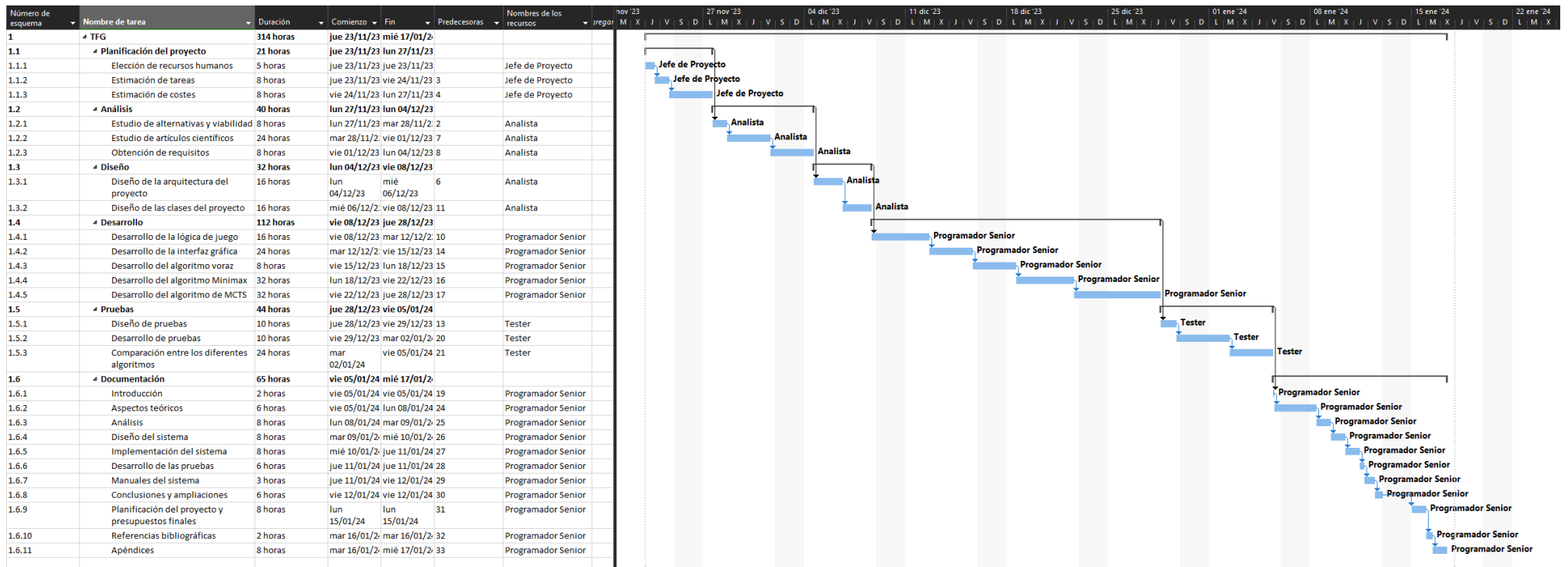


Ilustración 9 Diagrama de Gantt con la planificación inicial

4.2 Presupuesto Inicial

Aunque el proyecto sea realizado por un solo estudiante, el presupuesto inicial ha sido realizado dividiendo las tareas entre varios perfiles profesionales y por lo tanto diferentes salarios. El presupuesto ha sido realizado simulando el funcionamiento y los gastos de una empresa.

4.2.1 Definición de empresa

Para el proyecto se asume una pequeña empresa de desarrollo de software de alrededor de diez empleados. Entre ellos se encuentra un programador senior, un tester y un analista de software que, junto al jefe de proyecto, serán los encargados de llevar a cabo el proyecto. El desarrollador senior será el encargado de realizar el desarrollo, así como la documentación, el tester realizará el diseño y desarrollo de los diferentes tipos de pruebas que pasará el software y por último el analista se encargará de realizar las investigaciones iniciales y de realizar el diseño del software a implementar.

En la Tabla 3 se muestra cada miembro del equipo encargado del proyecto con su respectivo salario.

Identificador	Empleado	Función	Salario mensual	Salario por hora
1	Jefe de proyecto	Gestionar al equipo y planificar el proyecto	3.118€	19,49€
2	Programador Senior	Llevar a cabo la implementación del sistema y de las inteligencias artificiales	2.872€	17,95€
3	Tester	Realizar las pruebas del sistema	2.624€	16,40€
4	Analista	Realizar el análisis y el diseño del sistema a implementar	2.339€	14,62€

Tabla 3 Equipo de desarrollo

A continuación, en las tablas Tabla 4 y Tabla 5 se definen los costes asociados a los empleados de la empresa en mayor detalle. A cada empleado se le supone una productividad dentro de los proyectos de la empresa. El coste salarial es lo que le cuesta a la empresa cada empleado al año. A partir del coste salarial y la productividad se pueden calcular los costes directos. En la Tabla 5, la facturación indica lo que se espera ganar por cada empleado. En la Tabla 6 se muestra cómo la facturación total supera los costes de la empresa con un margen del 5%. Esto debería pasar en todos los proyectos de cualquier empresa, ya que, si no se superasen los costes, la empresa tendría pérdidas.

Personal	Num	Saldo Bruto Año	Coste Salarial Año	Total	Prod (%)	Coste Directo	CI (%)	Coste Indirecto
Jefe del proyecto	1	12.000,00 €	19.000,00 €	19.000,00 €	30%	5.700,00 €	70%	13.300,00 €
Programador senior	1	9.000,00 €	15.000,00 €	15.000,00 €	95%	14.250,00 €	5%	750,00 €
Tester	1	8.000,00 €	11.000,00 €	11.000,00 €	60%	6.600,00 €	40%	4.400,00 €
Analista	1	9.500,00 €	14.000,00 €	14.000,00 €	50%	7.000,00 €	50%	7.000,00 €
Total	4			59.000,00 €		33.550,00 €		25.450,00 €

Tabla 4 Personal (I)

Personal	Prod (%)	Horas / año	Horas productivas / año (por persona)	Horas productivas (total empresa)	Precio / hora	Facturación	Precio / hora (sin beneficios)
Jefe del proyecto	30%	2016	604,8	604,8	25,99 €	15.716,74 €	19,49 €
Programador senior	95%	2016	1915,2	1915,2	23,93 €	45.837,12 €	17,95 €
Tester	60%	2016	1209,6	1209,6	21,87 €	26.449,92 €	16,40 €
Analista	50%	2016	1008	1008	19,49 €	19.649,28 €	14,62 €
Total						107.653,06 €	

Tabla 5 Personal (II)

Resumen	
Concepto	Importe
Total de los costes directos	33.550,00 €
Total de los costes indirectos	48.192,58 €
Suma de los costes directos e indirectos	81.742,58 €
Beneficio deseado (25%)	20.435,65 €
Coste total (suma de los costes directos, indirectos y beneficios)	102.178,23 €
Facturación posible en función de las horas de producción y de los precios por hora calculados	107.653,06 €
Margen entre el coste total y la facturación	5%

Tabla 6 Resumen de costes

La empresa no solo gasta en sus empleados, sino que tiene costes indirectos que luego se reflejan en el coste del proyecto. Estos gastos se muestran en la Tabla 7.

Costes Indirectos		
Servicio	Coste mes	Coste año
Alquileres o arrendamientos de inmuebles (locales, naves, oficinas)	1.000,00 €	12.000,00 €

Gastos de mantenimiento, reparación y conservación	90,00 €	1.080,00 €
Consumos de electricidad (excepto consumos para producción)	150,00 €	1.800,00 €
Consumos de combustible para calefacción	50,00 €	600,00 €
Consumos de agua	50,00 €	600,00 €
Primas de seguros	120,00 €	1.440,00 €
Portes y gastos de transporte	50,00 €	600,00 €
Gastos en material de oficina	10,00 €	120,00 €
Gastos financieros	150,00 €	1.800,00 €
Total	1.670,00 €	20.040,00 €

Tabla 7 Costes indirectos

Además de los costes indirectos, los costes de producción (aquellos que proceden del desarrollo del producto) también se incluyen en el presupuesto. Se muestran en la Tabla 8. A cada uno de los 4 componentes del equipo asignado al proyecto se le asigna un equipo. Al jefe de proyecto se le asigna un equipo con menos prestaciones y por lo tanto menos coste que a los miembros del equipo que realizan desarrollo de software. Esto se debe a que equipos que necesitan ejecutar software para el desarrollo del proyecto necesitan más recursos. A cada uno de los componentes del equipo se le asigna también un portátil. Para realizar el desarrollo, el programador senior y el tester tendrán que disponer de un entorno de desarrollo adecuado. En este caso usarán PyCharm Professional [42]. Por último, será necesario utilizar una máquina virtual para entrenar y comparar los algoritmos. El cálculo del precio se estimó utilizando [43]. Concretamente se estimó una máquina virtual en Londres con sistema operativo Ubuntu Linux, 1 vCPU, 2GB de RAM, un espacio de 80GB y un espacio de usuario de 10GB.

Costes de la producción						
Equipo / Licencia	Unid.	Precio	Coste Total	Coste año	Tipo	Plazo
Equipos de administración	1	1.000,00 €	1.000,00 €	250,00 €	Amortizac.	4
Equipos de desarrollo	3	1.400,00 €	4.200,00 €	1.050,00 €	Amortizac.	4
Portátiles	4	800,00 €	3.200,00 €	800,00 €	Amortizac.	4
Licencias de desarrollo	2	301,29 €	602,58 €	602,58 €	Alquiler	-
Máquina virtual	1	31,00 €	31,00 €	372,00 €	Alquiler	-

Tabla 8 Costes de producción

4.2.2 Desarrollo de Presupuesto Detallado (Empresa)

A continuación, se presentan una serie de partidas en las que se muestran los costes relacionados con cada una de las partes del proyecto.

Partida I: planificación del proyecto									
I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Planificación del proyecto						409,29 €
	01		Elección de recursos humanos					97,45 €	
		01	Jefe del proyecto	5	horas	19,49 €	97,45 €		
	02		Estimación de tareas					155,92 €	
		01	Jefe del proyecto	8	horas	19,49 €	155,92 €		
	03		Estimación de costes					155,92 €	
		01	Jefe del proyecto	8	horas	19,49 €	155,92 €		

Tabla 9 Partida I, planificación del proyecto

Partida II: análisis									
I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Análisis						584,80 €
	01		Estudio de alternativas y viabilidad					116,96 €	
		01	Analista	8	horas	14,62 €	116,96 €		
	02		Estudio de artículos científicos					350,88 €	
		01	Analista	24	horas	14,62 €	350,88 €		
	03		Obtención de requisitos					116,96 €	
		01	Analista	8	horas	14,62 €	116,96 €		

Tabla 10 Partida II, análisis

Partida III: diseño									
I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Diseño						467,84 €
	01		Diseño de la arquitectura del sistema					233,92 €	
		01	Analista	16	horas	14,62 €	233,92 €		

	02		Diseño de las clases del proyecto					233,92 €	
		01	Analista	16	horas	14,62 €	233,92 €		

Tabla 11 Partida III diseño**Partida IV: desarrollo**

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Desarrollo						2.010,40 €
	01		Desarrollo de la lógica del juego					287,20 €	
		01	Programador sénior	16	horas	17,95 €	287,20 €		
	02		Desarrollo de la interfaz gráfica					430,80 €	
		01	Programador sénior	24	horas	17,95 €	430,80 €		
	02		Desarrollo del algoritmo voraz					143,60 €	
		01	Programador sénior	8	horas	17,95 €	143,60 €		
	03		Desarrollo del algoritmo minimax					574,40 €	
		01	Programador sénior	32	horas	17,95 €	574,40 €		
	04		Desarrollo del algoritmo de MCTS					574,40 €	
		01	Programador sénior	32	horas	17,95 €	574,40 €		

Tabla 12 Partida IV desarrollo**Partida V: pruebas**

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Pruebas						721,60 €
	01		Diseño de pruebas					164,00 €	
		01	Tester	10	horas	16,40 €	164,00 €		
	02		Desarrollo de pruebas					164,00 €	

	01	Tester	10	horas	16,40 €	164,00 €		
	02	Comparación entre los diferentes algoritmos					393,60 €	
	01	Tester	24	horas	16,40 €	393,60 €		

Tabla 13 Partida V pruebas

Partida VI: documentación									
I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Documentación						1.166,75 €
	01		Introducción					35,90 €	
		01	Programador Senior	2	horas	17,95 €	35,90 €		
	02		Aspectos teóricos					107,70 €	
		01	Programador Senior	6	horas	17,95 €	107,70 €		
	03		Análisis					143,60 €	
		01	Programador Senior	8	horas	17,95 €	143,60 €		
	04		Diseño del sistema					143,60 €	
		01	Programador Senior	8	horas	17,95 €	143,60 €		
	05		Implementación del sistema					143,60 €	
		01	Programador Senior	8	horas	17,95 €	143,60 €		
	06		Desarrollo de las pruebas					107,70 €	
		01	Programador Senior	6	horas	17,95 €	107,70 €		
	07		Manuales del sistema					53,85 €	
		01	Programador Senior	3	horas	17,95 €	53,85 €		
	08		Conclusiones y ampliaciones					107,70 €	
		01	Programador Senior	6	horas	17,95 €	107,70 €		

09		Planificación del proyecto y presupuestos finales					143,60 €	
	01	Programador Senior	8	horas	17,95 €	143,60 €		
10		Referencias bibliográficas					35,90 €	
	01	Programador Senior	2	horas	17,95 €	35,90 €		
11		Apéndices					143,60 €	
	01	Programador Senior	8	horas	17,95 €	143,60 €		

Tabla 14 Partida VI documentación

En la tabla Tabla 15 se muestra un resumen general de las partidas del presupuesto.

Resumen del presupuesto							
I1	I2	Descripción	Cantidad	Unidades	Precio	Subtotal (2)	Total
01		Proyecto					5.360,68 €
	01	Planificación del proyecto	1	uds.	409,29 €	409,29 €	
	02	Análisis	1	uds.	584,80 €	584,80 €	
	03	Diseño	1	uds.	467,84 €	467,84 €	
	04	Desarrollo	1	uds.	2.010,40 €	2.010,40 €	
	05	Pruebas	1	uds.	721,60 €	721,60 €	
	06	Documentación	1	uds.	1.166,75 €	1.166,75 €	

Tabla 15 Resumen del presupuesto

4.2.3 Desarrollo de Presupuesto Simplificado (Cliente)

En esta sección se presenta el presupuesto simplificado del proyecto

Presupuesto		
COD	Partida	Total
1	Planificación del proyecto	511,61 €
2	Análisis	731,00 €
3	Diseño	584,80 €
4	Desarrollo	2.513,00 €
5	Pruebas	902,00 €
6	Documentación	1.458,44 €

Total (con IVA 21%)		8.108,03 €

Tabla 16. Este presupuesto es el que verá el cliente. El beneficio y el IVA ya están incluidos, la única información que se le proporciona al cliente es el total de cada partida habiendo incluido el beneficio.

Presupuesto		
COD	Partida	Total
1	Planificación del proyecto	511,61 €
2	Análisis	731,00 €
3	Diseño	584,80 €
4	Desarrollo	2.513,00 €
5	Pruebas	902,00 €
6	Documentación	1.458,44 €
Total (con IVA 21%)		8.108,03 €

Tabla 16 Presupuesto del cliente

Capítulo 5. Análisis

En este capítulo se presenta el análisis y los requisitos del sistema. A partir de los requisitos definidos en este capítulo se realiza el diseño y la implementación.

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

El sistema busca presentar una interfaz para jugar al Juego de las Amazonas y permitir tanto a un usuario humano como a una serie de inteligencias artificiales jugar. De esta forma se puede hacer que dos inteligencias artificiales jueguen entre ellas, que dos jugadores humanos jueguen entre ellos o que un jugador humano juegue contra una de las inteligencias artificiales.

Existirán 3 inteligencias artificiales distintas capaces de jugar al Juego de las Amazonas. Será posible referirse a cada una de ellas mediante el algoritmo que utilicen. Los algoritmos que se utilizarán son los siguientes: algoritmo voraz, Minimax y árbol de búsqueda de Monte Carlo. Aunque se pueda ampliar el número de inteligencias del sistema, las anteriormente mencionadas son las que se pretende desarrollar. Por lo tanto, estas son las que están dentro del alcance del proyecto.

La interfaz gráfica será muy sencilla por 2 motivos. El primero es que el proyecto se centre en el desarrollo de las inteligencias artificiales, que son el objetivo principal de este trabajo. El segundo es dejar la posibilidad a proyectos posteriores para que mejoren la interfaz gráfica. Esta interfaz gráfica tendrá un tablero para permitir que se jueguen las partidas y presentará la opción al usuario de elegir qué jugadores participarán en la partida (inteligencias artificiales o humanos).

5.2 Requisitos del Sistema

En este capítulo se enumeran los requisitos del sistema a implementar. Tanto los requisitos funcionales como los requisitos no funcionales.

5.2.1 Obtención de los Requisitos del Sistema

Antes de que se pase a describir los requisitos de bajo nivel, en la Tabla 17 se presentan los requisitos de alto nivel que ha de cumplir el sistema.

Requisitos de alto nivel
Elegir jugadores
Jugar al Juego de las Amazonas
Entrenar modelo

Tabla 17 Requisitos de alto nivel

Elegir jugadores

EJ1: Un usuario podrá elegir qué jugadores participarán en una partida

EJ1.1: Un usuario podrá elegir qué jugador jugará con blancas de entre los siguientes:

EJ1.1.1: Jugador humano

EJ1.1.2: Algoritmo voraz

EJ1.1.3: Minimax

EJ1.1.4: MCTS

EJ1.2: Un usuario podrá elegir qué jugador jugará con negras de entre los siguientes:

EJ1.2.1: Jugador humano

EJ1.2.2: Algoritmo voraz

EJ1.2.3: Minimax

EJ1.2.4: MCTS

Jugar al Juego de las Amazonas

JA1: Las blancas realizarán el primer movimiento

JA1.1: Un movimiento consiste en tres fases

JA1.1.1: El jugador seleccionará una de sus piezas que tenga movimientos legales

JA1.1.2: El jugador moverá la pieza en cualquier dirección

JA1.1.2.1: Ortogonal

JA1.1.2.1.1: Sin atravesar una casilla ocupada por una pieza

JA1.1.2.1.2: Sin atravesar una casilla bloqueada

JA1.1.2.2: Diagonal

JA1.1.2.2.1: Sin atravesar una casilla ocupada por una pieza

JA1.1.2.2.2: Sin atravesar una casilla bloqueada

JA1.1.3: El jugador elegirá una casilla para bloquear en cualquier dirección a partir de la nueva posición de la pieza

JA1.1.3.1: Ortogonal

JA1.1.3.1.1: Sin atravesar una casilla ocupada por una pieza

JA1.1.3.1.2: Sin atravesar una casilla bloqueada

JA1.1.3.2: Diagonal

JA1.1.3.2.1: Sin atravesar una casilla ocupada por una pieza

JA1.1.3.2.2: Sin atravesar una casilla bloqueada

JA2: Las piezas negras y blancas alternarán turnos hasta que un jugador gane siguiendo los movimientos descritos en JA1.1:

JA3: El juego terminará cuando un jugador no sea capaz de realizar un movimiento legal, según lo descrito en JA1.1:

JA3.1: El jugador que no sea capaz de realizar un movimiento legal será el perdedor de la partida

JA3.2: El último jugador que haya sido capaz de realizar un movimiento legal será el ganador de la partida

Entrenar Algoritmos

EA1: El desarrollador podrá entrenar los siguientes algoritmos

EA1.1: Algoritmo voraz

EA1.2: Algoritmo minimax

EA1.3: Algoritmo MCTS

EA2: Cada algoritmo será entrenado de una forma diferente

EA2.1: Algoritmo voraz

EA2.1.1: El desarrollador podrá elegir entre diferentes algoritmos voraces

EA2.1.2: El desarrollador podrá simular partidas entre diferentes algoritmos voraces

EA2.1.2.1: Se podrá ver los resultados de las partidas con el fin de escoger al mejor

EA2.2: Algoritmo minimax

EA2.2.1: El desarrollador podrá elegir diferente profundidad para el árbol generado por el algoritmo

EA2.2.1.1: La profundidad será mayor o igual a 1

EA2.3: Algoritmo MCTS

A continuación, se enumerarán los requisitos no funcionales.

RNF1: Un algoritmo deberá tardar menos de 1 minuto en realizar un movimiento.

RNF2: El proyecto debe ser desarrollado en el lenguaje de programación Python.

RNF3: Un usuario debe ser capaz de utilizar la interfaz tras leer las reglas del juego.

RNF4: Las partidas durante el entrenamiento deberán ser persistidas.

RNF5: Se debe poder elegir qué inteligencias artificiales aparecen en la interfaz gráfica.

RNF6: Se debe poder elegir las inteligencias artificiales que participan en el entrenamiento.

5.2.2 Identificación de Actores del Sistema

- **Jugador:** usuario humano que participa en el juego. Esto puede ser solo eligiendo las inteligencias artificiales que van a jugar, jugando contra una inteligencia artificial o jugando contra otro jugador humano. El jugador es capaz de elegir jugadores y comenzar partida.
- **Desarrollador:** usuario experto con acceso al código fuente y un sistema en el que entrenar y comparar los distintos algoritmos. El desarrollador es capaz de crear un entorno en el que dos inteligencias artificiales jueguen entre ellas a lo largo de numerosas partidas. De esta forma se pueden perfeccionar y comparar para saber cuál funciona mejor. El desarrollador también es capaz de acceder a los resultados de los encuentros.

5.2.3 Especificación de Casos de Uso

A continuación, en la Ilustración 10 se presenta un diagrama con los principales casos de uso del sistema.

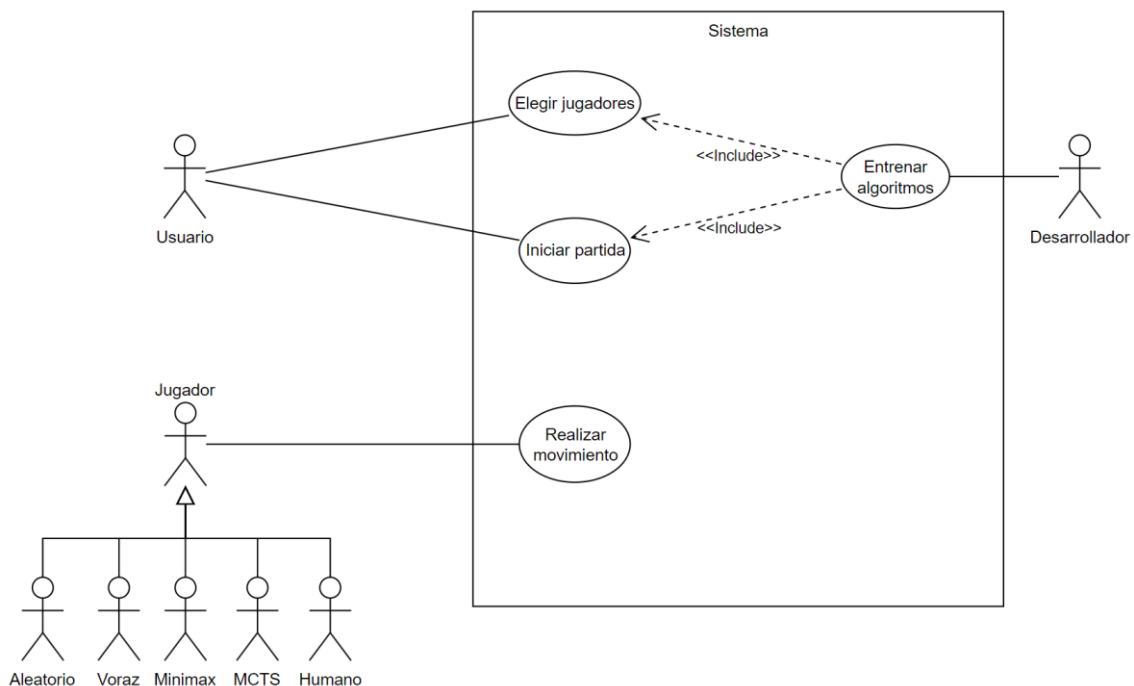


Ilustración 10 Diagrama de casos de uso

Elegir Jugadores
Elegir los jugadores que participarán en una partida
Descripción
Un usuario elegirá qué jugadores participarán en una partida del Juego de las Amazonas. Para ello se seleccionará un jugador para las piezas blancas y un jugador para las piezas negras de entre los siguientes posibles jugadores:

<ul style="list-style-type: none"> • Jugador Humano • Algoritmo Aleatorio • Algoritmo Voraz • Algoritmo Minimax • Algoritmo MCTS

Tabla 18 Caso de uso elegir jugadores

Iniciar Partida
Iniciar la partida
Descripción
Un usuario podrá iniciar una partida. Para ello se tiene que elegir a los jugadores que participarán en la partida previamente.

Tabla 19 Caso de uso iniciar partida

Realizar Movimiento
Realizar un movimiento durante una partida
Descripción
Un jugador realizará un movimiento legal durante la partida. Un movimiento consistirá en 3 pasos: <ul style="list-style-type: none"> • Seleccionar una pieza • Mover la pieza • Seleccionar una casilla a bloquear Tanto el movimiento de la pieza como la casilla a bloquear deben estar en una dirección ortogonal o diagonal sin atravesar casillas bloqueadas u otras piezas.

Tabla 20 Caso de uso realizar movimiento

Entrenar algoritmos
Elegir los mejores parámetros para los algoritmos y comparar su eficacia
Descripción
El desarrollador podrá entrenar a los algoritmos mediante la simulación de partidas entre ellos. Las simulaciones de partidas se harán para entrenar a los algoritmos y para compararlos entre ellos. Al realizar las simulaciones de partidas se elegirán los algoritmos a entrenar y se iniciarán un número elevado de partidas para obtener datos significativos.

Tabla 21 Caso de uso entrenar algoritmos

5.3 Identificación de los Subsistemas en la Fase de Análisis

En este apartado se dividirá el sistema en componentes más pequeños que luego serán tratados y explicados por separado para facilitar su comprensión.

El objetivo de esta sección es analizar el sistema para poder descomponerlo en sistemas más pequeños (subsistemas) que faciliten su posterior análisis.

5.3.1 Descripción de los Subsistemas

Los componentes que forman el sistema son 3:

- Interfaz Gráfica
- Jugadores
- Algoritmos
- Lógica

5.3.2 Descripción de los Interfaces entre Subsistemas

Los 4 subsistemas se comunicarán por medio de llamadas a métodos de clases de los distintos subsistemas. Esto quiere decir que se ejecutarán dentro de la misma máquina, de forma local.

5.4 Diagrama de Clases Preliminar del Análisis

En esta sección se mostrará un diagrama de clases del sistema. Este diagrama de clases está hecho antes del análisis, por lo tanto, aunque se conserve la estructura principal, puede haber algunos cambios entre este diagrama y el diagrama realizado tras el análisis.

5.4.1 Diagrama de Clases

En la Ilustración 11 se puede ver el diagrama de clases preliminar dividido en los diferentes subsistemas identificados anteriormente. Para entender el diagrama hay que considerar que el lenguaje de programación utilizado será Python.

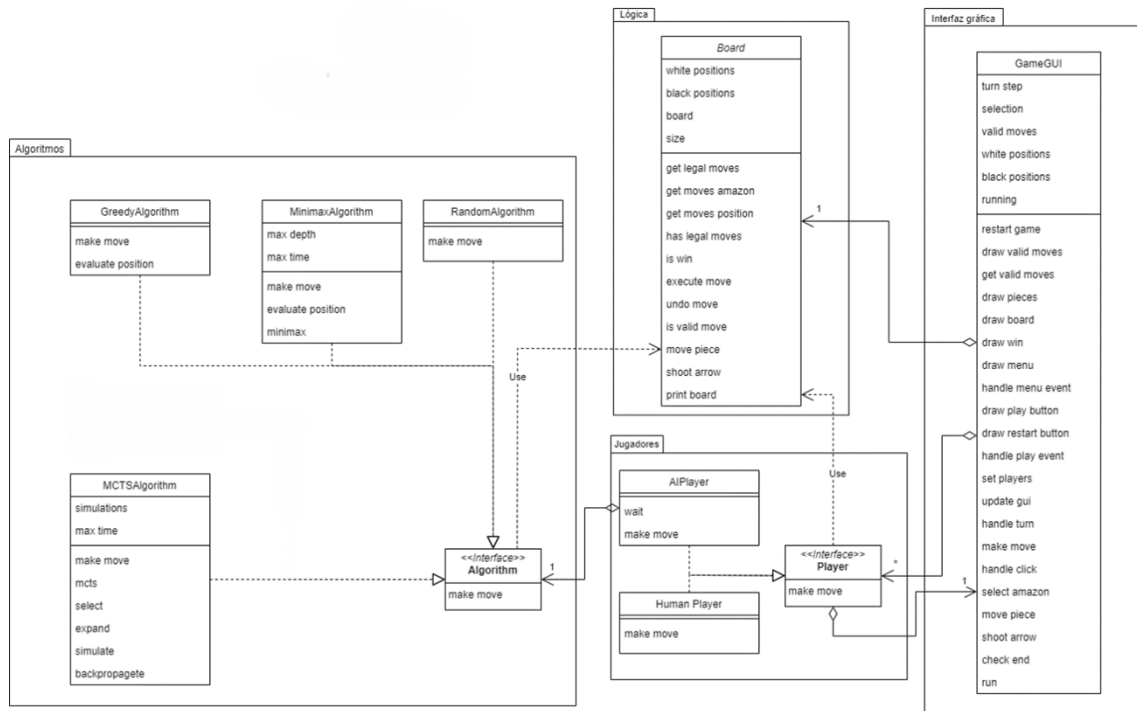


Ilustración 11 Diagrama de clases preliminar

5.4.2 Descripción de las Clases

A continuación, se describirá cada una de las clases del diagrama presentado en el apartado anterior.

5.4.2.1 Algoritmos

Nombre de la Clase	RandomAlgorithm
Descripción	Algoritmo aleatorio para el Juego de las Amazonas
Métodos Propuestos	make move: devuelve un movimiento para un jugador, este movimiento es determinado aleatoriamente de entre los posibles movimientos que tiene el jugador en un estado concreto de la partida.

Tabla 22 Clase RandomAlgorithm

Nombre de la Clase	GreedyAlgorithm
Descripción	Algoritmo voraz para el Juego de las Amazonas
Métodos Propuestos	make move: devuelve un movimiento para un jugador, este movimiento es determinado de forma heurística de entre los posibles movimientos que tiene el jugador en un estado concreto de la partida.

Tabla 23 Clase GreedyAlgorithm

Nombre de la Clase
MinimaxAlgorithm
Descripción
Algoritmo minimax para el Juego de las Amazonas
Atributos Propuestos
max depth: profundidad máxima a la que llegará el árbol minimax. max time: tiempo máximo que puede tomar el algoritmo minimax al crear el árbol.
Métodos Propuestos
make move: devuelve uno de los movimientos posibles para un jugador, este movimiento es determinado por el árbol minimax generado por el algoritmo usando como raíz un estado concreto de la partida. evaluate position: evalúa una posición concreta y devuelve un valor alto en caso de que sea una posición favorable para las piezas blancas y un valor bajo en caso de que sea una posición favorable para las piezas negras. minimax: crea un árbol de búsqueda siguiendo el algoritmo minimax.

Tabla 24 Clase MinimaxAlgorithm

Nombre de la Clase
MCTSAlgorithm
Descripción
Algoritmo de árbol de búsqueda de Monte Carlo para el Juego de las Amazonas
Atributos Propuestos
simulations: número de simulaciones que se harán durante la ejecución del algoritmo. max time: tiempo máximo que se ejecutará el algoritmo.
Métodos Propuestos
make move: devuelve un movimiento para un jugador, este movimiento es determinado mediante el uso de un árbol de búsqueda de Monte Carlo. mcts: implementa el algoritmo de búsqueda en árboles de Monte Carlo. select: implementa la fase de selección del algoritmo MCTS. expand: implementa la fase de expansión del algoritmo MCTS. simulate: implementa la fase de simulación del algoritmo MCTS. backpropagate: implementa la fase de retro propagación del algoritmo MCTS.

Tabla 25 Clase MCTSAlgorithm

5.4.2.2 Jugadores

Nombre de la Clase
AIPlayer
Descripción
Jugador que simula un jugador humano mediante la elección de movimientos siguiendo un algoritmo de inteligencia artificial.
Responsabilidades
Actuar como intermediario entre la interfaz gráfica y los algoritmos.
Atributos Propuestos
wait time: tiempo que debe tomar el jugador en realizar un movimiento.
Métodos Propuestos
wait: esperar para hacer más larga la duración de un movimiento. make move: realiza el movimiento sobre la interfaz gráfica gracias a un algoritmo.

Tabla 26 Clase AIPlayer

Nombre de la Clase
HumanPlayer
Descripción
Jugador que obtiene los movimientos del usuario al jugar al Juego de las Amazonas.
Responsabilidades
Actuar como intermediario entre la interfaz gráfica y el usuario.
Métodos Propuestos
make move: realiza el movimiento sobre la interfaz gráfica gracias a la interacción humana.

Tabla 27 Clase HumanPlayer

5.4.2.3 Lógica

Nombre de la Clase
Board
Descripción
Tablero que contiene toda la información sobre la partida.
Responsabilidades
Contener todos los datos de la partida, incluyendo el estado del tablero y las posiciones de las piezas y casillas bloqueadas.
Atributos Propuestos
white positions: lista con las posiciones de las piezas blancas. black positions: lista con las posiciones de las piezas negras. board: matriz que contiene la información del tablero con el contenido de cada casilla. size: dimensión del tablero.
Métodos Propuestos
get legal moves: obtener todos los movimientos legales a partir de una posición para un jugador concreto. get moves amazon: obtener los movimientos posibles para una pieza (solo los movimientos físicos de la pieza en el tablero, sin bloquear una casilla). get moves position: obtener los posibles movimientos ortogonales y diagonales que se pueden realizar a partir de una casilla. is win: devuelve un valor booleano para indicar si un jugador ha ganado la partida. execute move: ejecuta un movimiento sobre el tablero. is valid move: devuelve un valor booleano indicando si el movimiento es legal. move piece: mueve una pieza en el tablero. shoot arrow: bloquea una casilla en el tablero. print board: imprime por pantalla el tablero.

Tabla 28 Clase Board

5.4.2.4 Interfaz gráfica

Nombre de la Clase
GameGUI
Descripción
Interfaz gráfica que permite al usuario visualizar las partidas y elegir los jugadores que participarán en ellas.
Responsabilidades
Mostrar el tablero al usuario, permitirle realizar movimientos y elegir jugadores.
Atributos Propuestos
turn step: entero que marca de qué jugador es el turno y que parte del movimiento está haciendo.

<p>selection: pieza seleccionada en un instante determinado por un jugador.</p> <p>valid moves: lista de movimientos legales para un momento concreto del movimiento de un jugador.</p> <p>white positions: lista de posiciones de las piezas blancas.</p> <p>black positions: lista de posiciones de las piezas negras.</p> <p>running: valor booleano que indica si se está ejecutando la interfaz gráfica.</p>
Métodos Propuestos
<p>restart game: esperar para hacer más larga la duración de un movimiento.</p> <p>draw valid moves: dibuja en la interfaz gráfica los movimientos válidos al seleccionar una pieza.</p> <p>get valid moves: obtiene los movimientos válidos a partir de una posición.</p> <p>draw pieces: dibuja las piezas sobre la interfaz gráfica.</p> <p>draw board: dibuja el tablero sobre la interfaz gráfica.</p> <p>draw win: dibuja el resultado en caso de que un jugador gane la partida.</p> <p>draw menu: dibuja el menú de selección de jugadores sobre la interfaz gráfica.</p> <p>handle menu event: gestiona los eventos del menú.</p> <p>draw play button: dibuja el botón de iniciar la partida sobre la interfaz gráfica.</p> <p>draw restart button: dibuja el botón de reiniciar la partida sobre la interfaz gráfica tras la victoria de un jugador.</p> <p>handle play event: gestiona todas las acciones necesarias tras el inicio de la partida.</p> <p>set players: establece quiénes serán los jugadores (humanos o inteligencia artificial y qué algoritmo usa en ese caso).</p> <p>update gui: actualiza la interfaz gráfica para que se mantenga actualizada al estado actual de la partida.</p> <p>handle turn: gestiona un turno dándole la posibilidad de realizar un movimiento al jugador al que pertenezca el turno.</p> <p>make move: ejecuta un movimiento sobre el tablero y la interfaz gráfica.</p> <p>handle click: gestiona el clic del ratón.</p> <p>select amazon: ejecuta las acciones que se deben realizar al seleccionar una pieza (marcarla como seleccionada y mostrar los posibles movimientos que tiene).</p> <p>move piece: mueve una pieza en el tablero y lo muestra en la interfaz gráfica.</p> <p>shoot arrow: bloquea una casilla y muestra la casilla como bloqueada en la interfaz gráfica.</p> <p>check end: comprueba si la partida ha terminado y su resultado en caso de que sí lo haya hecho.</p> <p>run: método con el que se inicia la ejecución de la interfaz gráfica.</p>

Tabla 29 Clase GameGUI

5.5 Análisis de Casos de Uso y Escenarios

En esta sección se describirán en mayor detalle los casos de uso identificados en 5.2.3. Para cada caso de uso se presentará una tabla que describa el caso de uso, indicando: nombre, descripción, precondiciones, postcondiciones, situaciones de error, actores, disparador, proceso estándar y posibles procesos alternativos si es que los tiene.

5.5.1 Elegir jugadores

Escenario 1. Elegir jugadores	
Nombre	Elegir jugadores.
Precondiciones	El sistema está ejecutándose.
Postcondiciones	Se establecerá qué jugadores jugarán la partida.

Situaciones de error	No hay.
Actores	Usuario.
Proceso estándar	<p>(1.1.1) El usuario pulsa sobre el desplegable del jugador con piezas blancas.</p> <p>(1.1.2) El usuario selecciona uno de los posibles jugadores.</p> <p>(1.1.3) El usuario pulsa sobre el desplegable del jugador con piezas negras.</p> <p>(1.1.4) El usuario selecciona uno de los posibles jugadores.</p>
Procesos alternativos	<p>(1.2.1) El usuario pulsa sobre el desplegable del jugador con piezas negras.</p> <p>(1.2.2) El usuario selecciona uno de los posibles jugadores.</p> <p>(1.2.3) El usuario pulsa sobre el desplegable del jugador con piezas blancas.</p> <p>(1.2.4) El usuario selecciona uno de los posibles jugadores.</p> <p>(1.3.1) El usuario no cambia el jugador de piezas blancas.</p> <p>(1.3.2) El usuario pulsa sobre el desplegable del jugador con piezas negras.</p> <p>(1.3.3) El usuario selecciona uno de los posibles jugadores.</p> <p>(1.4.1) El usuario pulsa sobre el desplegable del jugador con piezas blancas.</p> <p>(1.4.2) El usuario selecciona uno de los posibles jugadores.</p> <p>(1.4.3) El usuario no cambia el jugador de piezas negras.</p> <p>(1.5.1) El usuario no cambia el jugador de piezas blancas.</p> <p>(1.5.2) El usuario no cambia el jugador de piezas negras.</p>

Tabla 30 Análisis caso de uso elegir jugadores

5.5.2 Iniciar partida

Escenario 2. Iniciar partida	
Nombre	Iniciar partida.
Precondiciones	El sistema está ejecutándose y se han seleccionado los jugadores (bien sea manual o automáticamente).
Postcondiciones	<p>(2.1.1) La partida comienza.</p> <p>(2.1.2) No es posible modificar los jugadores.</p>
Situaciones de error	No hay.
Actores	Usuario.
Proceso estándar	El usuario pulsa sobre el botón de iniciar partida.
Procesos alternativos	No hay.

Tabla 31 Análisis caso de uso iniciar partida

5.5.3 Realizar movimiento

Escenario 3. Realizar movimiento	
Nombre	Realizar movimiento.
Precondiciones	Se ha iniciado la partida.
Postcondiciones	El estado del tablero cambia.
Situaciones de error	<p>(3.1.1) El jugador (solo jugador humano) intenta realizar un movimiento ilegal.</p> <p>(3.1.2) El jugador (solo jugador humano) intenta realizar un movimiento después de que acabe la partida.</p>

Actores	Jugador.
Proceso estándar	(3.2.1) El jugador pulsa sobre una de sus piezas. (3.2.2) El jugador mueve la pieza a otra casilla. (3.2.3) El jugador selecciona una casilla a bloquear.
Procesos alternativos	No hay.

Tabla 32 Análisis caso de uso realizar movimiento

5.5.4 Entrenar algoritmos

Escenario 4. Entrenar algoritmos	
Nombre	Entrenar algoritmos.
Precondiciones	Los algoritmos han sido creados previamente por el desarrollador.
Postcondiciones	Comienzan una serie de partidas entre los algoritmos.
Situaciones de error	El hardware se apaga o se queda sin memoria.
Actores	Desarrollador.
Proceso estándar	(4.1.1) El desarrollador selecciona una o más inteligencias artificiales para entrenar. (4.1.2) El desarrollador selecciona el número de partidas. (4.1.3) El desarrollador inicia el proceso de entrenamiento.
Procesos alternativos	No hay.

Tabla 33 Análisis caso de uso entrenar algoritmos

5.6 Relación Escenarios – Casos de Uso – Requisitos

La numeración identificativa de los escenarios se ha escogido de tal forma que sea sencillo identificar el caso de uso a partir del escenario. El primer número del escenario represente el caso de uso al que está asociado. El segundo número lo relacionará con el grupo de escenarios dentro de un caso de uso (ya que puede haber varias secuencias de escenarios en un mismo caso de uso). El tercer y último número es el número de escenario dentro del grupo de escenarios al que pertenece.

A continuación, se muestra una tabla que representa la relación entre los casos de uso identificados y los escenarios que se tratan en cada uno de ellos. De esta forma es más sencillo entender visualmente la relación entre casos de uso y escenarios.

Casos de Uso / Escenarios:	1	2	3	4
1.1.1	X			
1.1.2	X			
1.1.3	X			
1.1.4	X			
1.2.1	X			
1.2.2	X			

1.2.3	X			
1.2.4	X			
1.3.1	X			
1.3.2	X			
1.3.3	X			
1.4.1	X			
1.4.2	X			
1.5.1	X			
1.5.2	X			
2.1.1		X		
2.1.2		X		
3.1.1			X	
3.1.2			X	
3.2.1			X	
3.2.2			X	
3.2.3			X	
4.1.1				x
4.1.2				x
4.1.3				x

Tabla 34 Relación casos de uso - escenarios

5.7 Análisis de Interfaces de Usuario

La interfaz gráfica de usuario del sistema no es el tema principal del trabajo. Por lo tanto, será una interfaz simple que permita al usuario realizar las acciones básicas comentadas en el apartado de Análisis de Casos de Uso y Escenarios.

5.7.1 Descripción de la Interfaz

La interfaz gráfica del sistema solo tendrá una pantalla. En esta pantalla se podrá elegir a los jugadores y jugar al Juego de las Amazonas. A continuación, en la Ilustración 12 Mockup de la interfaz gráfica del sistema se muestra un mockup (ver Ilustración 12) de la pantalla de la interfaz gráfica.

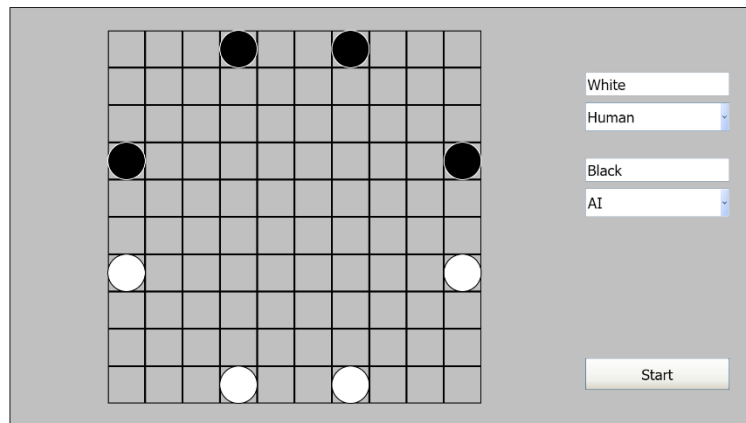


Ilustración 12 Mockup de la interfaz gráfica del sistema

5.7.2 Descripción del Comportamiento de la Interfaz

La única entrada de datos de la interfaz gráfica son las pulsaciones sobre la pantalla. Las posibles interacciones con la pantalla son las siguientes:

Botón de cerrar: el botón por defecto de la ventana (situado en la parte superior derecha de la ventana) es la única forma de cerrar la aplicación mediante la interfaz gráfica.

Botón de iniciar partida: el botón para iniciar partida (o reiniciar partida en caso de que haya terminado la partida)

Desplegables de elección de jugadores: el usuario puede interactuar con los desplegables para elegir los jugadores que participarán en la partida.

Movimientos en el tablero: durante la partida el usuario podrá realizar movimientos sobre el tablero mediante pulsaciones del ratón (solo si ha elegido al menos un jugador humano).

La propia interfaz gráfica se encargará de no permitir al usuario realizar acciones que pongan en riesgo la integridad del sistema. Por ejemplo, no se permitirá al usuario realizar movimientos ilegales, elegir algoritmos que no existan o pulsar el botón de iniciar partida durante una partida. De la misma forma, siempre le será posible al usuario pulsar el botón de cerrar la ventana.

En cuanto a la información proporcionada al usuario por parte de la interfaz, se encuentran las siguientes fuentes:

Desplegables que muestran los jugadores: además de elegir los jugadores, el usuario podrá ver qué jugadores están participando en la partida en todo momento.

Estado de la partida: el estado de la partida se puede dividir en 2 partes. La primera es el estado del tablero en una partida en curso que se podrá ver durante el transcurso de esta en la representación gráfica del tablero. La segunda parte es la información que hace saber al usuario si la partida está en curso o no. Esto lo podrá saber el usuario mediante el botón de iniciar partida ya que este solo aparece cuando la partida no ha empezado o ya ha acabado.

Movimientos posibles: los movimientos legales en una partida se indicarán marcando las casillas a las que se puede mover una pieza, así como las casillas que pueden ser bloqueadas por el usuario durante la segunda parte del movimiento.

Ganador: tras terminar una partida, se mostrará sobre el tablero un texto que indique si ha ganado el jugador con piezas blancas o el jugador con piezas negras (nótese que no existen empates en el Juego de las Amazonas).

5.8 Especificación del Plan de Pruebas

A continuación, se explicará cada uno de los tipos de pruebas que se llevarán a cabo en el sistema para garantizar su correcto funcionamiento y la ausencia de errores en la implementación.

5.8.1 Estudio experimental

Dentro del proyecto, uno de los principales objetivos es el de extraer información mediante la comparación de los diferentes algoritmos implementados. Por esto, el estudio experimental es un aspecto de mucha importancia. Debido a cómo se realizará este estudio, encaja dentro del plan de pruebas. Se seleccionarán por fases los diferentes algoritmos implementados y se compararán entre ellos simulando un elevado número de partidas.

5.8.2 Pruebas unitarias

Para reducir la cantidad de errores en la implementación del sistema se diseñarán y realizarán varias pruebas unitarias. Estas procurarán sobre todo comprobar el funcionamiento de la lógica del juego y del sistema.

5.8.3 Pruebas del sistema

Estas pruebas no serán automatizadas, al contrario que las dos categorías de pruebas anteriores. En estas pruebas se comprobará el funcionamiento general del sistema de manera manual. De esta forma se prueba que los diferentes subsistemas se hayan integrado correctamente y que permitan el correcto funcionamiento del sistema.

5.8.4 Pruebas de rendimiento

Para intentar mejorar el rendimiento y reducir el tiempo de ejecución del sistema se utilizarán pruebas de rendimiento. Durante estas pruebas se medirán tiempos y se analizarán las distintas partes del sistema para intentar optimizarlo en la medida de lo posible. Hay que tener en cuenta durante las pruebas de rendimiento la naturaleza del sistema, así como las partes más importantes del mismo. De esta forma se podrá decidir si es necesario optimizar el sistema, en qué medida y en qué partes. El principal objetivo del proyecto no es desarrollar un sistema extremadamente eficiente por lo que se buscará que el rendimiento facilite un uso normal del sistema y permita el estudio de los algoritmos implementados.

5.8.5 Pruebas de usabilidad

Debido a que la interfaz gráfica es un elemento de menor importancia en los objetivos del proyecto, las pruebas de usabilidad de la interfaz no serán muy exhaustivas. Con estas pruebas se buscará comprobar que la interfaz sea usable y sencilla para que cualquier usuario pueda utilizarla sin complicaciones.

5.8.6 Pruebas de accesibilidad

Debido a que el sistema tiene interfaz gráfica, se desarrollarán pruebas de accesibilidad manuales sencillas para comprobar qué partes del sistema facilitan la accesibilidad y cuáles la dificultan. De esta forma, en posibles ampliaciones del sistema y cambios futuros que puedan realizarse sobre la interfaz, se tendrá una base para mejorar la accesibilidad.

Capítulo 6. Diseño del Sistema

En este capítulo se detallará el diseño del sistema completo. La fase de diseño es una de las más importantes en el desarrollo del software [44] y su éxito influye en gran medida en el éxito final del proyecto.

La fase de diseño se ha plasmado en este documento dividiéndolo en varios apartados. Arquitectura del sistema, diagramas de clases del sistema, diagramas de interacción y de estados, diagramas de actividades, diseño de persistencia, diseño de la interfaz gráfica y especificación técnica del plan de pruebas.

6.1 Arquitectura del Sistema

En este apartado se detallará la arquitectura del sistema y se dividirá en 3 subapartados. Diagramas de paquetes, diagramas de componentes y diagramas de despliegue.

6.1.1 Diagramas de Paquetes

El diagrama de paquetes se dividirá en varias partes más pequeñas para que sea más fácil entenderlas. Primero se presentará una visión general de todos los paquetes en conjunto para que se vean las relaciones entre ellos y después se mostrará cada parte por separado entrando en detalle en cada uno de ellos.

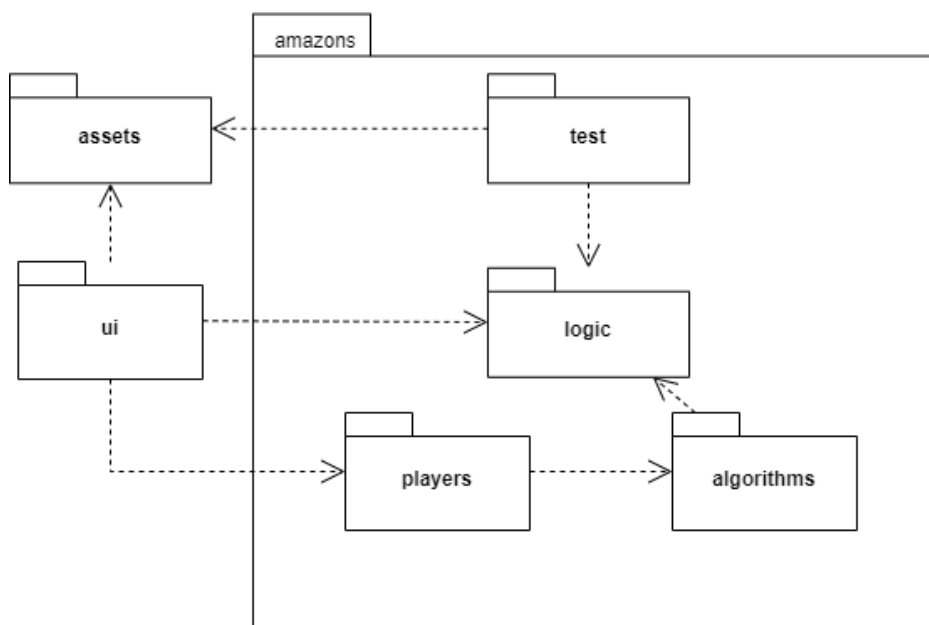


Ilustración 13 Diagrama de paquetes general

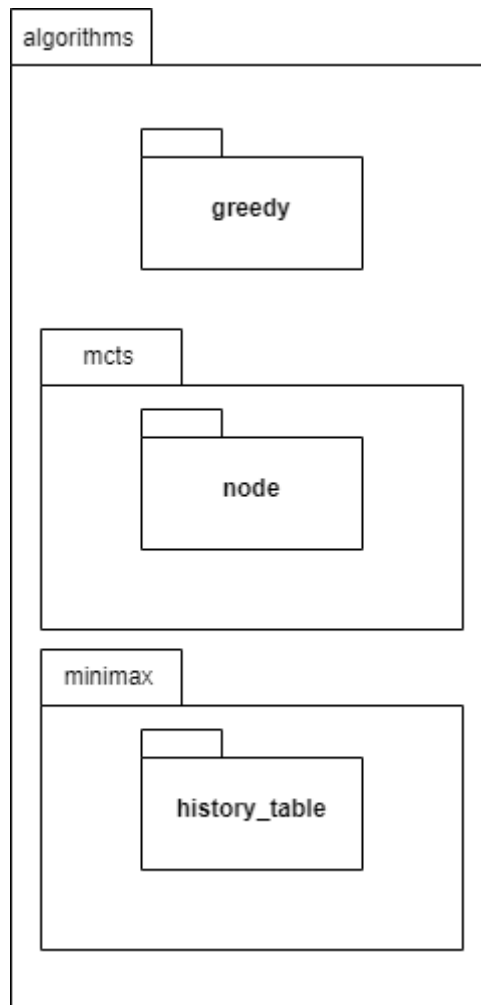


Ilustración 14 Diagrama de paquetes algoritmos

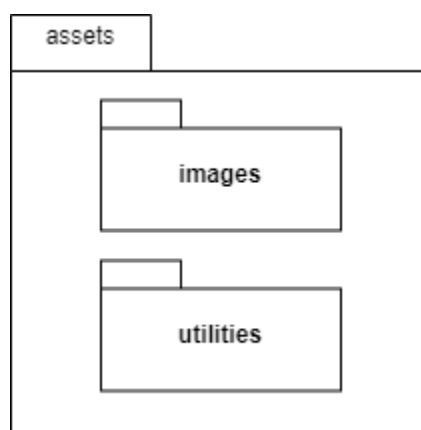


Ilustración 15 Diagrama de paquetes activos

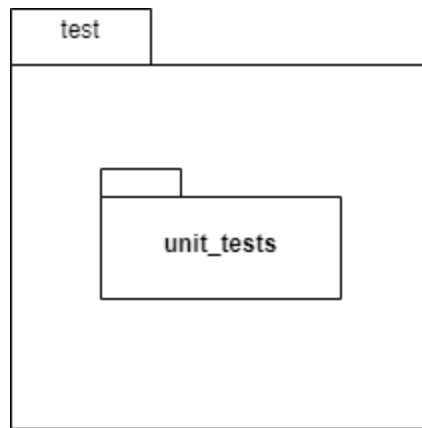


Ilustración 16 Diagrama de paquetes pruebas

Ahora se describirá cada uno de los paquetes mostrados en los diferentes diagramas para facilitar su comprensión.

6.1.1.1 ui

Este paquete contiene toda la implementación necesaria para la interfaz gráfica de usuario. Su única responsabilidad es mostrar la interfaz y permitir al usuario interactuar con ella.

6.1.1.2 amazons

Este paquete contiene todo lo relacionado con el sistema del juego de las amazonas. Tanto la lógica, los algoritmos, jugadores, test y utilidades diversas que en conjunto forman el sistema de juego de las Amazonas. Este paquete contiene otros paquetes que serán descritos a continuación.

6.1.1.2.1 logic

Este paquete contiene todo el código que implementa la lógica del sistema.

6.1.1.2.2 players

Este paquete contiene las implementaciones del jugador humano y del jugador artificial que sirven de mediador entre los algoritmos y la interfaz.

6.1.1.2.3 algorithms

Este paquete contiene las implementaciones de los diferentes algoritmos de inteligencia artificial para el Juego de las Amazonas. Para las implementaciones de algoritmos existen más paquetes dentro de este.

6.1.1.2.3.1 greedy

Este paquete contiene las implementaciones de los algoritmos voraces.

6.1.1.2.3.2 mcts

Este paquete contiene las implementaciones de los algoritmos de árboles de búsqueda de Monte Carlo.

6.1.1.2.3.2.1 node

Este paquete contiene los nodos implementados para crear el árbol de búsqueda.

6.1.1.2.3.3 minimax

Este paquete contiene las implementaciones de los algoritmos de minimax.

6.1.1.2.3.3.1 history_table

Este paquete contiene las tablas históricas utilizadas por los algoritmos de minimax.

6.1.1.2.4 test

Este paquete contiene los tests que se realizarán al sistema.

6.1.1.2.4.1 unit_tests

Este paquete contiene los tests unitarios del sistema.

6.1.1.3 assets

Este paquete contiene varias utilidades que se usan a través de todo el sistema.

6.1.1.3.1 images

Este paquete contiene las imágenes que serán utilizadas en la interfaz gráfica para mostrar las fichas y las casillas bloqueadas.

6.1.1.3.2 utilities

Este paquete contiene utilidades que pueden ser utilizadas en varias partes de la aplicación. La principal utilidad es la clase MatchRecorder (ver 6.2) que se encarga de llevar un registro de las partidas durante las pruebas.

6.1.2 Diagramas de Componentes y de Despliegue

Por la naturaleza del proyecto, no existen muchos componentes ni es necesario un despliegue complejo. El código será ejecutado en la máquina del usuario o en una máquina virtual por parte del desarrollador durante el entrenamiento. Aun así, a continuación, se presenta el diagrama de componentes y de despliegue del sistema.

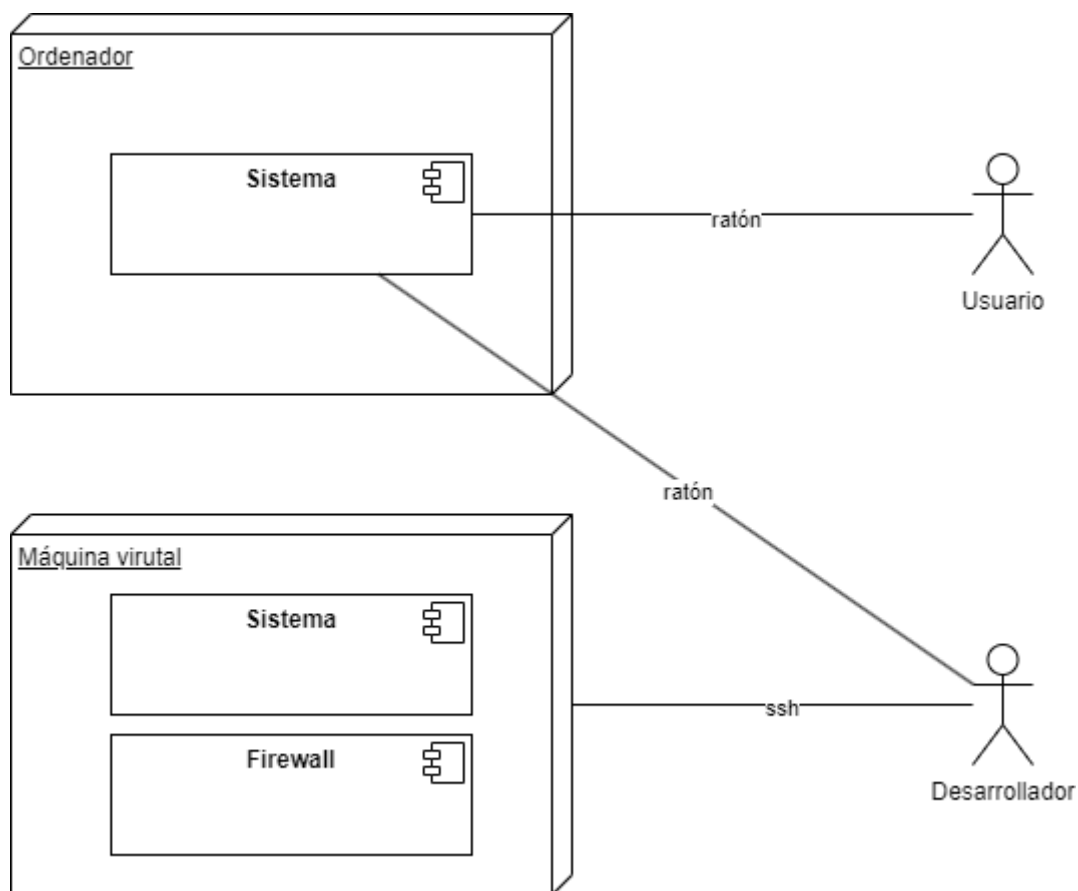


Ilustración 17 Diagrama de componentes y despliegue

6.2 Diseño de Clases

En este apartado se presentará el diagrama de clases principal del sistema. Con este se pretende representar la estructura de clases y paquetes que seguirá el sistema. Este diagrama es una evolución del diagrama presentado en el apartado 5.4.1, por lo que es esperable que existan algunos cambios.

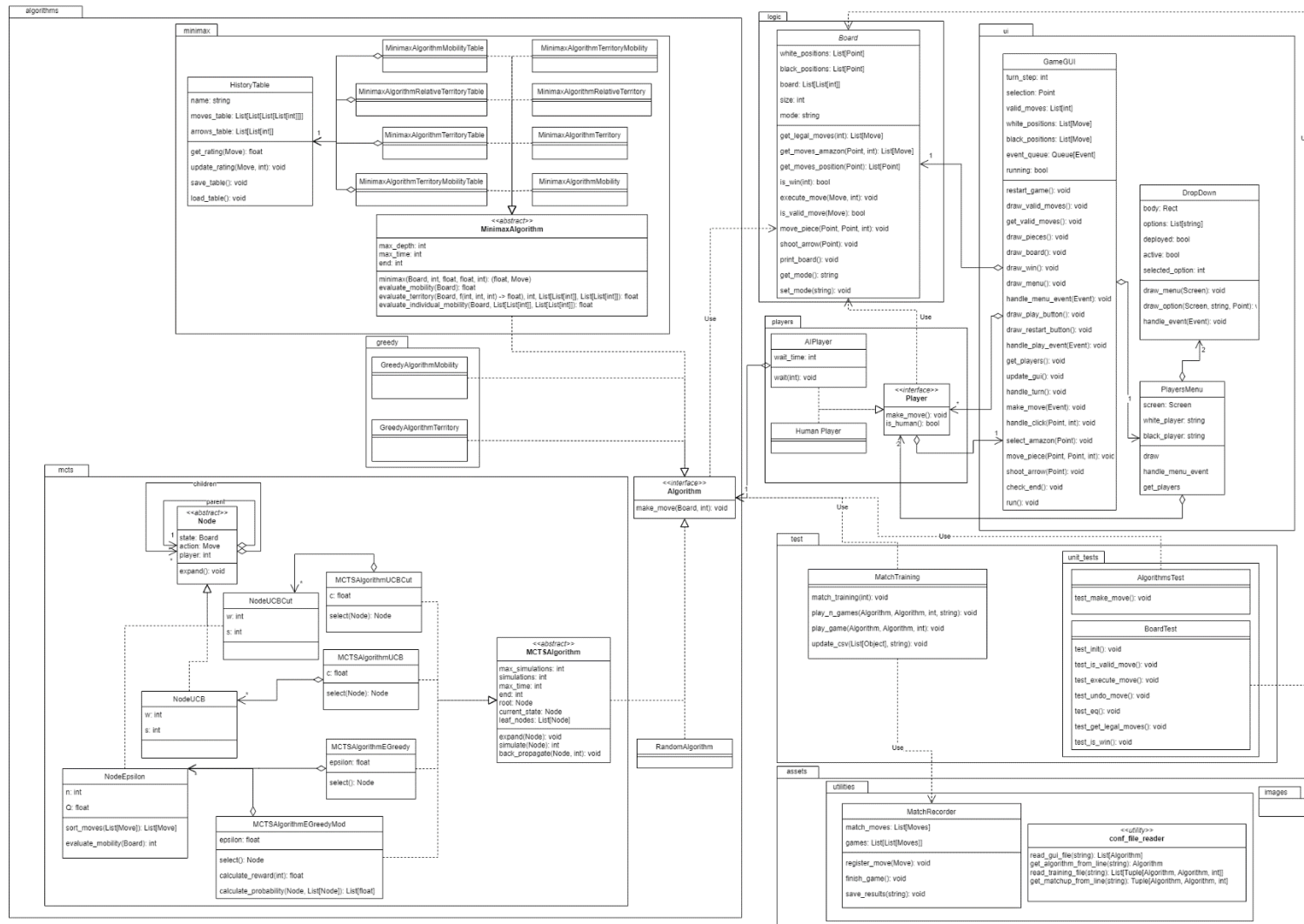


Ilustración 18 Diagrama de clases

6.2.1 Consideraciones sobre el Diseño

Cabe mencionar que, pese a que debe ser un reflejo fiel de la implementación, en el diagrama de clases se omiten algunos métodos, funciones y clases que no se han considerado relevantes para la comprensión del sistema. Estos incluyen sobre todo métodos o funciones auxiliares que no forman parte de las funciones principales de las clases y también clases que han sido mejoradas o descartadas, pero que no han sido eliminadas.

Para entender de una forma correcta el diseño se presentan algunas consideraciones importantes. El tipo Move, no es una clase implementada en el sistema. Se utiliza la representación del tipo Move para indicar que en esa posición en la llamada de un método se debe introducir un movimiento. Sin embargo, un movimiento es representado en la implementación como una tupla de 3 elementos que representan respectivamente: la casilla de la que parte la amazona, la casilla a la que se mueve y la casilla a la que dispara. Cada casilla se representa a su vez como una tupla que indica sus coordenadas. De la misma forma, el tipo Point no existe si no que es una tupla con coordenadas. Otra consideración importante es que en UML estándar no hay forma de modelar una función de orden superior, por eso se ha utilizado el siguiente formato: `f(int, int, int) -> float` es una función 'f' que toma como parámetros 3 enteros y devuelve un decimal.

6.3 Diagramas de Secuencia

Los diagramas de secuencia permiten entender las interacciones entre distintos objetos dentro del sistema a lo largo del tiempo. A continuación, se presentan los diagramas de secuencia para los casos de uso realizar movimiento y entrenar inteligencias artificiales.

6.3.1 Realizar Movimiento

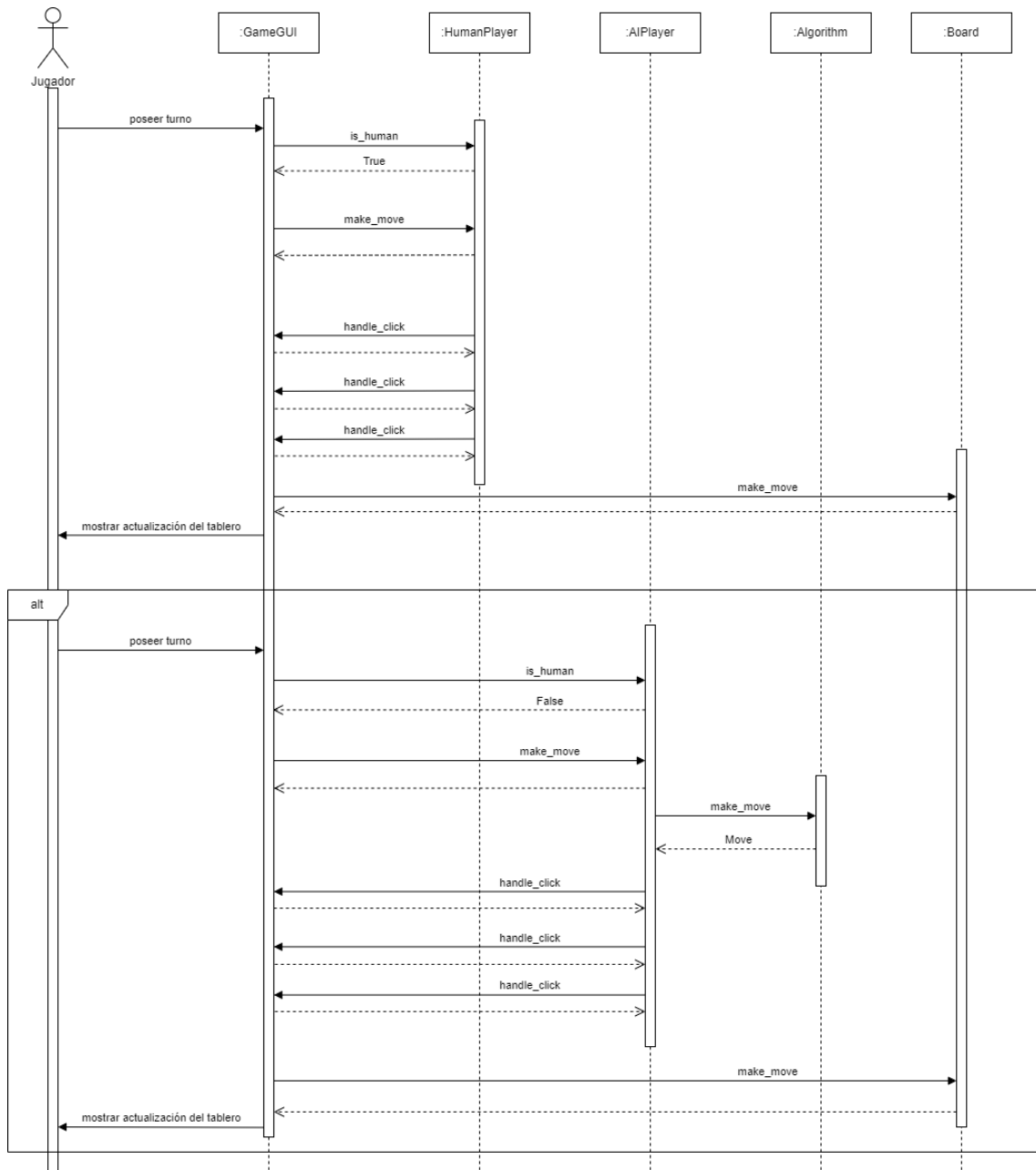


Ilustración 19 Diagrama de secuencia, realizar movimiento

6.3.2 Entrenar Inteligencias Artificiales

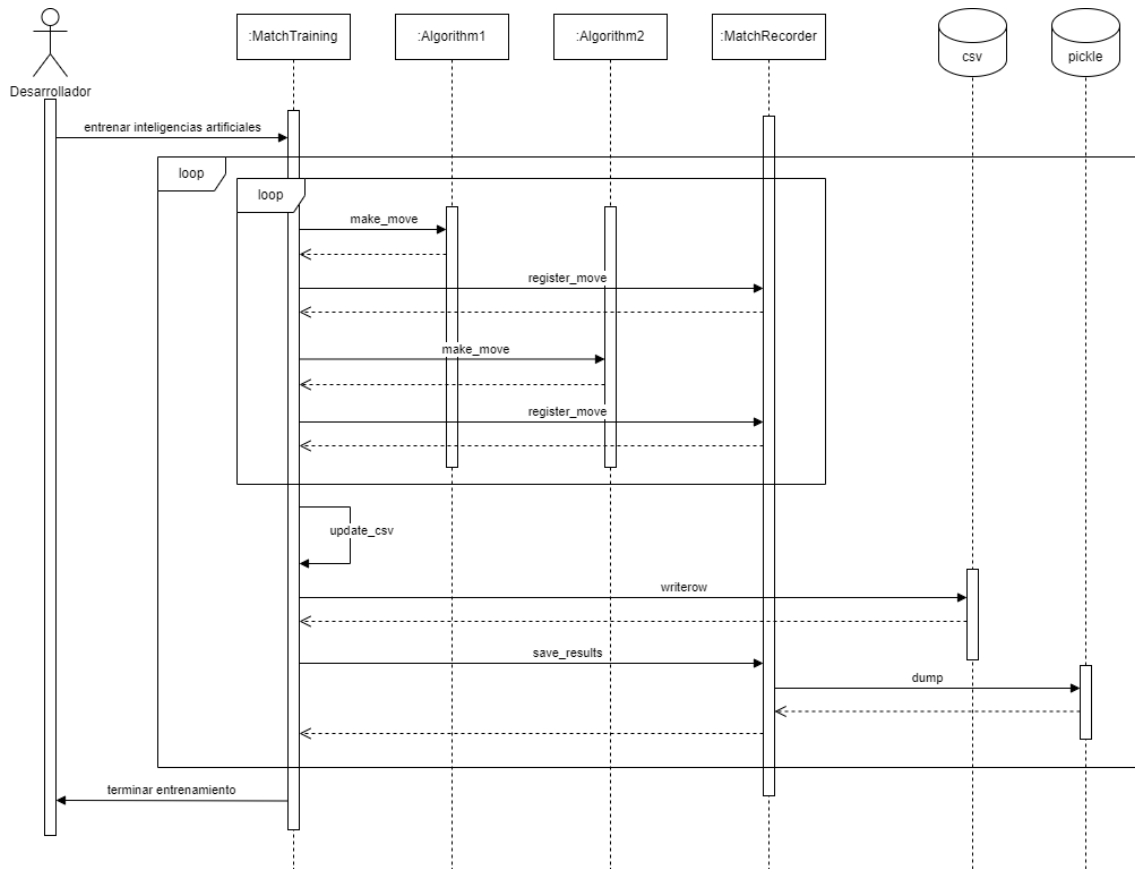


Ilustración 20 Diagrama de secuencia, entrenar inteligencias artificiales

6.4 Diseño de Persistencia

Debido a la naturaleza del proyecto, este no requiere de una base de datos para guardar información. Sin embargo, el proyecto sí requiere de persistencia en 2 partes. Primero, en la tabla histórica que acompaña a algunas implementaciones del algoritmo Minimax. Segundo, a la hora de registrar las partidas jugadas durante las pruebas.

6.4.1 Descripción del Sistema de Persistencia

Usado

Para cumplir con las necesidades de persistencia anteriormente descritas se ha utilizado el módulo Python pickle [45]. Pickle es un módulo de serialización y deserialización de objetos para Python. Este es el módulo de Python más utilizado para serializar objetos. Es capaz de convertir cualquier objeto en un flujo de bytes que puede ser almacenado en un equipo y posteriormente recuperado. Es importante mencionar que los objetos serializados por el módulo pickle solo pueden ser recuperados por programas basados en Python.

6.4.2 Integración del Sistema de Persistencia

Como ya se ha comentado anteriormente, se utiliza pickle para implementar la tabla histórica utilizada por algunas implementaciones del algoritmo Minimax y también para registrar movimientos durante la fase de pruebas.

Para la implementación de la tabla histórica se emplea la clase HistoryTable (ver 6.2). Esta clase es la encargada de almacenar y recuperar las evaluaciones de posiciones. Esto lo hace empleando 2 parámetros [46]. El primero almacena los movimientos de amazonas y el segundo los disparos (ver 3.1).

El primer parámetro es una matriz de 4 dimensiones (implementada mediante listas de Python). Con estas 4 dimensiones almacena la coordenada X de la casilla de partida, la coordenada Y de la casilla de partida, la coordenada X de la casilla de destino y la coordenada Y de la casilla de destino. Como ejemplo, la evaluación del movimiento (0, 0) -> (1, 1) se almacenaría de forma similar a esta: `mov[0][0][1][1] = eval`.

El segundo parámetro es una matriz de 2 dimensiones (implementada de la misma forma que la matriz de movimientos). En sus 2 dimensiones almacena la coordenada X del disparo y la coordenada Y del disparo. Como ejemplo, la evaluación del disparo (3, 1) se almacenaría de forma similar a esta: `disp[3][1] = eval`.

Es finalmente la clase HistoryTable (con todos sus atributos dentro) la que más tarde se serializa y se deserializa.

Para el registro de los movimientos durante la comparación y entrenamiento de los algoritmos durante la fase de pruebas se utiliza la clase MatchRecorder. Esta clase se encarga de guardar los movimientos ordenados de los algoritmos. Para ello esta clase posee dos listas. La primera contiene los movimientos que se juegan durante una partida (los movimientos se almacenan como una tripleta, ver 6.2.1). La segunda es una lista que contiene las listas de movimientos mencionadas. De esta forma durante las partidas se van almacenando los movimientos en la primera lista y al finalizar una partida se almacena la lista en la segunda. Cuando termina una tanda de partidas entre 2 algoritmos se almacena mediante pickle únicamente la segunda lista.

6.5 Diseño de la Interfaz

En este apartado se mostrará la interfaz gráfica definitiva de la aplicación tal y como se ve durante la ejecución del sistema. Se mostrarán todas las partes de la interfaz gráfica y se describirá su utilidad y cómo interactúa el usuario con ellas.

En la Ilustración 21 se muestra la pantalla principal de la interfaz gráfica del sistema, identificando las partes principales con las que puede interactuar el usuario.

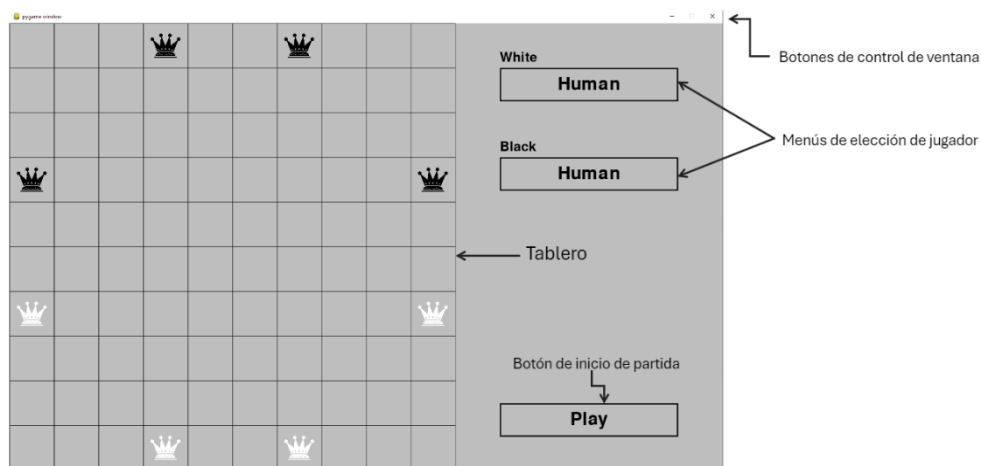


Ilustración 21 Interfaz gráfica del sistema por partes

Botones de control de ventana: estos botones son propios de las ventanas de cualquier sistema operativo. En este caso se pueden utilizar para terminar la ejecución del programa (botón de cerrar) o minimizar la ventana (botón de minimizar).

Menú de elección de jugadores: este menú está compuesto de 2 desplegables que permiten al usuario elegir los jugadores que participarán en una partida antes de que esta empiece o después de que una termine. Se puede ver su uso con cada menú desplegado en las ilustraciones Ilustración 22 e Ilustración 23.

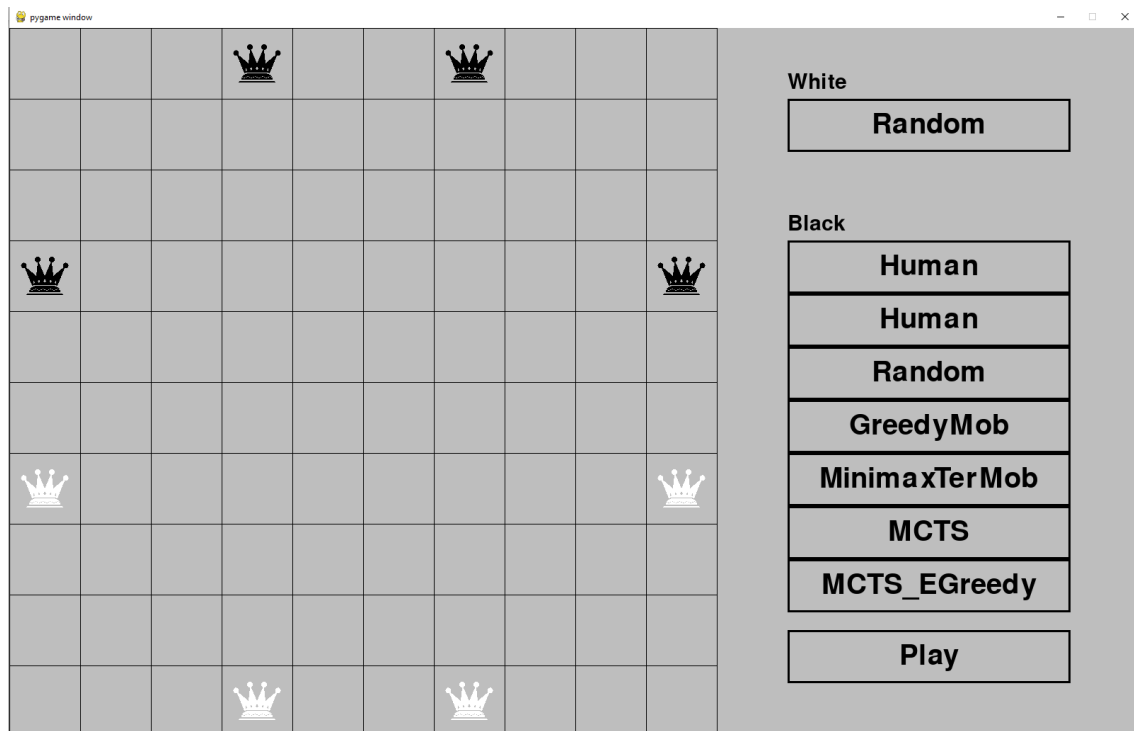


Ilustración 22 Interfaz gráfica, menú negras desplegado

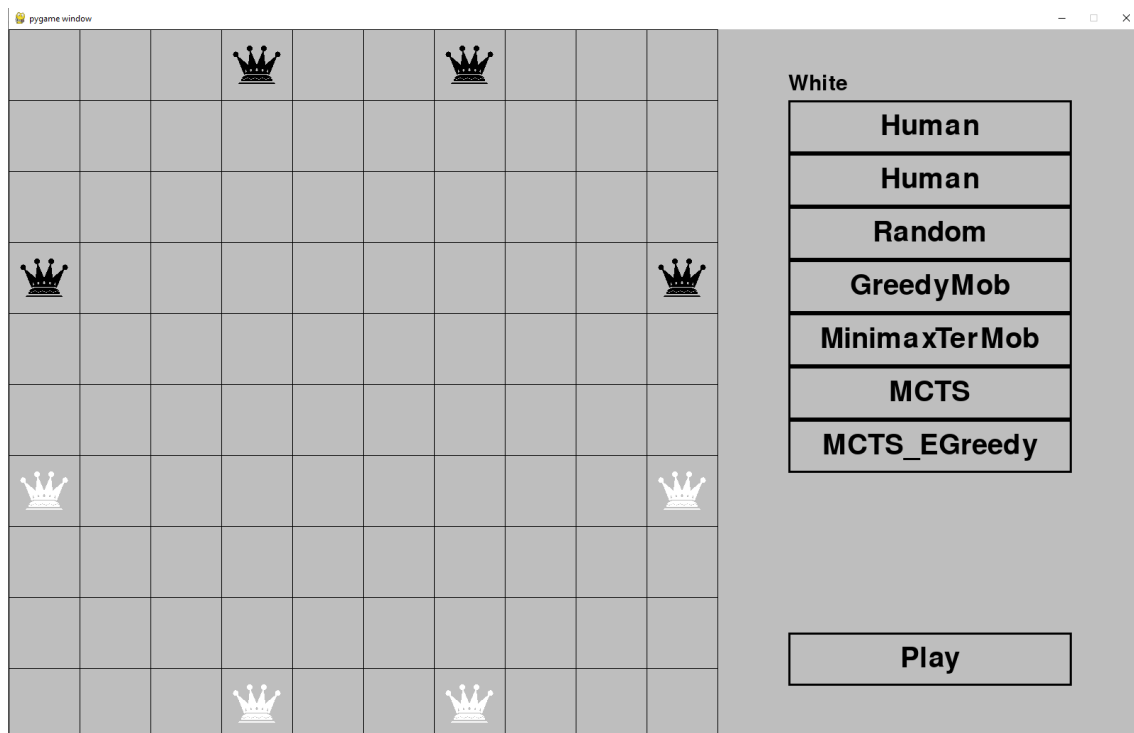


Ilustración 23 Interfaz gráfica, menú blancas desplegado

Botón de iniciar partida: este botón es utilizado por el usuario para dar comienzo a una partida. Una vez comience una partida este botón desaparecerá y solo volverá a aparecer cuando esta termine, para dar la oportunidad al usuario a comenzar una nueva partida. A continuación, se muestran estas 3 situaciones en las ilustraciones Ilustración 24, Ilustración 25 e Ilustración 26.

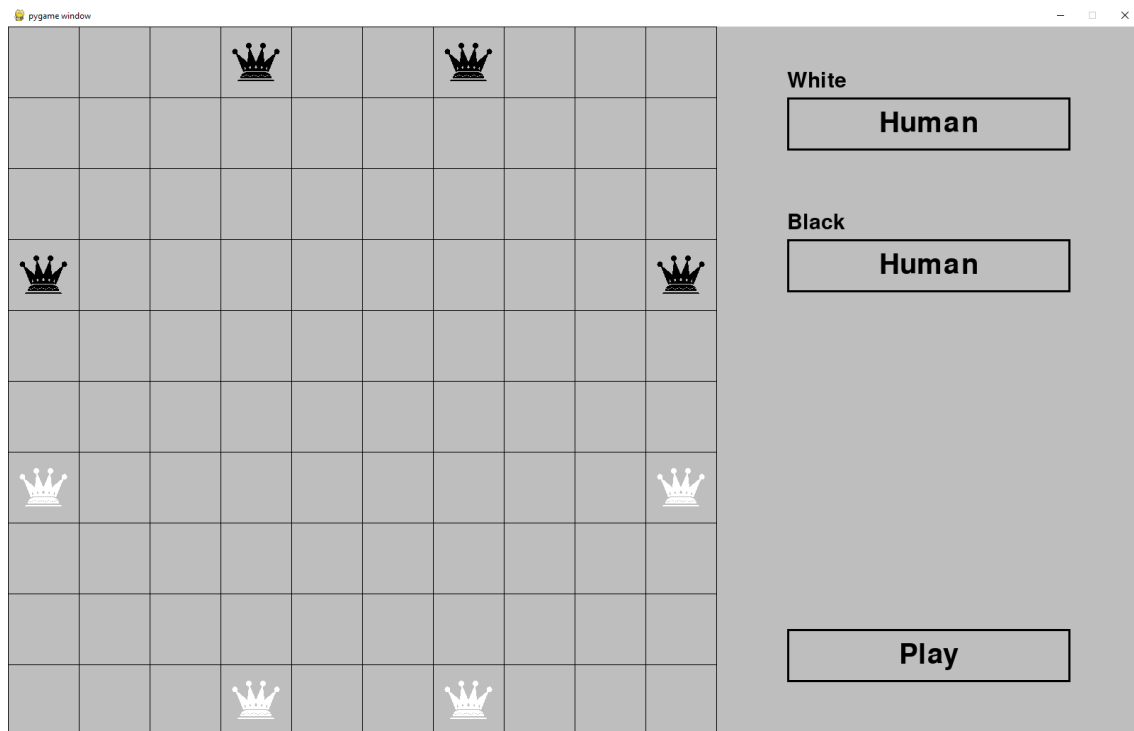


Ilustración 24 Interfaz gráfica, botón iniciar partida en estado inicial

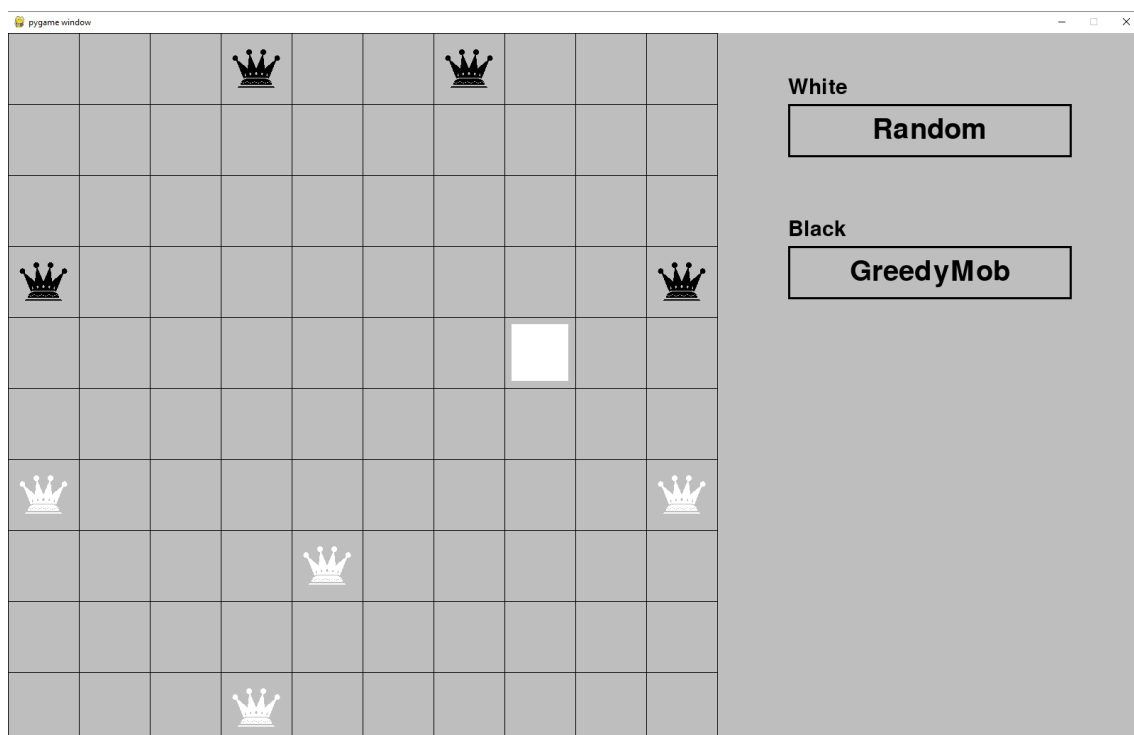


Ilustración 25 Interfaz gráfica, botón iniciar partida oculto

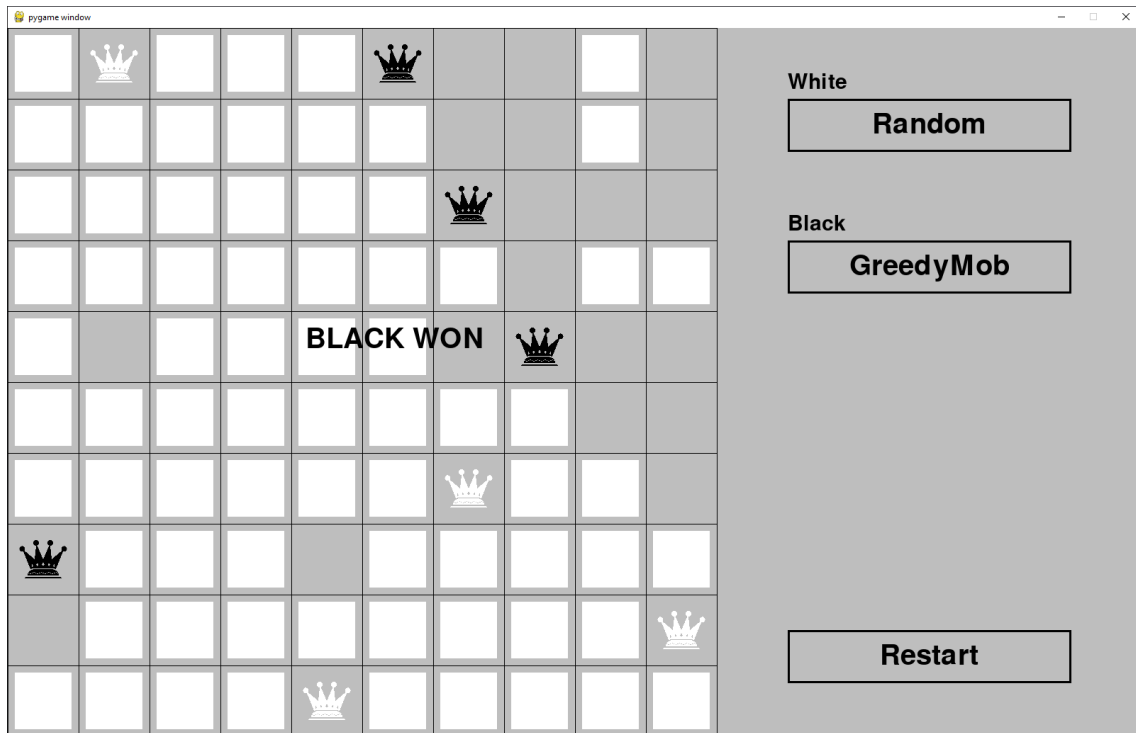


Ilustración 26 Interfaz gráfica, botón iniciar partida tras terminar partida

Tablero: esta parte de la pantalla se dedica al tablero del Juego de las Amazonas. En el caso de que el usuario seleccione al menos un jugador humano, el usuario podrá interactuar con el tablero para realizar movimientos. En las ilustraciones Ilustración 27, Ilustración 28 e Ilustración 29 se muestran las 3 partes de un movimiento del Juego de las Amazonas en la interfaz gráfica.

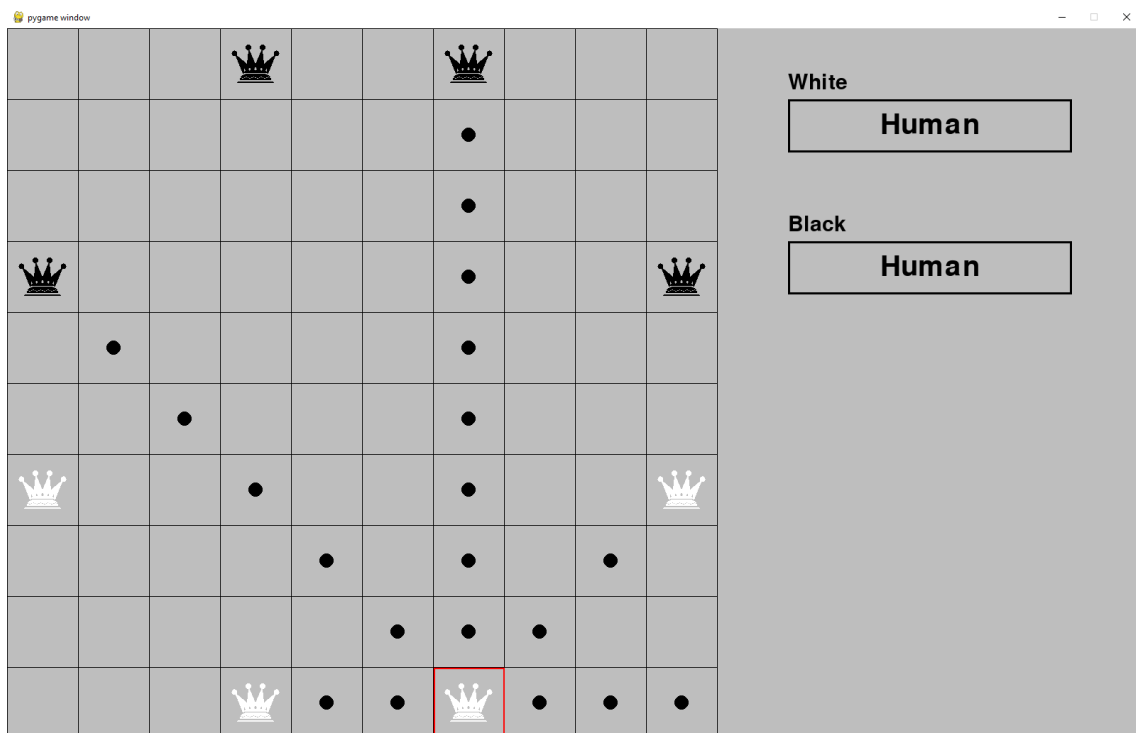


Ilustración 27 Interfaz gráfica, amazona seleccionada

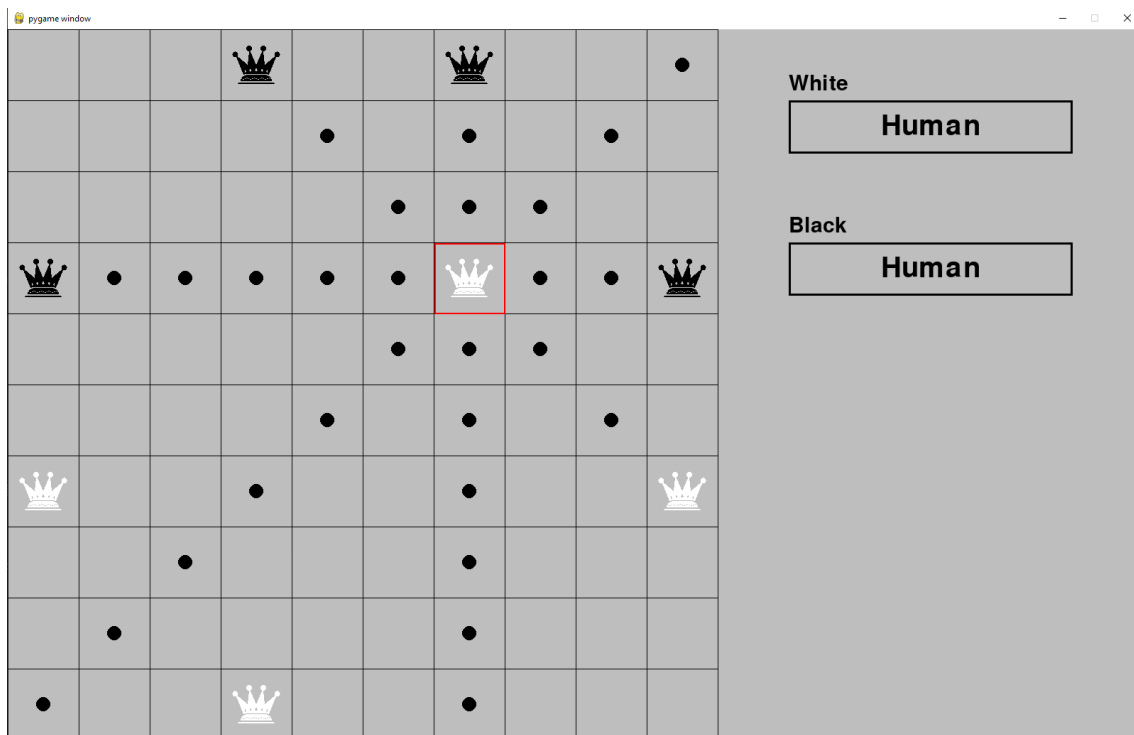


Ilustración 28 Interfaz gráfica, amazona movida

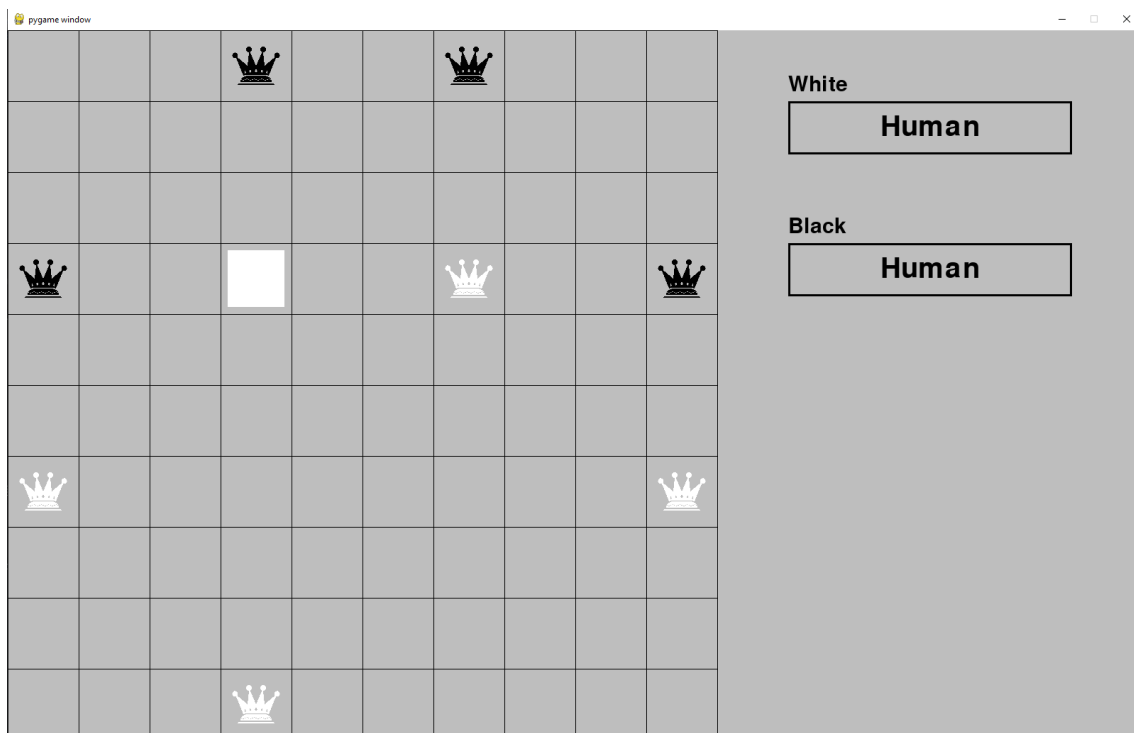


Ilustración 29 Intefaz gráfica, disparo realizado

6.6 Especificación Técnica del Plan de Pruebas

En este apartado se especificará el plan de pruebas que se llevará a cabo durante el desarrollo del sistema.

6.6.1 Pruebas Unitarias

En este apartado se describirán las diferentes pruebas unitarias realizadas al sistema. Las pruebas unitarias han sido realizadas sobre los 2 subsistemas más importantes, el subsistema de lógica y el de algoritmos (ver 5.3). Para más información acerca de este tipo de pruebas se puede consultar el apartado 5.8.2 y para información acerca del módulo utilizado para implementarlas, 7.2.1.11.

6.6.1.1 Subsistema Lógica

Sobre el subsistema de lógica se fundamenta el resto del sistema, tanto la interfaz gráfica como los jugadores y algoritmos. Es por esto por lo que es importante comprobar que funcione correctamente.

Para afrontar las pruebas unitarias del paquete de lógica, se ha probado la clase Board. Para realizar las pruebas unitarias a dicha clase se han dividido en varios subproblemas que se han probado por separado. Estos subproblemas corresponden a los métodos más importantes de la clase. Los métodos serán enumerados y descritos a continuación y se explicará para cada uno de ellos la estrategia de pruebas a seguir.

6.6.1.1.1 `__init__`

Este método es el constructor de la clase, si bien es privado, se le llama cada vez que se crea la clase. Es importante comprobar que se crea adecuadamente la clase desde cero y también pasándole un tablero para que lo copie.

Para este subproblema se utilizará la técnica de clases de equivalencia con una estrategia combinada. Esto es debido a que solo existen 3 clases de equivalencia por lo que es sencillo realizar todas las combinaciones posibles. Se comprobará que se crea correctamente el tablero y se inicializan correctamente las posiciones.

Clases de equivalencia	Valores
Tablero creado o copiado	Creado
	Copiado
Posiciones modificadas	Sí
	No
Modo modificado	Sí
	No

Tabla 35 Clases de equivalencia `__init__`

Entradas		Salidas
----------	--	---------

Creado o Copiado	Posiciones Modificadas	Modo modificado	
Creado	No	No	Correcto
Copiado	No	No	Correcto
Copiado	No	Sí	Correcto
Copiado	Sí	No	Correcto
Copiado	Sí	Sí	Correcto

Tabla 36 Pruebas `__init__`

6.6.1.1.2 `is_valid_move`

Este subproblema será dividido en 4 partes diferentes. Por la naturaleza de un movimiento, al método se le pasa una tupla de 3 elementos (coordenadas de la amazona, coordenadas de la casilla donde se mueve y coordenadas de la casilla a disparar) y un entero que representa el jugador (1 o -1). Para cada uno de los componentes del movimiento se utilizarán las mismas clases de equivalencia y habrá otras distintas para el jugador.

Clases de equivalencia (movimiento)	Valores
Tipo del movimiento	Tipo correcto (tupla)
	Tipo incorrecto
Coordenadas	< 0
	> límite
	Casilla sin amazona
	Casilla con amazona rival
	Casilla con amazona propia
	Movimiento imposible

Tabla 37 Clases de equivalencia `is_valid_move` (movimiento)

Clases de equivalencia (jugador)	Valores
Tipo del jugador	Tipo correcto (entero)
	Tipo incorrecto
Jugador	1
	-1
	otro

Tabla 38 Clases de equivalencia `is_valid_move` (jugador)

Para este subproblema se utilizará la técnica de clases de equivalencia con una estrategia minimizada. Esto es debido a la gran cantidad de clases de equivalencia que hacen complicado probar todas las combinaciones y así se podrán probar las clases negativas sin enmascarar errores. La Tabla 37 sirve para los 3 casos anteriormente mencionados y la Tabla 38 para el jugador. Para las coordenadas se utilizarán valores límite.

Entradas			Salidas	
Coordenadas			Jugador	
(0, 6)	(1, 7)	(2, 8)	1	Verdadero
(0, 3)	(0, 4)	(0, 5)	-1	Verdadero
"str"	"str"	"str"	1	Falso
(0, 6)	(-1, 6)	(-2, 5)	1	Falso

(0, 6)	(1, 7)	(1, 10)	1	Falso
(0, 0)	(1, 7)	(2, 8)	1	Falso
(0, 6)	(1, 7)	(2, 8)	-1	Falso
(0, 6)	(1, 8)	(0, 6)	1	Falso
(0, 6)	(1, 7)	(2, 8)	"str"	Falso
(0, 6)	(1, 7)	(2, 8)	2	Falso

Tabla 39 Pruebas is_valid_move

6.6.1.1.3 execute_move

Se debe comprobar que los movimientos se realizar correctamente y que solo se ejecutan si son movimientos válidos.

Para este método se utilizará el mismo método que en 6.6.1.1.2 ya que lo único que cambia es que tras la ejecución de execute_move, en vez de devolver verdadero o falso, lanza un error.

6.6.1.1.4 undo_move

De forma similar al método execute_move, se debe comprobar que funciona correctamente.

Este método será probado del mismo modo que el método execute_move.

6.6.1.1.5 _eq_

Este método es privado, pero se le llama para comprobar si dos tableros son iguales, por lo que se debe comprobar que funcione según lo esperado.

Para este método se aplican clases de equivalencia y combinación múltiple por el bajo número de clases de equivalencia.

Clases de equivalencia	Valores
Tablero	Igual
	No igual
Posiciones de las piezas	Igual
	No igual

Tabla 40 Clases de equivalencia _eq_

Entradas		Salidas
Tablero	Posiciones de las piezas	
No igual	No igual	Falso
No igual	Igual	Falso
Igual	No igual	Falso
Igual	Igual	Verdadero

Tabla 41 Pruebas _eq_

6.6.1.1.6 `get_legal_moves`

Este método sirve para obtener los movimientos que tiene un jugador en un estado de partida concreto. Será probado para comprobar que devuelve los movimientos adecuados.

Para este método se aplican clases de equivalencia y combinación múltiple al haber solo 1 clase de equivalencia.

Clases de equivalencia	Valores
Número de movimientos	0
	1
	> 1

Tabla 42 Clases de equivalencia `get_legal_moves`

Entradas	Salidas
Número de movimientos	
0	Lista vacía
1	Lista con 1 elemento
2	Lista con 2 elementos

Tabla 43 Pruebas `get_legal_moves`

6.6.1.1.7 `is_win`

Este método sirve para comprobar si un jugador ha ganado la partida. Se comprobará que funciona correctamente.

Para este método se aplican clases de equivalencia y combinaciones parciales.

Clases de equivalencia	Valores
Victoria blancas	Sí
	No
Victoria negras	Sí
	No
Jugador	1
	-1
	Otro
Tipo jugador	Entero
	Otro

Tabla 44 Clases de equivalencia `is_win`

Entradas			Salidas
Victoria blancas	Victoria negras	Jugador	
No	No	"str"	Error
No	No	0	Error
No	No	1	Falso
No	No	-1	Falso
Sí	No	1	Verdadero
Sí	No	-1	Falso
No	Sí	1	Falso
No	Sí	-1	Verdadero

Tabla 45 Pruebas is_win

6.6.1.2 Subsistema Algoritmos

Aunque los algoritmos sean el foco central del proyecto, debido a su funcionamiento no hay muchas formas de probarlos. Cada uno funciona de una manera distinta. De todas formas, se realizarán pruebas unitarias sencillas para comprobar que realizan movimientos según lo esperado.

Para cada algoritmo se comprobará que devuelve un movimiento legal en un estado inicial de la partida, en un estado donde solo existe un movimiento posible y un estado en el que no existe movimiento posible. Se comprobará que el movimiento que devuelven sea válido en todos los casos para blancas y para negras.

Clases de equivalencia	Valores
Número de movimientos	0
	1
	> 1

Tabla 46 Casos de prueba Algorithms

Entradas		Salidas
Número de movimientos	Jugador	
0	Blancas	Ningún movimiento
0	Negras	Ningún movimiento
1	Blancas	Único movimiento legal
1	Negras	Único movimiento legal
2176	Blancas	Un movimiento legal
	Negras	Un movimiento legal

Tabla 47 Pruebas Algorithms

6.6.2 Pruebas de Integración y del Sistema

Las pruebas de integración se realizarán de forma manual y consistirán en probar la mayor parte del sistema en su conjunto por medio de la interfaz gráfica. Para ello se ha construido un procedimiento y una checklist (ver 15.1). El procedimiento se puede ver en la Tabla 48 y la checklist se puede ver en la Tabla 49. Nótese que para cada punto del procedimiento existe al menos una comprobación en la checklist.

ID	Descripción
1	Iniciar el sistema
2a	Seleccionar jugador aleatorio para las blancas
2b	Seleccionar jugador humano para las negras
3	Iniciar partida
4	Jugar la partida
5	Cambiar jugador de piezas negras por jugador MCTS E-Greedy
6	Reiniciar partida
7	Esperar a que acabe la partida
8	Cerrar sistema

Tabla 48 Procedimiento pruebas de integración del sistema

ID	Descripción
1	Comprobar que se muestra la interfaz gráfica
1'	Comprobar que el tablero está en el estado inicial
1''	Comprobar que se muestra el menú de elección de jugadores
1'''	Comprobar que se muestra el botón de iniciar partida
2a	Comprobar que al pulsar el menú este se despliega
2a'	Comprobar que al pulsar el jugador el menú se repliega
2b	Comprobar que al pulsar el menú este se despliega
2b'	Comprobar que al pulsar el jugador el menú se repliega
3	Comprobar que al pulsar el botón este se oculta
3'	Comprobar que ya no se pueden cambiar los jugadores
4	Comprobar que ambos jugadores realizan correctamente los movimientos
4'	Comprobar que cuando la partida termina se muestra el ganador
4''	Comprobar que se muestra el botón de reiniciar
5	Comprobar que al pulsar el menú este se despliega
5'	Comprobar que al seleccionar el jugador el menú se repliega
6	Comprobar que el botón se oculta
6'	Comprobar que el tablero vuelve al estado inicial
6''	Comprobar que ya no se pueden cambiar los jugadores
7	Comprobar que ambos jugadores realizan correctamente los movimientos
7'	Comprobar que cuando la partida termina se muestra el ganador
7''	Comprobar que se muestra el botón de reiniciar
8	Comprobar que se cierra el sistema

Tabla 49 Checklist pruebas de integración del sistema

6.6.3 Pruebas de Usabilidad y Accesibilidad

En este apartado se describirá el plan de pruebas para las pruebas de usabilidad y accesibilidad. Este tipo de pruebas ya han sido descritas en el apartado 5.8.5.

Para este tipo de pruebas se usarán una serie de cuestionarios enfocados a obtener las opiniones de diversos usuarios reales. Primero se hará una breve presentación del tipo de cuestionarios que serán realizados y posteriormente se presentarán dichos cuestionarios que más tarde serán cumplimentados por los usuarios y el responsable de las pruebas.

6.6.3.1 Diseño de los Cuestionarios

El cuestionario para los usuarios se dividirá en 4 partes diferentes. La primera parte del cuestionario son una serie de preguntas de carácter general para comprender el nivel informático del usuario, así como otros datos que puedan ser de interés. La segunda parte será una lista de actividades que deberán completar los usuarios bajo la supervisión del responsable de las pruebas. La tercera parte serán preguntas sobre la experiencia del usuario con el sistema. Por último, en la cuarta parte se dará la oportunidad al usuario de indicar cualquier observación u aspecto sobre el sistema que le parezca adecuado.

El cuestionario para el responsable de las pruebas será un cuestionario que permita al encargado de supervisar las pruebas anotar información durante la realización de las pruebas.

6.6.3.1.1 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
<ol style="list-style-type: none"> 1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
<ol style="list-style-type: none"> 1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
<ol style="list-style-type: none"> 1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
<ol style="list-style-type: none"> 1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias

Tabla 50 Preguntas de carácter general

6.6.3.2 Actividades guiadas

Durante las actividades guiadas es importante que, aunque el responsable esté presente, intente ayudar lo menos posible y solo en casos de necesidad (por ejemplo, si el usuario no sabe continuar) al usuario. De esta forma se obtienen resultados más realistas y fieles sobre la experiencia del usuario.

Las actividades guiadas que serán realizadas por los usuarios se indican a continuación.

- Elegir un jugador para las piezas blancas y uno distinto para las piezas negras.
- Iniciar una partida.
- Jugar una partida contra un algoritmo.
- Reiniciar una partida.
- Cerrar el sistema.

6.6.3.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?				

¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz		Si	No	A veces
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				

Tabla 51 Preguntas cortas sobre la aplicación

6.6.3.4 Observaciones

Observaciones
Cualquier comentario del usuario

Tabla 52 Observaciones

6.6.3.5 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	Notas
...	...

Tabla 53 Cuestionario para el responsable de las pruebas

6.6.3.6 Pruebas de Accesibilidad

Para las pruebas de accesibilidad se completará un cuestionario sencillo que se presenta en la Tabla 54. Este cuestionario está basado en algunas de las guías presentadas en [47].

Preguntas:
¿La aplicación contiene mnemónicos como alternativa a todas las operaciones con ratón?

¿Se proveen instrucciones en el manual de usuario?
¿Es sencillo el uso de la aplicación usando la documentación?
¿Las pestañas están ordenadas de forma lógica garantizando buena navegabilidad?
¿Hay atajos para acceder a los menús?
¿Soporta todos los sistemas operativos?
¿Son apropiados para todos los usuarios los colores de la aplicación?
¿Están escritas correctamente todas las etiquetas?
¿Son las imágenes e iconos fáciles de entender?
¿Tienen sonido las alertas de la aplicación?
¿Se puede cambiar la fuente y su tamaño?
¿Se usa el color como único medio para transmitir información?

Tabla 54 Cuestionario accesibilidad

6.6.4 Pruebas de Rendimiento

Para las pruebas de rendimiento se realizarán 2 tipos de pruebas. Primero se realizará un profiling (ver 15.1) de la ejecución de un algoritmo para obtener el primer movimiento de una partida utilizando el módulo cProfiling (ver 7.2.1.12). Después, durante la fase de entrenamiento de los algoritmos, se obtendrán los tiempos de ejecución de partidas simuladas y se presentarán algunos de ellos para comprobar que el tiempo de ejecución no sea excesivo.

Capítulo 7. Implementación del Sistema

En este capítulo se explica en detalle los aspectos fundamentales de la fase de implementación del sistema. En este capítulo se enumeran los distintos estándares seguidos, los lenguajes de programación, librerías, herramientas usadas y los detalles de la implementación del sistema, incluyendo los retos encontrados. Toda esta información se divide en 4 apartados, 7.1, 7.2, 7.3 y 7.4. En el primero se incluirá todo lo relacionado con los estándares y normas utilizados, el segundo en módulos y librerías utilizadas, el tercero se centrará en las herramientas utilizadas en el proyecto, el cuarto en los problemas y retos encontrados durante el desarrollo.

7.1 Estándares y Normas Seguidos

Para la programación en Python principalmente se ha seguido la guía de estilo para código Python oficial [48]. El entorno de desarrollo utilizado, PyCharm (ver 7.3.1), facilita al desarrollador seguir dicha guía.

Para los diagramas presentados en este documento se ha utilizado el lenguaje UML [49]. Este es el lenguaje de modelado de software más extendido y utilizado en la actualidad. En aquellos puntos en los que no se haya utilizado este lenguaje se ha indicado apropiadamente.

7.2 Lenguajes de Programación

El lenguaje de programación principal utilizado en el proyecto ha sido Python. Hay 2 motivos principales por los cuales se ha elegido este lenguaje de programación y no otros. Para entender algunas de las ventajas y desventajas que proporciona el lenguaje Python se recomienda consultar 2.3.1.2.3. Los motivos son los siguientes.

Uso común en el desarrollo de IA: el lenguaje Python ha sido ampliamente usado para resolver o al menos para presentar soluciones a problemas relacionados con IA. Además, cuenta con un gran soporte en el ámbito del desarrollo de IAs, por ejemplo, con librerías o proyectos de IA desarrollados en Python.

Sencillez del lenguaje: debido al principal objetivo del proyecto (consultar Objetivos del Proyecto), es importante que la implementación de los algoritmos sea fácil de comprender y de modificar. El lenguaje Python permite ilustrar las características principales de cada algoritmo de forma clara y entendible por gente con menos experiencia en el desarrollo de software.

La versión de Python que se ha utilizado es la 3.11.9.

7.2.1 Módulos y Librerías

A continuación, se enumerarán y describirán brevemente los módulos y librerías utilizados para el desarrollo.

7.2.1.1 *Math*

Math [50] es uno de los módulos básicos de Python utilizado para realizar operaciones matemáticas. Se usa a lo largo de todo el sistema, tanto para los algoritmos como para la lógica o la interfaz.

7.2.1.2 *Time*

Time [51] es otro modulo básico de Python utilizado para medir y realizar operaciones con el tiempo. Se usa principalmente por los algoritmos para poder calcular el límite de tiempo establecido para realizar un movimiento, por el jugador artificial para simular el tiempo que tardaría un humano en realizar un movimiento y durante la simulación de partidas en las pruebas para registrar lo que duran los movimientos y partidas.

7.2.1.3 *ABC*

ABC (Abstract Base Classes) [52] es un módulo de Python que permite definir clases base abstractas. Este módulo se ha utilizado a la hora de crear las estructuras jerárquicas de clases. El módulo ABC permite que se puedan crear clases y métodos abstractos. De esta forma se simulan las interfaces y las clases abstractas típicas de la programación orientada a objetos.

7.2.1.4 *Pygame*

Pygame [53] es una biblioteca de Python orientada a permitir y facilitar el desarrollo de videojuegos. Es una de las librerías más utilizadas para el desarrollo de videojuegos sencillos en Python, en su mayoría 2D. En este proyecto se utiliza para implementar la interfaz gráfica que permite al usuario jugar al Juego de las Amazonas.

7.2.1.5 *Threading*

Threading [54] es un módulo Python que ofrece al desarrollador interfaces de alto nivel para programación multihilo. Si bien no soporta paralelismo real debido al GIL (ver 2.3.1.2.3.2), se utiliza en el proyecto para ejecutar la interfaz gráfica en un hilo distinto al hilo de ejecución de los jugadores y los algoritmos. De esta forma se evita paralizar la interfaz gráfica cada vez que un algoritmo se está ejecutando.

7.2.1.6 Multiprocessing

A diferencia del módulo threading (ver 7.2.1.5), el módulo multiprocessing [55] sí permite ejecución paralela real de código Python. El módulo multiprocessing evita el GIL (ver 2.3.1.2.3.2) debido a que usa subprocesos y no hilos. Este módulo se probó en algunas implementaciones del algoritmo Minimax que fueron posteriormente descartadas.

7.2.1.7 Random

El módulo random [56] es un módulo básico de Python que permite simular aleatoriedad. Esta es una utilidad importante y muy útil para varios de los algoritmos que se implementan en el sistema.

7.2.1.8 Os.Path

El módulo os.path [57] es un módulo que implementa algunas funciones útiles relacionadas con las rutas de ficheros del sistema operativo. En especial se utiliza en el sistema para comprobar si ya existe una tabla histórica para poder recuperar de ella las evaluaciones de movimientos o si por el contrario hay que crear una nueva o para comprobar si existen o no ficheros de configuración.

7.2.1.9 Sys

El módulo sys [58] es un módulo que permite acceder a variables y funciones relacionadas con el intérprete de Python. En el sistema se usa sobre todo para obtener los argumentos de ejecución del programa.

7.2.1.10 Pickle

El módulo pickle es utilizado para toda la persistencia del sistema. Para más información consultar 0.

7.2.1.11 Unittest

El módulo Python unittest [59] es el módulo básico para tests unitarios. Es utilizado para realizar todos los tests unitarios del sistema.

7.2.1.12 CProfile

El módulo cProfile [60] permite realizar profiling (ver 15.1) sobre una ejecución de código Python. Será utilizado para las pruebas de rendimiento del sistema.

7.2.1.13 *Sphinx*

El software Sphinx [61] permite generar documentación sobre el software de un sistema. Ha sido utilizado para generar toda la documentación de este sistema (ver 7.4.2).

7.2.1.14 *PyInstaller*

El módulo PyInstaller [62] permite generar un ejecutable a partir del código fuente. Ha sido utilizado para generar el ejecutable en este proyecto.

7.2.1.15 *Collections*

El módulo collections [63] es un módulo Python muy útil para tener acceso a implementaciones de colecciones típicas. Este módulo se utiliza en el sistema por su colección deque o cola doblemente terminada [64] (*double ended queue*) que se utiliza en el algoritmo Minimax a la hora de calcular el número mínimo de movimientos necesarios para mover una amazona de una casilla a otra.

7.2.1.16 *Copy*

El módulo Python copy [65] permite operaciones para copiar objetos de forma superficial o profunda. Este módulo se utiliza en varios algoritmos para obtener una copia base del tablero en un instante de tiempo determinado. De esta forma, las operaciones que se hagan en la copia no afectarán al tablero sobre el que se está jugando la partida.

7.2.1.17 *Csv*

El módulo Python csv [66] presenta funciones para leer y escribir ficheros CSV. Se utiliza en el sistema para guardar los resultados de las partidas simuladas entre algoritmos de forma que sea fácil de procesar posteriormente.

7.3 Herramientas y Programas Usados para el Desarrollo

A continuación, se describirán las herramientas utilizadas durante el proyecto. Tanto en la fase de análisis como en el diseño y en la implementación.

7.3.1 PyCharm Community Edition

El IDE (ver 147Glosario y Diccionario de Datos) utilizado ha sido PyCharm Community Edition en su versión 2023.3.2. Debido a la naturaleza del proyecto (centrado sobre todo en el estudio y comparación de distintos algoritmos), no se ha visto necesario utilizar una versión profesional. Muchas de las características que ofrece la Professional Edition [67] no son necesarias para el proyecto.

Se ha seleccionado PyCharm como entorno de desarrollo por varios motivos. PyCharm (y en general muchas de las herramientas desarrolladas por JetBrains) es considerado por la comunidad de desarrolladores como uno de los mejores entornos de desarrollo. Es un IDE que facilita mucho el desarrollo, generando sugerencias y corrigiendo al desarrollador durante la implementación. El sistema de gestión de archivos y paquetes es también de alta calidad. Además, al estar creado para el desarrollo en Python, no se requiere mucha configuración para poder utilizar el entorno de forma eficiente. En este aspecto es una filosofía distinta a los entornos de desarrollo pensados para varios lenguajes (por ejemplo, Visual Studio Code [68]).



Ilustración 30 Logo de PyCharm Community Edition

7.3.2 draw.io

Para una gran parte de los diagramas UML presentados en este documento se ha utilizado la herramienta online draw.io [69]. Esta herramienta es muy útil para la creación de diagramas de software y soporta la mayoría de los tipos de diagramas UML que existen.



Ilustración 31 Logo de draw.io

7.3.3 Visual paradigm

La herramienta Visual Paradigm, en concreto en su versión online [70], ha sido utilizada para crear alguno de los diagramas UML de este documento. Tiene muchas herramientas y soporta muchos tipos de gráficos diferentes, incluyendo de otros ámbitos distintos a la informática.



Ilustración 32 Logo de Visual Paradigm Online

7.3.4 Git

Git [71] es un sistema software de control de versiones especialmente diseñado para el control de versiones de un producto software. Una de las principales características es que se puede clonar y posteriormente editar de forma remota un repositorio en un sistema local y luego subir los cambios al repositorio original. Se ha vuelto el sistema de control de versiones más utilizado y extendido en el desarrollo de software.

Para el proyecto se ha utilizado como principal sistema de control de versiones para el código del sistema.



Ilustración 33 Logo de Git

7.3.5 GitHub

GitHub [72] es una plataforma pensada para alojar proyectos utilizando el sistema de control de versiones Git (ver 7.3.4). Se basa en la colaboración entre desarrolladores y en proyectos de código abierto. Actualmente es la plataforma proveedora de Git más utilizada en el mundo [73].

El código del proyecto se ha almacenado en esta plataforma y puede encontrarse a través del siguiente enlace: <https://github.com/gonzalo-rr/game-of-the-amazons-ai>.



Ilustración 34 Logo de GitHub

7.3.6 Mendeley Reference Manager

Mendeley Reference Manager [74] es un sistema de gestión de referencias bibliográficas. Permite almacenar referencias en una librería personal y compartirlas. Además, presenta extensiones para los navegadores más comunes, de esta forma se pueden guardar referencias navegando por internet, de forma fácil y rápida. También tiene una extensión para Word que permite gestionar e insertar las referencias directamente desde la aplicación de escritorio.

Mendeley ha sido utilizado para gestionar las referencias de este documento.

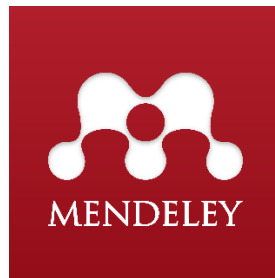


Ilustración 35 Logo de Mendeley

7.4 Creación del Sistema

En este apartado se tratarán aspectos importantes que se han tenido que considerar durante el desarrollo del sistema.

7.4.1 Problemas Encontrados

A continuación, se mencionarán y explicarán algunos de los problemas encontrados durante el desarrollo del sistema que han sido considerados importantes o interesantes.

7.4.1.1 *Copia del Tablero*

Uno de los primeros problemas encontrados durante el desarrollo de la versión más básica de Minimax fue pasar la información del tablero de un nodo a otro al crear el árbol de búsqueda. En un estado de partida concreto en el que se ejecute el algoritmo Minimax, dicho estado hace de raíz del árbol. A partir de ese nodo raíz, todos los posibles movimientos de un jugador que se puedan hacer en ese estado son hijos del nodo. Esto se puede repetir sucesivamente con cada hijo hasta una profundidad concreta o hasta que se llegue a completar el árbol llegando a los estados terminales.

Con cada posible movimiento se debe de pasar de un nodo a su nodo hijo el estado del padre tras realizar el movimiento. Para copiar un objeto en Python se puede utilizar la función `copy` del módulo con el mismo nombre (ver 7.2.1.16). Sin embargo, esta función realiza una copia superficial. Una copia superficial quiere decir que se hace un nuevo objeto y se le insertan las referencias de los objetos contenidos en el primero. El problema con la clase `Board` es que no copia los objetos de forma recursiva. De esta forma no se copian los elementos dentro de la lista bidimensional que representa el tablero. Otra opción sería crear un objeto nuevo de forma manual y copiar con la utilidad `copy` solo las columnas del tablero. Una tercera solución es utilizar la función `deepcopy` del mismo módulo. Esta función sí que realiza una copia profunda, no solo copia el objeto, sino que también copia de forma recursiva los objetos que este contiene.

El problema con las 2 soluciones comentadas en el párrafo anterior es el número de veces que se debe copiar el tablero. En el juego de las amazonas existen 2176 posibilidades para el primer movimiento de las blancas. Es por esto que el tiempo que se tarda en realizar las copias se acumula muy rápidamente y hace imposible que se pueda jugar una partida en un tiempo razonable.

La solución a la que se llegó durante la implementación es la de utilizar métodos en la lógica que eviten tener que realizar copias. Estos métodos son *execute_move* y *undo_move*. El método para ejecutar un movimiento ya existía, pero el método para deshacer un movimiento permite que los nodos no modifiquen el estado del tablero. Cuando se van a obtener los hijos de un nodo se puede utilizar un solo tablero y se procede de esta forma: se ejecuta un movimiento sobre el tablero, se llama al algoritmo Minimax con este nuevo estado (creando así un nuevo nodo) y finalmente se deshace el movimiento volviendo así al estado original. Esto se repite con cada uno de los posibles movimientos para una posición. El proceso se ilustra en código en la Ilustración 36.

```
for move in moves:
    board.execute_move(move, player)
    score, _ = self._minimax(board, -player, alpha, beta, depth + 1)
    board.undo_move(move, player)
```

Ilustración 36 Código para creación de nodos a partir de acciones

Para dar una idea de la diferencia en el rendimiento se ha realizado una comparativa de los 3 enfoques: copia manual, deepcopy y deshacer movimientos. La comparativa en tiempo se muestra en la tabla Tabla 55. En ella se exponen los resultados de seguir cada enfoque en 1000 iteraciones repetidas en 3 momentos distintos para obtener la media.

Método	Tiempo exp1 (ns)	Tiempo exp2 (ns)	Tiempo exp3 (ns)	Media (ns)
deepcopy	36.033.000	37.033.400	35.032.000	36.032.800
copy (columnas)	4.003.600	3.002.700	3.003.100	3.336.467
undo move	1.000.900	1.000.200	1.000.600	1.000.567

Tabla 55 Comparación copias de tablero

Como se puede ver en los resultados de los experimentos, la función *deepcopy* es muy lenta comparada con los otros enfoques. Además, utilizar un método para deshacer movimientos es, de media, 3 veces más rápido que una copia manual con la función *copy*.

7.4.1.2 Evaluación Territorial Minimax

Durante la implementación de las diferentes evaluaciones para el algoritmo Minimax se consultaron varios artículos, algunos de ellos mencionados en el Capítulo 9. Uno de los artículos más interesantes fue [34]. Una explicación más extensa del artículo se provee en apartado 9.2.

De las diferentes formas de evaluar un tablero, se proponen varios parámetros (t_1 , t_2 , c_1 y c_2) a los que se les da diferente importancia según la fase de la partida, obtenida a partir del cálculo del parámetro w aplicándole una función f_i . Las funciones f_1 , f_2 , f_3 y f_4 de la Ecuación 2 no son presentadas de forma explícita. Se mencionan restricciones

que deben de cumplir, sin embargo, buscar funciones que cumplan las restricciones propuestas y doten de diferente importancia a los diferentes parámetros no es una tarea trivial. Es por esto por lo que se ha decidido utilizar varios de los conceptos del artículo para desarrollar una función de evaluación propia. Esta función de evaluación guía la búsqueda del algoritmo implementado en la clase `MinimaxAlgorithmTerritoryMobility`.

$$t = f_1(w)t_1 + f_2(w)c_1 + f_3(w)c_2 + f_4(w)t_2$$

Ecuación 2 Cálculo de la evaluación territorial en [34]

La función de evaluación utilizada en la clase `MinimaxAlgorithmTerritoryMobility` se basa en el uso de t_1 , la evaluación territorial básica basada en movimientos de reina. El parámetro t_1 se suma al parámetro m que se calcula de la forma propuesta por el artículo. Sin embargo, la función que combina w con α tampoco se presenta. Para esta función se ha utilizado la Ecuación 3 a la que se ha llegado de forma experimental para balancear adecuadamente la movilidad de las amazonas de forma individual durante la partida y el territorio del jugador.

$$f(w, \alpha) = w \frac{\alpha}{100}$$

Ecuación 3 Función para el balanceo de w y α

7.4.1.3 Epsilon-Greedy MCTS

Para los diferentes algoritmos de MCTS implementados en el sistema se han consultado artículos científicos. Concretamente para la implementación de la clase `MCTSAlgorithmEGreedy` se ha consultado el artículo [36]. Una explicación más extensa del artículo se encuentra en el apartado 9.4. Como se puede ver en dicha explicación el artículo deja varios aspectos sin concretar. El primer aspecto es la probabilidad de elegir la acción voraz, que se ha dejado en $1 - \epsilon$. Para la obtención de Q y su posterior retro propagación no se ha seguido la fórmula de la Ecuación 6 Actualización del valor de una acción según [36] propuesta en el artículo, ya que esta dejaba varios parámetros sin especificar. Es por esto que se actualiza el valor Q de los nodos simplemente sumando al anterior valor Q el resultado de la simulación de la partida (1 en caso de victoria, 0 en caso de derrota).

De forma adicional a la clase `MCTSAlgorithmEGreedy` se ha creado una clase similar, `MCTSAlgorithmEGreedyMod`. En esta clase se pretende dar un enfoque más basado en el problema MAB [39]. Es por esto que se calcula y propaga Q de otra forma distinta. El objetivo es intentar que el algoritmo promueva finales de partida con la mayor ventaja posible (es decir, que busque llegar al final de la partida con la mayor superioridad de movimientos disponibles posible). Para lograr dicho objetivo se debe evaluar una victoria ajustada como menos valiosa que una victoria holgada. Es por eso por lo que se utiliza la Ecuación 4 para evaluar una partida terminada, donde x representa la ventaja de movimientos del ganador. Por ejemplo, una partida que acabase en victoria de las blancas con una ventaja de 2 movimientos tendría $x = 2$ y $Q = 0,7$. Esta fórmula solo se utiliza para rangos de x en los intervalos $(-\infty, -1]$ y $[1, \infty)$ ya que una partida no puede quedar en empate por lo que valores entre -1 y 1 nunca aparecerán. Este es el valor

que se propagará a través del árbol de búsqueda de Monte Carlo y servirá para guiar la búsqueda.

$$y = \begin{cases} 1 - \frac{1}{2 + \frac{2x}{3}} & \text{si } x \in [1, \infty) \\ 1 - \frac{1}{2 - \frac{2x}{3}} & \text{si } x \in (-\infty, -1] \end{cases}$$

Ecuación 4 Cálculo retro propagación de Q modificado

7.4.2 Descripción Detallada de las Clases

En este apartado se presentará la ruta donde se encuentra la documentación del código dentro del entregable con el sistema. La ruta es la siguiente:

`/game_of_the_amazons_ai/docs/index.html`.

En esta documentación se encuentran comentados todos los módulos, clases, funciones y métodos principales del sistema.

Capítulo 8. Desarrollo de las Pruebas

En este capítulo se presentarán los resultados de las pruebas realizadas sobre el sistema descritas en el apartado 6.6.

8.1 Pruebas Unitarias

En la Tabla 56 se presentan los resultados de las pruebas unitarias.

Nombre de la clase	Nombre del test	Descripción	Resultado
AlgorithmTest			
	test_make_move	Comprobar que los algoritmos dan movimientos legales	OK
BoardTest			
	test_init	Comprobar que el constructor de la clase funciona correctamente	OK
	test_is_valid_move	Comprobar que se valida correctamente los movimientos	OK
	test_execute_move	Comprobar que se ejecutan correctamente los movimientos	OK
	test_undo_move	Comprobar que se deshacen correctamente los movimientos	OK
	test_eq	Comprobar que se comparan correctamente tableros	OK
	test_get_legal_moves	Comprobar que se obtienen los movimientos	OK

		de forma correcta	
	test_is_win	Comprobar que se valida correctamente si la partida ha terminado en victoria	OK

Tabla 56 Resultado de los tests unitarios

8.2 Pruebas de Integración y del Sistema

En este apartado se presentará el resultado de utilizar la checklist de la Tabla 49 presentada en el apartado 6.6.2.

ID	Resultado
1	Se muestra la interfaz gráfica correctamente
1'	El tablero se presenta en la posición inicial adecuada
1''	El menú de elección de jugadores se muestra con el jugador humano preseleccionado
1'''	El botón de iniciar partida se muestra correctamente
2a	Al pulsar el menú de elección de jugadores para las blancas este se despliega
2a'	Al pulsar uno de los posibles jugadores o fuera del desplegable para las blancas este se repliega
2b	Al pulsar el menú de elección de jugadores para las negras este se despliega
2b'	Al pulsar uno de los posibles jugadores o fuera del desplegable para las negras este se repliega
3	Al pulsar el botón de iniciar partida este desaparece
3'	Al iniciar partida no es posible cambiar los jugadores
4	Ambos jugadores realizan los movimientos correctamente
4'	Cuando la partida termina se muestra el ganador
4''	Cuando la partida termina se vuelve a mostrar el botón de reiniciar partida
5	Tras terminar la partida se despliega el menú de elección de jugadores al pulsar sobre él
5'	Tras pulsar sobre uno de los posibles jugadores
6	Tras reiniciar la partida, el botón de reiniciar partida desaparece
6'	Al reiniciar la partida el tablero vuelve al estado inicial
6''	Al reiniciar la partida ya no se pueden cambiar los jugadores
7	Ambos jugadores realizan correctamente los movimientos
7'	Al terminar la partida se muestra el jugador
7''	Al terminar la partida se muestra el botón de reiniciar
8	Al pulsar el botón de cerrar la ventana termina la ejecución del programa

Tabla 57 Resultados pruebas de integración y del sistema

8.3 Pruebas de Usabilidad y Accesibilidad

En este apartado se presentarán los resultados de las pruebas de usabilidad y accesibilidad.

8.3.1 Pruebas de Usabilidad

Los resultados de las pruebas de usabilidad serán presentados como los cuestionarios propuestos en el apartado 6.6.3.1 y rellenados por los usuarios que participaron en las pruebas.

8.3.1.1 Usuario 1

8.3.1.1.1 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
<ol style="list-style-type: none"> 1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
<ol style="list-style-type: none"> 1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
<ol style="list-style-type: none"> 1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
<ol style="list-style-type: none"> 1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias

Tabla 58 Preguntas de carácter general usuario 1

8.3.1.1.2 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?	X			
¿Existe ayuda para las funciones en caso de que tenga dudas?				X
¿Le resulta sencillo el uso de la aplicación?		X		
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca

¿Funciona cada tarea como Vd. Espera?	X			
¿El tiempo de respuesta de la aplicación es muy grande?				X
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letras		X		
Los iconos e imágenes usados son		X		
Los colores empleados son		X		
Diseño de la Interfaz		Si	No	A veces
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?				X
¿Cree que el programa está bien estructurado?		X		

Tabla 59 Preguntas cortas sobre la aplicación usuario 1

8.3.1.2 Cuestionario para el Responsable de las Pruebas

Notas
Al principio al usuario le cuesta entender que no se pueden capturar piezas.

Tabla 60 Cuestionario para el responsable de las pruebas sobre el usuario 1

8.3.1.3 Usuario 2

8.3.1.3.1 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
<ol style="list-style-type: none"> 1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
<ol style="list-style-type: none"> 1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
<ol style="list-style-type: none"> 1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca

¿Qué busca Vd. Principalmente en un programa?

1. Que sea fácil de usar
2. Que sea intuitivo
3. Que sea rápido
4. Que tenga todas las funciones necesarias

Tabla 61 Preguntas de carácter general usuario 2

8.3.1.3.2 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?	X			
¿Existe ayuda para las funciones en caso de que tenga dudas?				X
¿Le resulta sencillo el uso de la aplicación?	X			
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Funciona cada tarea como Vd. Espera?	X			
¿El tiempo de respuesta de la aplicación es muy grande?			X	
Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son	X			
Los colores empleados son		X		
Diseño de la Interfaz		Si	No	A veces
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?		X		
¿Cree que el programa está bien estructurado?		X		

Tabla 62 Preguntas cortas sobre la aplicación usuario 2

8.3.1.4 Observaciones

Observaciones

Lento al principio, quizá se podrían mostrar mensajes o avisos indicando cuándo ha realizado una IA un movimiento.

Tabla 63 Observaciones usuario 2

8.3.1.5 Cuestionario para el Responsable de las Pruebas

Notas

Al principio al usuario le resulta difícil entender que un movimiento consta de dos partes.

Al principio al usuario le cuesta entender cuándo termina la partida.

Tabla 64 Cuestionario para el responsable de las pruebas sobre el usuario 2

8.3.1.6 Usuario 3

8.3.1.6.1 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
<ul style="list-style-type: none"> 5. Todos los días 6. Varias veces a la semana 7. Ocasionalmente 8. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
<ul style="list-style-type: none"> 5. Es parte de mi trabajo o profesión 6. Lo uso básicamente para ocio 7. Solo empleo aplicaciones estilo Office 8. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
<ul style="list-style-type: none"> 4. Sí, he empleado software similar 5. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 6. No, nunca
¿Qué busca Vd. Principalmente en un programa?
<ul style="list-style-type: none"> 5. Que sea fácil de usar 6. Que sea intuitivo 7. Que sea rápido 8. Que tenga todas las funciones necesarias

Tabla 65 Preguntas de carácter general usuario 3

8.3.1.6.2 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?	X			
¿Existe ayuda para las funciones en caso de que tenga dudas?				X
¿Le resulta sencillo el uso de la aplicación?		X		
Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Funciona cada tarea como Vd. Espera?	X			
¿El tiempo de respuesta de la aplicación es muy grande?				X

Calidad del Interfaz				
Aspectos gráficos	Muy Adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son	X			
Los colores empleados son		X		
Diseño de la Interfaz		Si	No	A veces
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?		X		
¿Cree que el programa está bien estructurado?		X		

Tabla 66 Preguntas cortas sobre la aplicación usuario 3

8.3.1.7 Observaciones

Observaciones
El juego podría ser más vistoso y atractivo visualmente.

Tabla 67 Observaciones usuario 3

8.3.1.8 Cuestionario para el Responsable de las Pruebas

Notas
Al usuario le resulta extraño que no se puedan arrastrar las piezas por el tablero.
El usuario en ocasiones no se da cuenta de que una IA ha realizado un movimiento por lo rápido que es.

Tabla 68 Cuestionario para el responsable de las pruebas sobre el usuario 3

8.3.1.9 Correcciones

Debido a que la interfaz gráfica no es una parte fundamental del proyecto se ha procurado realizar la menor cantidad posible de cambios en la interfaz, procurando no utilizar demasiados recursos en ella, pero intentando ofrecer un mejor producto. El cambio principal realizado ha sido cambiar el tiempo que una inteligencia artificial toma para realizar un movimiento. Se ha pasado de 1 a 2 segundos. Este tiempo es suficiente para que se vea perfectamente el movimiento, pero no es demasiado para que no añada demasiado tiempo de espera al usuario humano.

8.3.2 Pruebas de Accesibilidad

En esta sección se completará el cuestionario propuesto en 6.6.3.6 en referencia a las pruebas de accesibilidad.

Preguntas:	Respuestas:
¿La aplicación contiene mnemónicos como alternativa a todas las operaciones con ratón?	No

¿Se proveen instrucciones en el manual de usuario?	Sí
¿Es sencillo el uso de la aplicación usando la documentación?	Sí
¿Las pestañas están ordenadas de forma lógica garantizando buena navegabilidad?	No aplica
¿Hay atajos para acceder a los menús?	No aplica
¿Soporta todos los sistemas operativos?	No
¿Son apropiados para todos los usuarios los colores de la aplicación?	Sí
¿Están escritas correctamente todas las etiquetas?	Sí
¿Son las imágenes e iconos fáciles de entender?	Sí
¿Tienen sonido las alertas de la aplicación?	No aplica
¿Se puede cambiar la fuente y su tamaño?	No, pero se utiliza un tamaño grande de letra
¿Se usa el color como único medio para transmitir información?	No

Tabla 69 Resultado pruebas de accesibilidad

Pese a que se han encontrado algunos fallos o aspectos a mejorar en cuanto a accesibilidad, se ha decidido no llevar a cabo ningún proceso de corrección. Esto se ha decidido debido a que la interfaz gráfica no supone una parte de gran importancia en el proyecto. Con las pruebas de usabilidad y accesibilidad se ha comprobado que un usuario es capaz de utilizar la interfaz gráfica. Mejoras en la interfaz o su accesibilidad son propuestas para proyectos o mejoras futuras, ya que estas no forman parte de los objetivos o del alcance de este proyecto.

8.4 Pruebas de Rendimiento

En este apartado se presentarán los resultados de las pruebas de rendimiento presentadas en el apartado 6.6.4. Como se explicó en dicho apartado primero se presenta la Ilustración 37 Profiling del sistema con datos extraídos de un proceso de profiling sobre la ejecución del algoritmo Minimax con evaluación de movilidad y territorial (uno de los que más operaciones diferentes aplica al tablero). En amarillo se muestran los métodos de la lógica del sistema, en naranja los métodos del algoritmo Minimax y en gris otras funciones y métodos. Para ver el resultado del profiling original se pueden consultar los archivos adjuntos, en concreto tests/profiling.txt.

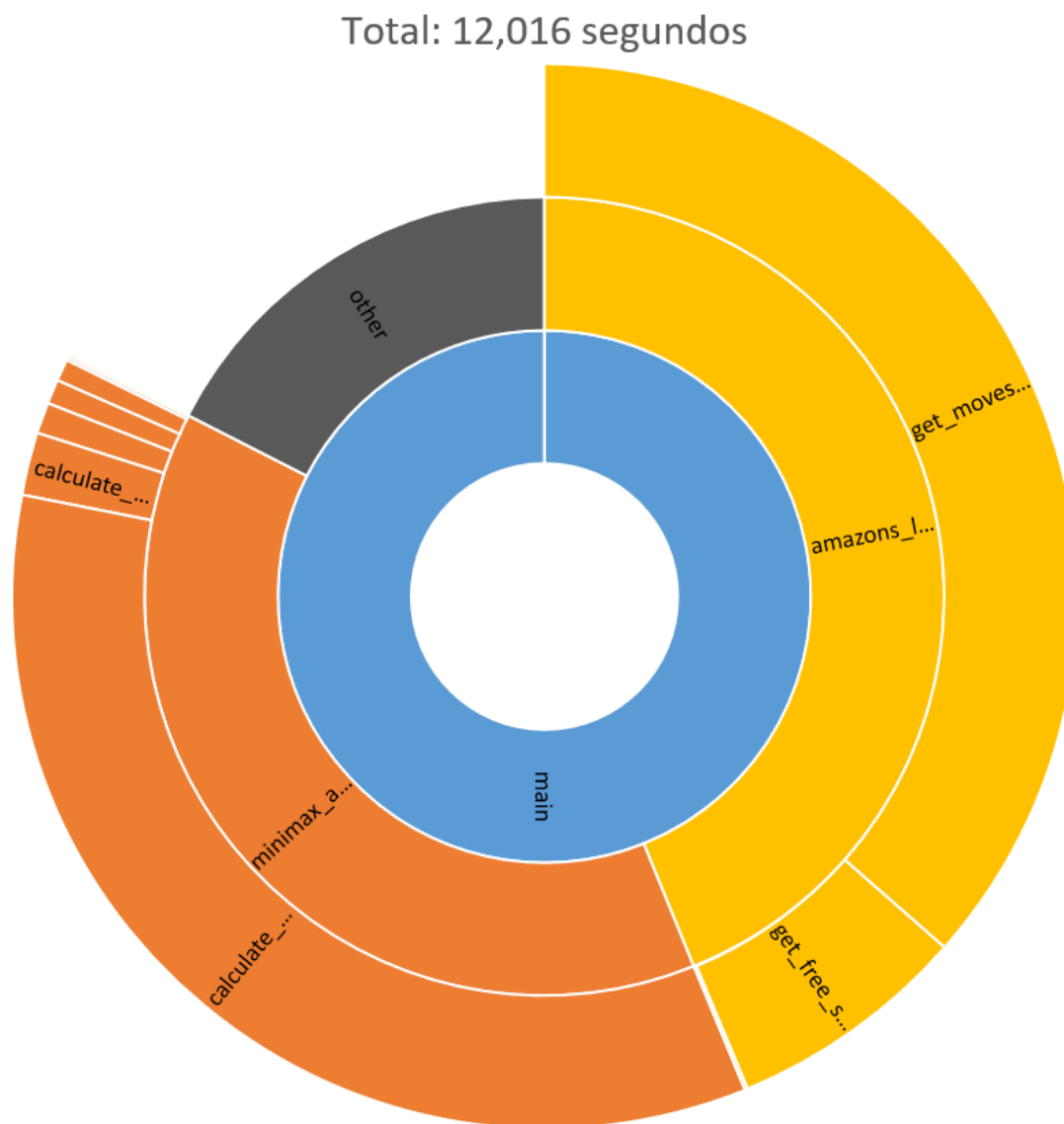


Ilustración 37 Profiling del sistema

El proceso de profiling ha sido muy útil debido a que se ha descubierto una mejora sustancial en la velocidad de ejecución de los algoritmos. Esta mejora recae en el método `is_win`, con el que se comprueba en el tablero si en una posición ha ganado un jugador determinado. Antes de realizar el profiling, se obtenían los movimientos de ambos jugadores y se comprobaba si el jugador opuesto tenía 0 movimientos legales. En este caso el jugador habría ganado y en caso contrario, no. Esto es muy lento debido a que se ha de calcular todos los movimientos legales solo para comprobar si un jugador ha ganado. En momentos de la partida en los que existen muchos movimientos permitidos para los jugadores, esto puede ser muy lento.

Tras el profiling se ha cambiado la implementación para que sea más rápido el proceso. En la versión final del método `is_win` se comprueba si alrededor de cada una de las piezas de un jugador hay casillas libres. Si y solo si todas las casillas alrededor de las piezas de un jugador están ocupadas (por una pieza o una flecha bloqueando la casilla)

este ha perdido. En la Ilustración 38 se puede ver el profiling antes de la mejora. Como se puede ver, se ha reducido el tiempo de obtención del primer movimiento a casi un tercio del original.

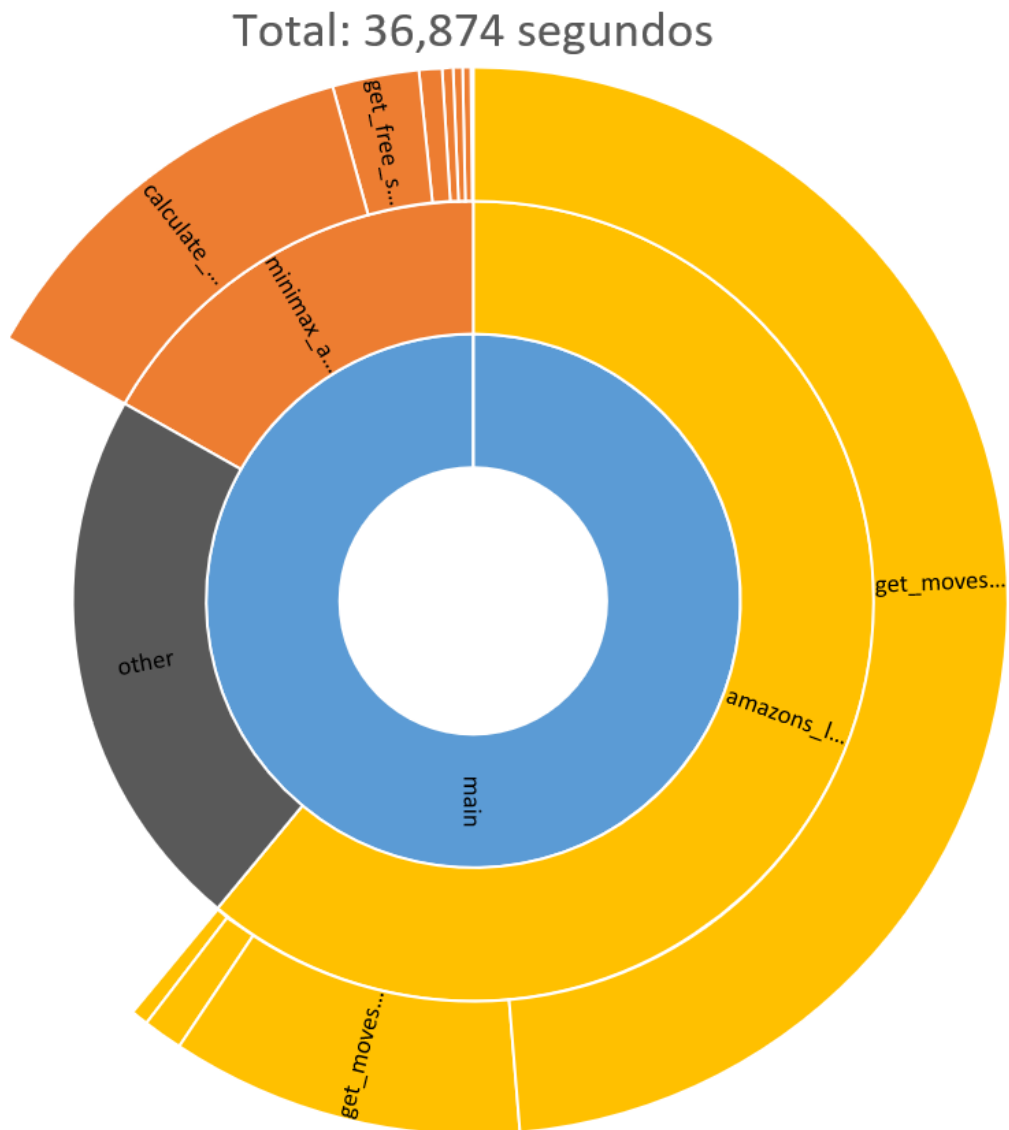


Ilustración 38 Profiling del sistema previo

Capítulo 9. Estudio Teórico

En este capítulo se estudiarán diferentes artículos científicos relacionados con algún aspecto del proyecto. Los objetivos de este estudio son:

- Conocer el estado del arte sobre inteligencias artificiales para el Juego de las Amazonas.
- Evaluar diferentes algoritmos inteligentes para determinar cuáles se pueden usar para el proyecto.
- Documentar de forma resumida la información extraída de los artículos científicos.

9.1 A Knowledge-based Approach of the Game of amazons (Hensgens P, Uiterwijk J et al. 2001)

En este artículo científico [29] publicado para la universidad de Maastricht, se presentan y estudian varias ideas para mejorar el algoritmo de poda Alpha-Beta (ver Minimax). El conocimiento adquirido se implementa en una inteligencia artificial llamada *Anky* que recopila las mejoras que los autores consideran más importantes.

En el artículo se trata varios aspectos cruciales en la mejora del algoritmo Alpha-Beta. Estos aspectos son: mejoras en la creación del árbol de búsqueda, funciones de evaluación de posición, ordenación de los movimientos y otras mejoras diversas.

Para mejorar el funcionamiento del algoritmo Alpha-Beta se han considerado 2 principales estrategias:

Iterative deepening (profundización iterativa): esta estrategia sirve para optimizar el algoritmo Alpha-Beta cuando se usa en conjunto con otras estrategias. La profundización iterativa consiste en aumentar la profundidad a la que llega el algoritmo Alpha-Beta de forma iterativa durante una búsqueda. De esta forma, en la primera iteración el árbol de búsqueda alcanzaría una profundidad de 1, en la segunda iteración 2 y así sucesivamente hasta que se agoten los recursos establecidos para el algoritmo (tiempo o memoria). La principal ventaja que ofrece esta técnica es que se puede utilizar el conocimiento adquirido en una iteración para que la siguiente iteración sea más eficiente. Esto se puede hacer utilizando las evaluaciones de una iteración para ordenar los movimientos que serán evaluados en la siguiente iteración. También se puede combinar con técnicas de ordenación como *Killer moves*, *history heuristic*, *transposition tables* que serán tratadas más adelante.

Selective search (búsqueda selectiva): esta estrategia es muy simple, pero puede mejorar la eficiencia de Alpha-Beta. Consiste en seleccionar solo los mejores (más prometedores) movimientos para que sean evaluados por el algoritmo. De esta forma se estarían 'podando' movimientos menos prometedores para que el algoritmo solo se

centre en los mejores. Esto hace que el árbol sea mucho más estrecho y no se pierdan recursos en movimientos no prometedores. El riesgo que se corre es que un movimiento que para una profundidad concreta sea no prometedor y se descarte, resulte ser muy buen movimiento si se evaluase a una mayor profundidad. Las posibilidades de que esto ocurra dependen de lo buena que sea la función de evaluación y de la profundidad a la que se llegue en la creación del árbol.

Las funciones de evaluación de posición consideradas en el artículo son las siguientes:

Random evaluation (evaluación aleatoria): como el nombre indica, esta evaluación dota de una puntuación aleatoria a una posición concreta. Si bien no tiene utilidad práctica, se usa en el artículo para comprobar ciertas propiedades de las funciones de evaluación. Puede parecer sorprendente, pero utilizando esta evaluación y debido a la naturaleza del algoritmo Alpha-Beta, se tiende a favorecer posiciones de mayor movilidad. Esto es debido a que hay más posibilidades de encontrar una evaluación alta para movimientos que generan posiciones de mayor movilidad.

Mobility evaluation (evaluación de movilidad): esta evaluación es simple y muy rápida. Se trata de restar el número de movimientos posibles para las blancas y restarle el número de movimientos posibles para las negras. De esta forma, si tienen mayor movilidad las piezas blancas, la evaluación será positiva y si tienen mayor movilidad las negras, será negativa.

Territory evaluation (evaluación territorial): se trata de calcular el número de movimientos que las piezas blancas y negras necesitan para llegar a cada una de las casillas libres del tablero. Una vez se ha asignado un número a cada casilla libre existen varias posibilidades para cada casilla: si es alcanzable por blancas y negras, la casilla será del 'territorio' de las piezas que lleguen en menos movimientos. Si las piezas de un color pueden llegar a la casilla y las piezas del otro color no, será directamente del 'territorio' de las piezas que puedan llegar. Si ninguna pieza puede acceder a la casilla se dice que no es del 'territorio' de blancas o negras. Para los dos primeros casos se dota de un valor de 1 o -1 a la casilla (dependiendo de si son blancas, 1, o negras, -1, las que llegan antes). En el tercer caso se ha de establecer un valor para la casilla, pero esto no se comenta en este artículo, si no en otros posteriores.

Relative territory evaluation (evaluación territorial relativa): esta evaluación es muy similar a la evaluación territorial. La única diferencia es que se calcula la diferencia de movimientos necesarios para llegar a una casilla de blancas y negras. De esta forma se tiene en cuenta lo 'rápido' que llega un jugador a la casilla frente al otro. En el caso de que un jugador pueda llegar a una casilla y el otro no, hay que establecer una puntuación (diferente de infinito) para darle a la casilla.

Para la ordenación de movimientos se consideran 3 formas de persistencia para poder conservar conocimiento y utilizarlo para la ordenación posteriormente. También se presenta una forma básica de ordenación sin necesidad de persistencia.

Killer moves (movimientos asesinos): este heurístico se basa en la idea de que el mejor movimiento en una posición podría ser un muy buen movimiento o incluso el mejor en otra posición similar. De esta forma si se guardan los movimientos 'asesinos' (mejor

movimiento en una posición), se podrá utilizar esta información para evaluar primero estos movimientos en un futuro.

History heuristic (heurístico histórico): este heurístico es similar al anterior con la diferencia de que se guarda información sobre todos los movimientos, no solo los 'asesinos'. Se guarda la evaluación de cada movimiento en dos matrices multidimensionales que indican las casillas del movimiento de la Amazona y del disparo de la 'flecha'. Esta información puede ser usada más tarde para ordenar los movimientos.

Transposition tables (tablas de transposición): esta estrategia es muy eficaz y utilizada en el ajedrez. Se trata de guardar la posición actual de la partida junto con su evaluación. De esta forma, si se encuentra la misma posición en otro momento de la partida, no es necesario volver a evaluarla, ahorrando cómputo. Esto es muy efectivo en el ajedrez, pero en el Juego de las Amazonas no existen muchas transposiciones (movimientos que hacen que se llegue a una misma posición de formas distintas). Este hecho es debido a la naturaleza del juego, donde una posición con un número determinado de casillas bloqueadas solo puede existir tras un mismo número de movimientos.

Static board scores (evaluaciones estáticas del tablero): otra opción es computar una evaluación de cada movimiento antes de iniciar la búsqueda para ordenar los movimientos. Pero no es un método tan fuerte debido a que las evaluaciones estáticas tienen poca precisión a la hora de determinar movimientos prometedores.

Adicionalmente en el artículo se presentan otras posibles mejoras para el algoritmo Alpha-Beta:

Principal variation search (PVS): es un método empleado para reducir el esfuerzo de clasificar buenos y malos movimientos, para más información consultar [30].

Opening book (libros de apertura): muy utilizado en ajedrez, se trata de evaluar de antemano cuáles son los mejores movimientos al principio de la partida para guardarlos en un sistema de persistencia. Normalmente se basan en conocimiento humano, efectivo para juegos más conocidos como ajedrez, pero menos útil para juegos con menos jugadores como el Juego de las Amazonas.

Endgame search (búsqueda en el final de partida): en los estados finales de una partida del Juego de las Amazonas, el tablero se divide en áreas en las que todas las amazonas que están dentro son de un mismo jugador. En estos estados se puede determinar con certeza la secuencia de movimientos óptima. Estas secuencias se pueden calcular y almacenar en un sistema de persistencia o utilizar un algoritmo diferente para esta fase del juego. Encontrar la secuencia de movimientos óptima en para llenar un área equivale a la búsqueda de un circuito de Hamilton en un grafo [31].

Por último, el artículo realiza diferentes comparaciones entre las diversas estrategias aplicadas al algoritmo de poda Alpha-Beta y presenta propiedades del Juego de las Amazonas para diferentes tamaños de tablero (4x4, 5x5, 6x6 y 7x7).

En los anexos se presentan resultados de la 5ª y 6ª *Computer Olympiad* ([32], [33]) y una base de datos para el tablero 4x4.

9.2 An evaluation function for the game of amazons, Lieberum J. 2005

En este artículo [34] se tratan varias ideas sobre la evaluación de posiciones en el Juego de las Amazonas. Algunas de ellas fueron ya mencionadas en el apartado anterior. Sin embargo, este artículo introduce algunos nuevos elementos y profundiza sobre otros. Esto lo hace en el contexto de una inteligencia artificial real creada por el autor y llamada *Amazong*. En la segunda mitad del artículo se tratan conceptos relacionados con partidas del Juego de las Amazonas como son los finales de partida o los *zugswangs* (situaciones en las que todos los posibles movimientos de un jugador son perjudiciales para él).

En el artículo se describe de forma similar al apartado 9.1 la evaluación territorial. Sin embargo, también se introduce otras formas de calcularla y distintos enfoques. Hasta ahora se ha explicado la evaluación territorial mediante el número de movimientos en los que un jugador llega a una casilla. Como las amazonas se mueven como reinas en ajedrez, esto serían movimientos de reina. Otra forma de calcular la distancia de una pieza a una casilla es por medio de movimientos de rey (igual que un rey en ajedrez, cualquier dirección, pero solo una casilla a la vez). Estas 2 formas de calcular las distancias se pueden utilizar en conjunto y para distintas fases de la partida. Además de t_1 y t_2 (evaluaciones territoriales comentadas anteriormente) se proponen 2 evaluaciones más: c_1 y c_2 . Estos parámetros se utilizan a la hora de calcular el valor de cada casilla en la evaluación territorial relativa. Estos dan un valor concreto para la diferencia en número de movimientos para alcanzar una casilla tanto usando movimientos de reina (c_1) como de rey (c_2). Lieberum propone utilizar t_1 , t_2 , c_1 y c_2 de forma conjunta con 4 funciones (f_1 , f_2 , f_3 y f_4) que ajusten la importancia de cada parámetro en función de la fase de la partida. Para tener un valor que indique la fase de la partida se propone el valor w . Este valor oscila entre 0 y 1. Si $w = 0$ entonces la posición de la partida pertenece a la '*filling phase*' o fase de llenado (ver Juego de las Amazonas).

Pese a todo lo comentado en el párrafo anterior, las funciones f_i (f_1 , f_2 , f_3 y f_4) no son especificadas en ningún momento. Solo se citan algunas de las características matemáticas que deberían tener. Tampoco se hace referencia a ninguna implementación de la inteligencia artificial que utiliza dichas funciones.

Una aportación importante del artículo es la consideración de la movilidad de amazonas individuales. Este concepto se utiliza con el objetivo de que la inteligencia artificial desarrollada por el autor busque atrapar las amazonas del rival en espacios pequeños. Esto se promueve dotando a la evaluación de un factor de movilidad a cada amazona. La evaluación territorial no castiga a un jugador que tenga las piezas atrapadas en espacios pequeños. En este artículo se utiliza en combinación con el parámetro w (comentado en el párrafo anterior) para penalizar de una forma más agresiva la falta de movilidad de las amazonas al principio de la partida.

La fórmula utilizada por el autor para ajustar la importancia en función de la fase de la partida tampoco se especifica. Esta vez, sin embargo, es más sencillo hacerse una idea de posibles funciones que cumplan que $f \geq 0$ y que $f(0, y) = 0$.

9.3 Comparative Study of Monte-Carlo Tree Search and Alpha-Beta Pruning in Amazons, Kato et al. 2015

En este artículo [35], se compara el algoritmo de poda Alpha-Beta y el algoritmo de búsqueda en árboles de Monte Carlo (ver Monte Carlo Tree Search). Para el algoritmo Alpha-Beta utiliza diferentes funciones de evaluación y para el algoritmo MCTS utiliza diferente número de simulaciones. Utiliza las mismas funciones de evaluación que en el apartado 9.1 para Alpha-Beta y utiliza la función UCB mostrada en Ecuación 5 para MCTS.

$$UCB1(G_i) = \bar{x}_i + \sqrt{\frac{\log n}{n_i} \min\left(\frac{1}{4}, \bar{x}_i - \bar{x}_i^2 + \sqrt{\frac{2 \log n}{n_i}}\right)}$$

Ecuación 5 Función UCB utilizada en [35]

Por último, se presentan los resultados experimentales y sus resultados. Los autores llegan a la conclusión de que cambiar la función de evaluación de Alpha-Beta influye mucho más en la fuerza de la IA que cambiar el número de simulaciones en MCTS.

9.4 Monte-Carlo tree search with Epsilon-Greedy for game of amazons, Tian C. 2023

En el artículo [36] se propone el uso de una función heurística en conjunto con un árbol de búsqueda de Monte Carlo (ver Monte Carlo Tree Search). También se propone el uso de una política Epsilon-Greedy (más información [37]).

En el artículo se define y expone un ejemplo en pseudocódigo de la forma tradicional del árbol de búsqueda de Monte Carlo. Después se introducen las novedades que propone el artículo. Lo primero es el uso de una función heurística para la evaluación y posterior ordenación de movimientos. Esta función es la misma que la evaluación de movilidad ya explicada en el apartado 9.1. Lo segundo es la política ϵ -Greedy, esta es una forma de balancear explotación y exploración en problemas como Multi-Armed Bandits [38] basado en el uso de la probabilidad. La acción con mayor valor según la función de evaluación (denominada la acción voraz) tiene una probabilidad determinada de ser escogida. Si esta no se escoge, se selecciona una de las no voraces mediante una distribución de probabilidad. Lo tercero es una explicación de cómo se puede aplicar dicha política a las fases del algoritmo MCTS. Por último, se presenta una comparación entre Alpha-Beta y MCTS junto con las conclusiones del autor.

Cabe destacar que el autor explica en el artículo algunos conceptos de forma poco clara e incluso contradictoria en algunas partes. Además, algunas especificaciones difieren del consenso general científico. En el apartado 3.1. *Epsilon-Greedy method* se describe ϵ como “A small positive number $\epsilon (< 1)$ representing the probability is used to select unknown action in a random manner, leaving $1 - \epsilon$ probability of selecting the action with highest value, which is defined as a greedy action”. Por lo que se entiende en esta cita, el número ϵ representa la posibilidad de que se seleccione una acción no voraz y que $1 - \epsilon$ es la probabilidad de elegir una acción voraz. Sin embargo, en el apartado 3.2.1. *Selection* se dice lo contrario, “The node with the largest Q value will be first selected by the greedy policy with probability of ϵ ”. Dando a entender que la probabilidad de seleccionar la acción voraz es ϵ . En general, para políticas Epsilon-Greedy se usa $1 - \epsilon$ como la probabilidad de escoger la acción voraz ([39], [40], [41]). Cabe destacar, además, que, para la retro propagación, no se explican todos los parámetros utilizados para actualizar la recompensa (valor Q) de las acciones. En la fórmula de la Ecuación 6 se explica que Q es el valor de una acción en un estado concreto, que d es la profundidad inversa del estado (tendiendo así un estado terminal $d = 0$), y es un factor de decadencia que controla cuánto se propaga el resultado de una partida, aunque no se indica ningún posible valor para él. Para el valor α no se adjunta ninguna explicación ni ningún valor posible.

$$Q'(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \gamma^d$$

Ecuación 6 Actualización del valor de una acción según [36]

Capítulo 10. Estudio Experimental

En este apartado se explicarán y se presentarán los resultados experimentales de la comparación entre los diferentes algoritmos implementados. Todos los algoritmos (salvo el aleatorio) serán comparados primero con el aleatorio y con los algoritmos voraces para comprobar que tienen un mínimo nivel a la hora de jugar. Luego, cada implementación del algoritmo será comparada con el resto de las implementaciones para intentar entender el nivel de cada implementación. Por último, se comparan algoritmos entre sí, de esta forma se puede comparar el nivel de juego de implementaciones de distintos algoritmos.

El resultado de las partidas será expuesto en una tabla en la que figurará el nombre de ambos algoritmos implicados, el número de partidas y el resultado de las partidas. Para cada enfrentamiento el resultado se indicará como el número de victorias del primer algoritmo frente al número de victorias del segundo algoritmo. Debido a que no existen empates en el Juego de las Amazonas, la suma de ambos números será igual al número total de partidas. Por ejemplo, si se juegan 100 partidas entre el algoritmo A y el algoritmo B, un resultado de 60/40 significa que el algoritmo A ha ganado 60 de las 100 y el algoritmo B ha ganado 40 de las 100 partidas. Se ha de tener en cuenta que el algoritmo que juegue con piezas blancas tiene cierta ventaja, al tener el primer movimiento. Es por esto por lo que las series de partidas son siempre un número par (100, 50) para que en la primera mitad de las partidas el primer algoritmo juegue con blancas y el segundo con negras y que en la segunda mitad el primer algoritmo juegue con negras y el segundo con blancas.

10.1 Aleatorio

El aleatorio no tiene gran interés debido a que consiste únicamente en la elección aleatoria de movimientos. Sin embargo, este algoritmo sirve como base para comprobar que todas y cada una de las inteligencias artificiales la superen. Si una inteligencia artificial tuviese el mismo desempeño que el algoritmo aleatorio esto significaría que realmente la inteligencia artificial no supone una mejoría con respecto a no usar un algoritmo.

10.2 Voraz

Los algoritmos voraces implementados son 2.

Uno de ellos se basa en la evaluación de movilidad del tablero (ver 9.1). Esta consiste en obtener la diferencia entre el número de movimientos que tienen las piezas blancas y el número de movimientos de las piezas negras. Cuantos más movimientos tenga el jugador y menos tenga el oponente, mayor será la diferencia, el movimiento que maximice esa diferencia será elegido por el algoritmo voraz.

El otro algoritmo se basa en la evaluación territorial del tablero (ver 9.1). Esta evaluación analiza el tablero y calcula el número de movimientos necesarios para que cada jugador alcance cada una de las casillas vacías. Si un jugador alcanza una casilla en menos movimientos que su oponente, se dice que pertenece a su territorio y se le otorga una puntuación a la casilla. El algoritmo voraz elegirá el movimiento que permita al jugador tener un mayor territorio frente a su oponente.

Primero se comparan las 2 implementaciones del algoritmo voraz (con heurístico de movilidad y heurístico de territorio) con el algoritmo aleatorio en la Tabla 70.

Algoritmo 1	Algoritmo 2	Resultado
Voraz Movilidad	Aleatorio	100/0
Voraz Territorial	Aleatorio	100/0

Tabla 70 Comparación voraz y aleatorio

Como se puede ver, ambos heurísticos para el algoritmo voraz superan al algoritmo aleatorio.

Después se presentan los resultados de comparar el algoritmo voraz con heurístico de movilidad y el algoritmo voraz con heurístico territorial en la Tabla 71.

Algoritmo 1	Algoritmo 2	Resultado
Voraz Territorial	Voraz Movilidad	100/0

Tabla 71 Comparación voraz territorial y voraz movilidad

En esta comparación queda claro que el heurístico territorial supera al de movilidad.

10.3 Minimax

De forma similar a los algoritmos voraces, se han implementado las evaluaciones de movilidad y territorial. Además, se han implementado 2 más.

La evaluación territorial relativa (ver 9.1) es muy similar a la territorial. En la evaluación territorial relativa, la diferencia en movimientos para alcanzar una casilla es significativa. Por ejemplo, si las blancas alcanzan una casilla en 1 movimiento y las negras necesitan 5 movimientos, esta casilla tendrá una mayor puntuación para las blancas que si la misma casilla se alcanzase en 2 movimientos por las negras. Esto contrasta con la evaluación territorial, donde ambos casos contribuirían de la misma forma al territorio de las blancas.

La evaluación territorial - movilidad (ver 9.2) es también similar a la evaluación territorial. La única diferencia es que, además de tener en cuenta el territorio del jugador, se tiene en cuenta la movilidad de sus piezas y las del oponente. Esto se hace evaluando la movilidad individual de cada pieza utilizando las casillas a las que tiene acceso. Además, una pieza tendrá movilidad 0 si está aislada del resto de las piezas del rival. Una pieza aislada no es deseable, sobre todo al principio de la partida. La evaluación territorial - movilidad también tiene en cuenta la etapa de la partida para la movilidad individual de las piezas.

En la Tabla 72 se comparan las diferentes implementaciones del algoritmo Minimax con el algoritmo aleatorio. El algoritmo Minimax se prueba con 1 de profundidad máxima y 3 segundos de parámetro temporal. Esto se hace ya que no se necesita un tiempo muy largo o una gran profundidad para comprobar que el algoritmo Minimax vence en todas las ocasiones al algoritmo aleatorio.

Algoritmo Minimax	Profundidad	Algoritmo 2	Resultado
Minimax Movilidad	2	Aleatorio	100/0
Minimax Movilidad	3	Aleatorio	100/0
Minimax Territorial	2	Aleatorio	100/0
Minimax Territorial	3	Aleatorio	100/0

Tabla 72 Comparación Minimax y aleatorio

En la comparación entre el algoritmo Minimax contra el algoritmo aleatorio se puede ver como para profundidad 2 y 3 las evaluaciones de movilidad y territorial superan en todas las ocasiones al algoritmo aleatorio. Al ver los resultados, no se espera que el algoritmo aleatorio supere en ningún caso a ninguna de las demás implementaciones del algoritmo Minimax.

Ahora se comparan las diferentes implementaciones del algoritmo Minimax con la mejor implementación del algoritmo voraz, la que utiliza evaluación territorial.

Algoritmo Minimax	Profundidad	Algoritmo Voraz	Resultado
Minimax Movilidad	2	Voraz Territorial	88/12
Minimax Movilidad	3	Voraz Territorial	91/9
Minimax Territorial	2	Voraz Territorial	100/0
Minimax Territorial-Movilidad	2	Voraz Territorial	25/25

Tabla 73 Comparación Minimax y voraz

En este caso, el algoritmo Minimax no parece tan dominante contra un algoritmo voraz en comparación con un algoritmo aleatorio. Las implementaciones de Minimax que utilizan evaluación de movilidad o territorial ganan la mayoría de las partidas contra el algoritmo voraz. Se puede observar cómo aumentando la profundidad para la evaluación de movilidad mejora el número de victorias ligeramente. Sin embargo, en la comparación entre el algoritmo Minimax con evaluación territorial-movilidad, el número de victorias de Minimax y del voraz son iguales. Esto contrasta con un porcentaje de victoria del 100% de la evaluación territorial de Minimax frente al algoritmo voraz. Sobre todo, si se ve el resultado de la comparación entre la evaluación territorial y la evaluación territorial-movilidad en la Tabla 74, donde quedan prácticamente con el mismo número de victorias.

Sin embargo, con profundidad 2 ya gana la mayoría de las partidas contra el algoritmo voraz. Además, se puede ver que, aumentando la profundidad, mejora el resultado para el algoritmo Minimax de evaluación de movilidad. Por último, la evaluación territorial supera el 100% de las veces al voraz sin necesidad de utilizar una profundidad mayor a 2.

En la Tabla 74 se comparan diferentes implementaciones del algoritmo Minimax entre sí. Para esta comparación se ha utilizado una profundidad máxima de 2 y un límite

temporal de 10 segundos. Es importante aclarar que el algoritmo Minimax solo comprueba que el límite temporal se ha alcanzado al alcanzar un nivel de profundidad mayor. Esto es debido a que se utiliza una estrategia de profundización iterativa. Entonces es posible que dentro de los 10 segundos se evalúe el árbol con profundidad 1, pero también se empiece a evaluar el árbol con profundidad 2. Si en ese momento ya han pasado 10 segundos, el algoritmo terminará de evaluar el árbol con profundidad 2 antes de finalizar.

En las comparaciones se ve cómo la evaluación de movilidad queda por detrás de las otras dos. También se ve que el desempeño de la evaluación territorial y la evaluación territorial-movilidad son muy similares.

Algoritmo 1	Algoritmo 2	Resultado
Minimax Movilidad	Minimax Territorial	0/100
Minimax Movilidad	Minimax Territorial - Movilidad	0/100
Minimax Territorial	Minimax Territorial Relativo	83/17
Minimax Territorial	Minimax Territorial-Movilidad	49/51

Tabla 74 Comparación evaluaciones Minimax

Por último, en la Tabla 75 se realiza una comparación a lo largo de 50 partidas entre la evaluación territorial de Minimax y la misma evaluación, pero con una tabla histórica para comprobar si esta hace mejora la efectividad de Minimax. Para ello se utiliza la evaluación territorial, que ha probado ser bastante efectiva frente a otras. Se han comparado en iguales condiciones, con un máximo de profundidad de 2 y un tiempo de 5 segundos.

Algoritmo 1	Algoritmo 2	Resultado
Minimax Territorial Tabla	Minimax Territorial	36/14

Tabla 75 Comparación con y sin tabla histórica para Minimax

Como se puede ver en la tabla y en el CSV que contiene los resultados ubicado en `./comparacion_algoritmos/minimax_territory_table_v_minimax_territory`, la tabla histórica ha probado ser útil. No solo ha ganado la mayoría de las partidas, sino que también ha reducido el tiempo que ha tardado el algoritmo en elegir movimientos, de una media de 11,65 segundos a 8,49 segundos.

10.4 MCTS

Utilizando como base el algoritmo de búsqueda en árboles de Monte Carlo (MCTS), se han desarrollado 4 implementaciones diferentes.

La llamada 'MCTS UCB' (ver 3.4) es una implementación clásica del algoritmo MCTS que usa el criterio UCB1 (ver Ecuación 1) (mencionado en este capítulo como UCB por simplicidad).

La llamada 'MCTS UCB expansión reducida' o 'MCTS UCB cut' es idéntica a la anterior salvo por la expansión de los nodos. En esta implementación, durante la expansión de un nodo, no se crean todos sus posibles hijos (como suele ser habitual en MCTS), sino

que se crea un número reducido de hijos. Se decide cuáles crear utilizando la evaluación de movilidad mencionada en el apartado 10.2.

La llamada 'MCTS Epsilon-Greedy' o 'MCTS Epsilon-Voraz' (ver 9.4) es una implementación distinta del algoritmo MCTS. En esta se utiliza la probabilidad para determinar si, durante las iteraciones del algoritmo, se selecciona el que se considera el mejor movimiento o si se selecciona uno de los otros posibles movimientos siguiendo una distribución de probabilidad. De esta forma se busca balancear lo mejor posible la explotación de los mejores movimientos y la exploración de nuevos movimientos.

La llamada 'MCTS Epsilon-Greedy modificado' o 'MCTS Epsilon-Greedy mod' es una modificación de la implementación MCTS Epsilon-Greedy que se diferencia en la forma de calcular el resultado de una simulación de partida. Con MCTS Epsilon-Greedy la partida tiene una puntuación de 1 si acaba en victoria y de 0 si acaba en derrota. Con la modificación propuesta, la puntuación de una partida depende de la ventaja al final de la partida del ganador. No valdría lo mismo una partida en la que se gana con 1 movimiento extra de ventaja que con 10 movimientos extra. Para esto se utiliza la Ecuación 4.

Primero se comparan las diferentes implementaciones del algoritmo de árbol de búsqueda de Monte Carlo con el algoritmo aleatorio. Se le da un tiempo máximo de ejecución de 10 segundos.

Algoritmo MCTS	Simulaciones	Algoritmo 2	Resultado
MCTS UCB	1000	Aleatorio	17/33
MCTS UCB Expansión Reducida	1000	Aleatorio	40/10
MCTS Epsilon-Greedy	1000	Aleatorio	40/10
MCTS Epsilon-Greedy Modificado	1000	Aleatorio	41/9

Tabla 76 Comparación MCTS y aleatorio

Se ve que el algoritmo MCTS no es tan efectivo como otros algoritmos frente al aleatorio. Sobre todo, la implementación base, que llega a perder en la mayoría de las partidas contra el aleatorio. Debido a la complejidad del Juego de las Amazonas y al reducido número de simulaciones es posible que el algoritmo MCTS con UCB se quede sin simulaciones disponibles mucho antes de obtener información significativa de la posición. También puede ser que MCTS dependa de las simulaciones y por lo tanto tenga un componente aleatorio mayor que otros algoritmos.

Después se compara una de las mejores implementaciones del algoritmo de árbol de búsqueda de Monte Carlo con las evaluaciones del algoritmo voraz.

Algoritmo MCTS	Simulaciones	Algoritmo 2	Resultado
MCTS Epsilon-Greedy	1000	Voraz Movilidad	3/47
MCTS Epsilon-Greedy	1000	Voraz Territorial	0/50

Tabla 77 Comparación MCTS y voraz

Se comprueba experimentalmente que el algoritmo MCTS no es tan efectivo como el voraz.

Por último, se comparan las diferentes implementaciones de MCTS entre sí. Para ellas se utiliza un máximo de 2000 simulaciones y 15 segundos.

Algoritmo 1		Algoritmo 2		Resultado
	Simulaciones		Simulaciones	
MCTS UCB	2000	MCTS UCB Expansión Reducida	2000	39/61
MCTS Epsilon-Greedy	2000	MCTS UCB Expansión Reducida	2000	50/50
MCTS Epsilon-Greedy	2000	MCTS Epsilon-Greedy Modificado	2000	60/40

Tabla 78 Comparación implementaciones MCTS

Según los resultados, la implementación modificada del algoritmo MCTS con Epsilon-Greedy no es superior a la no modificada. También parece que el rendimiento del algoritmo MCTS con expansión reducida y el algoritmo MCTS con Epsilon-Greedy son similares, por lo tanto, parece que la principal ventaja de ambos es que no se expanden todos los hijos de un nodo, sino solo los más prometedores.

10.5 Minimax vs MCTS

Por último, se han elegido comparaciones entre implementaciones del algoritmo Minimax e implementaciones del algoritmo de MCTS. Para el algoritmo MCTS se ha elegido un número de simulaciones de 2000 y un tiempo máximo de 10 segundos mientras que para Minimax se establece una profundidad máxima de 2 con un parámetro temporal de 5 segundos. Se juegan 100 partidas.

Algoritmo 1			Algoritmo 2			Resultado
Minimax	Profundidad	Tiempo	MCTS	Simulaciones	Tiempo	
Minimax Movilidad	2	5	MCTS UCB	2000	10	100/0
Minimax Movilidad	2	5	MCTS Epsilon-Greedy	3000	15	99/1
Minimax Territorial	2	5	MCTS Epsilon-Greedy	2000	10	50/0
Minimax Territorial-Movilidad	2	5	MCTS Epsilon-Greedy	2000	10	100/0

Tabla 79 Comparación Minimax y MCTS

Como se puede ver en los resultados, el algoritmo Minimax supera al algoritmo MCTS, incluso comparando una de las mejores implementaciones de MCTS aumentando el tiempo máximo y simulaciones con la peor evaluación de Minimax reduciendo el tiempo de ejecución hasta que ambos tomen el mismo tiempo en realizar un movimiento aproximadamente.

10.6 Comparación Final

Por último, en la Tabla 80 se presenta una tabla comparativa final en la que se muestran algunos de los resultados más significativos. Se toma siempre la implementación con mejores resultados de cada algoritmo y se comparan entre ellas. El resultado de una casilla es el número de victorias del algoritmo en la fila frente al número de victorias del algoritmo en la columna.

Victorias frente a Derrotas	Aleatorio	Voraz (Territorial)	Minimax (Territorial)	MCTS (Epsilon-Greedy)
Aleatorio		0/100	0/100	10/40
Voraz (Territorial)	100/0		0/100	50/0
Minimax (Territorial)	100/0	100/0		100/0
MCTS (Epsilon-Greedy)	40/10	0/50	0/100	

Tabla 80 Comparativa final

De esta tabla se pueden extraer algunas conclusiones interesantes. De forma experimental, el algoritmo aleatorio es el que ha tenido peores resultados al no obtener una mayoría de victorias frente a ningún algoritmo. El algoritmo MCTS es el siguiente en cuanto a resultados ya que solo ha conseguido ganar con una mayoría de partidas al algoritmo aleatorio, perdiendo contra el resto. El siguiente es el voraz, que consigue un 100% de victorias frente al aleatorio y a MCTS. Por último, el que mejores resultados ha obtenido es el algoritmo Minimax, ya que la implementación con evaluación territorial ha ganado en todas las ocasiones al resto de algoritmos desarrollados.

Capítulo 11. Manuales del Sistema

En este capítulo se explicarán por medio de manuales los pasos necesarios para instalar y utilizar el sistema desarrollado. Esto se hará en 4 subapartados que se enfocan en los distintos perfiles que pueden utilizar el sistema. En primer lugar, se mostrará un manual de instalación en el que se detallarán todos los pasos necesarios antes de poder ejecutar el sistema. En segundo lugar, se explicarán los pasos para ejecutar adecuadamente el sistema. En tercer lugar, se presentará el manual de usuario, donde se detalla cómo utilizar el sistema desde el punto de vista del usuario final. En cuarto y último lugar, se muestra el manual del programador, que incluye la información necesaria para que un desarrollador realice modificaciones y mejoras en el sistema.

11.1 Manual de Instalación

En este manual se incluyen todos los pasos de forma que son necesarios para instalar el sistema en un nuevo equipo. Este manual será dividido en 2 partes. Una para un usuario que solo quiera instalar el sistema y otra para un desarrollador que quiera poder ejecutar el código y modificarlo sobre su equipo.

11.1.1 Instalación para Usuario

La instalación para el usuario es muy sencilla y rápida ya que no es necesario descargar dependencias externas. Para utilizar el sistema es necesario descargar el archivo comprimido entregado en este proyecto. Una vez descargado se puede descomprimir con cualquier software de descompresión de archivos, por ejemplo, 7-Zip [75] o WinRAR [76]. Una vez descomprimido, el ejecutable se encuentra en /instalacion/dist. Para poder ejecutarlo se puede consultar el apartado 11.2.1.

11.1.2 Instalación para Desarrollador

La instalación para un desarrollador es relativamente sencilla, pero el sistema tiene varias dependencias que se han de instalar antes de poder ejecutar el sistema.

Python: es el lenguaje de programación utilizado para desarrollar el sistema y su intérprete es necesario para ejecutar el código. Se puede descargar e instalar desde la página oficial de Python [77].

Pygame: es una biblioteca para elementos gráficos enfocada a juegos (ver 7.2.1.4). Es necesario tenerla instalada en el sistema para poder ejecutar y modificar el código. Para descargar la última versión solo será necesario el siguiente comando desde una consola:

```
pip install pygame
```

Una vez instaladas las anteriores referencias se puede importar el proyecto a un IDE que soporte Python o se puede ejecutar mediante consola. Existen muchos IDEs para Python, el desarrollador puede utilizar cualquiera de ellos. No se mostrará cómo instalar cada uno de ellos en este manual debido a la enorme cantidad de IDEs que existen. Se presupone que un desarrollador que vaya a modificar el código tiene conocimientos técnicos y tiene experiencia con IDEs para el desarrollo en Python. Si se quiere ejecutar el código Python desde una consola se puede consultar el apartado 11.2.2.

11.2 Manual de Ejecución

En este manual se muestran los pasos necesarios para ejecutar el sistema ya instalado sobre un equipo. Es importante entender que el sistema tiene dos formas de ejecución, modo gráfico y modo de entrenamiento. Al igual que el manual de instalación, este se divide en 2, una parte para un usuario y otra para un desarrollador.

11.2.1 Ejecución para Usuario

La ejecución del sistema es a través de la ejecución del archivo `game_of_the_amazons_ai.exe`. Bien sea haciendo doble clic en el archivo o por medio de una consola. Si se ejecuta haciendo doble clic sobre el archivo siempre se ejecutará la interfaz gráfica y siempre se utilizará el fichero de configuración por defecto de la interfaz gráfica, `'gui_conf.txt'`.

Para poder realizar una ejecución con mayor control sobre lo que pasa se puede usar la consola. Para ello, en Windows se puede utilizar CMD o PowerShell. Es posible ejecutar alguno de los dos programas, navegar hasta el directorio que contiene el ejecutable y ejecutarlo. Sin embargo, una forma rápida y sencilla de ejecutar el sistema es dirigirse al directorio que contiene el ejecutable y escribir en el explorador de Windows `'cmd'` o `'powershell'`. De esta forma se abrirá una consola que ya está ubicada en el directorio del ejecutable.

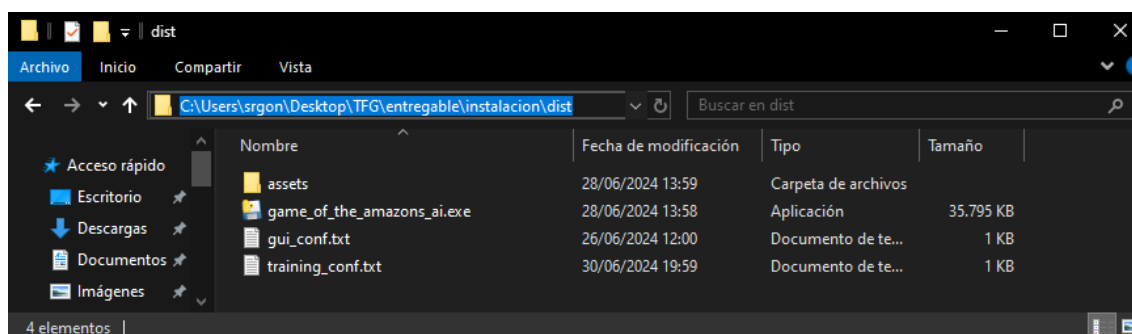


Ilustración 39 Directorio en el explorador de archivos de Windows

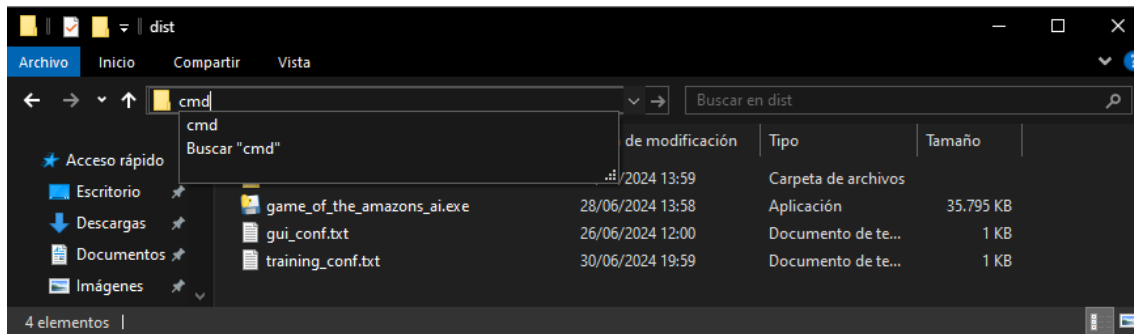


Ilustración 40 Abrir una consola desde el explorador de archivos de Windows

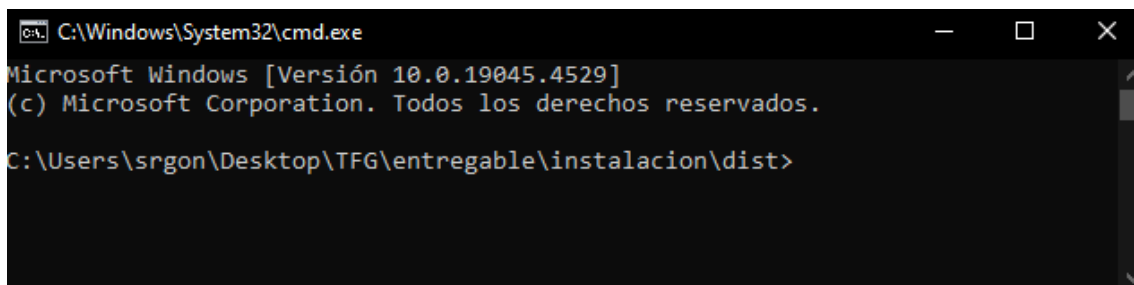


Ilustración 41 Consola abierta desde el explorador de archivos de Windows

Una vez abierta la consola se puede ejecutar el sistema escribiendo un comando. El comando tiene esta estructura:

nombre_ejecutable -opción archivo_configuración

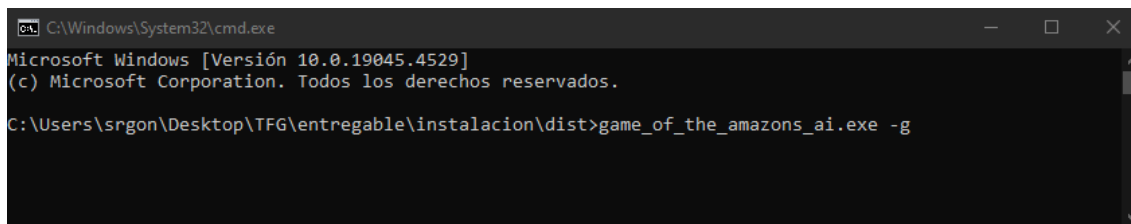
El nombre del ejecutable siempre es `game_of_the_amazons_ai.exe` en caso de estar en el directorio que contiene el ejecutable. Si se empieza a escribir el nombre del archivo y se pulsa el tabulador, este se autocompletará, por lo que no es necesario escribir el nombre completo del archivo.

Las opciones permitidas son 2: interfaz gráfica y entrenamiento. Para ejecutar el sistema en modo gráfico la opción será `-g` o `--graphic` y para la ejecución del entrenamiento será `-t` o `--training`.

Por último, siempre se necesita un fichero de configuración, bien sea para la interfaz gráfica o para el entrenamiento. Este fichero indica las inteligencias artificiales que se pueden seleccionar en la interfaz o las partidas entre inteligencias artificiales que se jugarán durante el entrenamiento. Dentro del directorio donde se encuentra el ejecutable existen dos ficheros predeterminados con estas configuraciones llamados `'gui_conf.txt'` y `'training_conf.txt'`. Si no se indica explícitamente un archivo de configuración distinto, se utilizarán estos archivos.

El formato de los ficheros de configuración se muestra en el apartado 15.2.3.

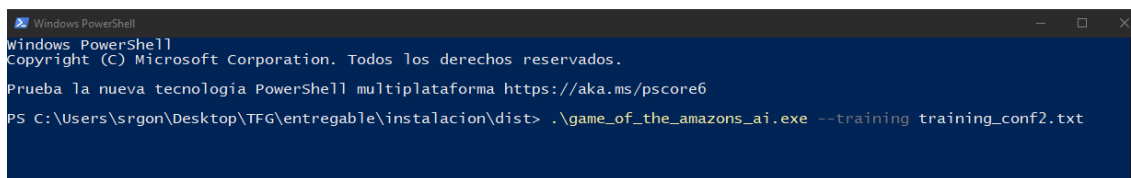
A continuación, se muestran algunos ejemplos de comandos con diferentes opciones para ejecutar el sistema tanto en cmd como en powershell.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.4529]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\srgon\Desktop\TFG\entregable\instalacion\dist>game_of_the_amazons_ai.exe -g
```

Ilustración 42 Ejecución gráfica en cmd con fichero de configuración por defecto



```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/powershell
PS C:\Users\srgon\Desktop\TFG\entregable\instalacion\dist> .\game_of_the_amazons_ai.exe --training training_conf2.txt
```

Ilustración 43 Ejecución entrenamiento en powershell con fichero de configuración personalizado

11.2.2 Ejecución para Desarrollador

Un desarrollador se puede ejecutar el sistema de 2 formas distintas. Se puede importar el proyecto a un IDE y ejecutarlo desde el IDE o se puede ejecutar por consola simplemente teniendo instalado Python y pygame.

El código del proyecto se encuentra en la carpeta /game_of_the_amazons_ai. Si se desea ejecutar por consola se puede utilizar el siguiente comando dentro de la carpeta mencionada anteriormente:

```
python3 main.py [opciones] [archivo_configuración]
```

Una explicación sobre las opciones y el archivo de configuración a utilizar se pueden encontrar en el apartado 11.2.1. En resumen, las opciones son 2: gráfica (-g o --graphic) o entrenamiento (-t o --training). Para usar los archivos de configuración por defecto no es necesario especificar el nombre del archivo, si se quieren crear y usar otros hay que indicarlo explícitamente.

11.3 Manual de Usuario

En este manual se detalla el funcionamiento del sistema para que un usuario pueda entender y utilizar el sistema.

Si se ejecuta el sistema en modo gráfico se presentará una interfaz. Esta está descrita y explicada en el apartado 6.5.

En general, el uso de la interfaz gráfica es muy sencillo. En resumen, antes de la partida se pueden elegir los jugadores. Un jugador para las piezas blancas y otro para las piezas negras. Para seleccionarlas se debe pulsar en uno de los desplegados de la parte derecha de la interfaz y, una vez desplegado, se debe seleccionar un jugador. Si no se selecciona ningún jugador se utilizará el primero de ellos por defecto. De forma análoga

se puede elegir el otro jugador. Una vez seleccionados los jugadores se puede pulsar el botón 'Play' para comenzar la partida.

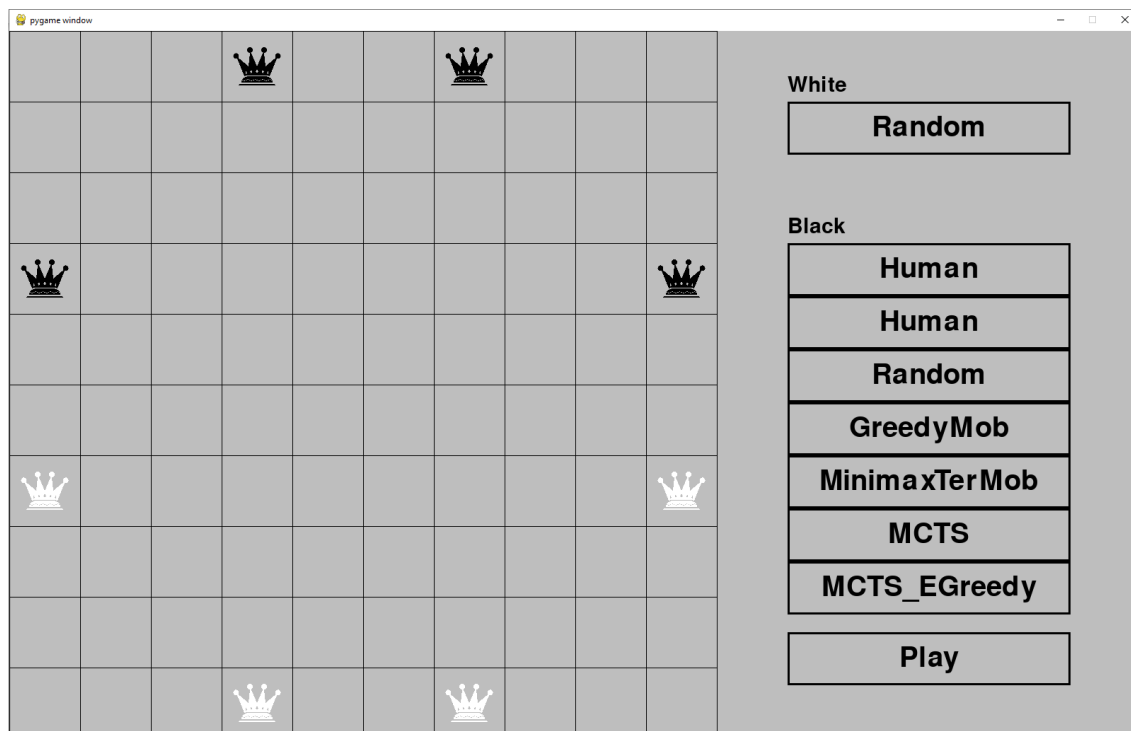


Ilustración 44 Jugador para blancas seleccionado y menú de elección para negras desplegado

En caso de que al menos uno de los jugadores sea un jugador humano, la persona en frente de la interfaz gráfica deberá realizar su movimiento cuando llegue su turno (las piezas blancas tienen el primer turno). Dentro de los documentos entregados se encuentra un archivo 'reglas.txt' que contiene las instrucciones del Juego de las Amazonas que se deberán seguir para realizar un movimiento legal. Durante el turno de un jugador que sea una inteligencia artificial se deberá esperar a que esta realice el movimiento. En función de los parámetros establecidos en el documento de configuración de la interfaz gráfica esto puede tardar más o menos. A continuación, se mostrará cómo se puede realizar un movimiento con la interfaz.

Primero el jugador ha de asegurarse de que es su turno. Si el jugador no tiene claro si es su turno basta con hacer clic sobre una de sus amazonas. Si es el turno del jugador la casilla se marcará con un recuadro más grueso de lo normal y de color rojo. Además, siempre que la amazona tenga movimientos legales, estos se mostrarán en la pantalla. Así se puede asegurar si es el turno del jugador. En la Ilustración 45 se puede ver que es el turno de las blancas. También se puede ver las casillas a las que se puede mover la amazona seleccionada. Mientras que no se mueva ninguna amazona, el jugador puede cambiar la amazona seleccionada. Es importante recordar que una amazona no puede mover a través de otra amazona ni a través de una casilla bloqueada.

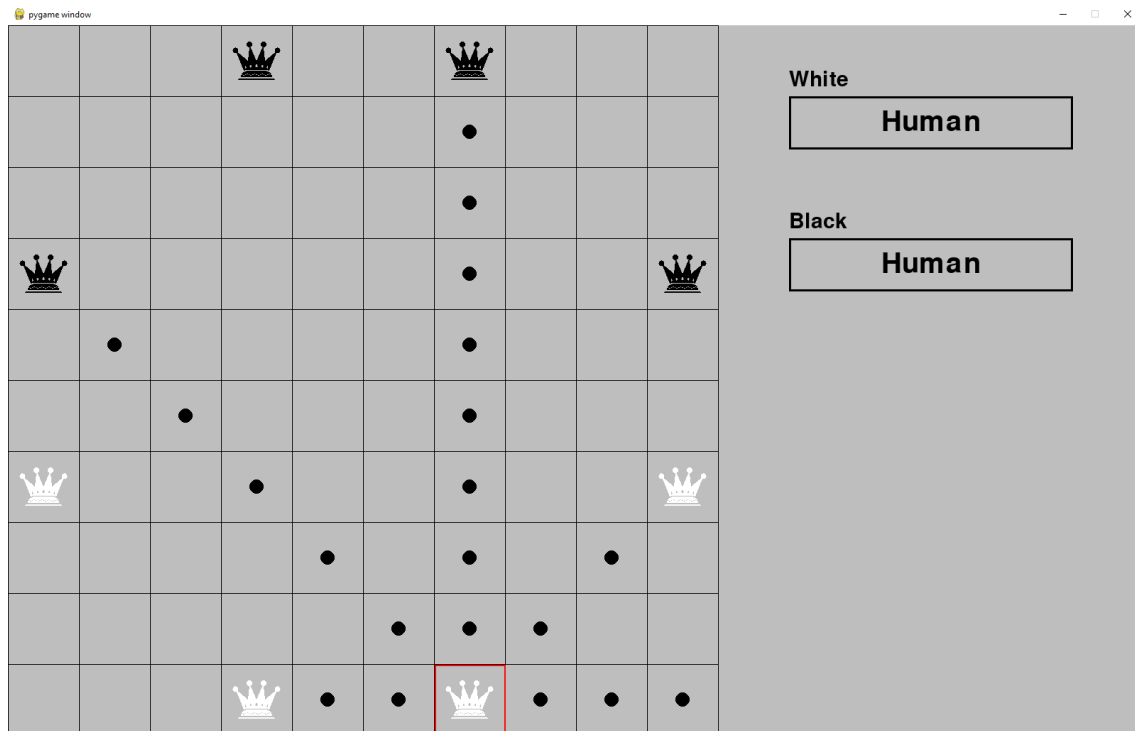


Ilustración 45 Amazona seleccionada

Una vez se mueve la amazona el movimiento no ha terminado. En este punto es necesario 'disparar' a una casilla. Esto quiere decir que se debe hacer clic sobre una de las casillas a las que la amazona tiene acceso. La casilla seleccionada quedará bloqueada durante el resto de la partida. En la Ilustración 46 se puede ver cómo se ha movido una amazona y ahora se puede seleccionar una de las casillas marcadas con un punto para bloquear. Es importante recordar que una amazona no puede disparar a una casilla a través de otra amazona o a través de otra casilla bloqueada.

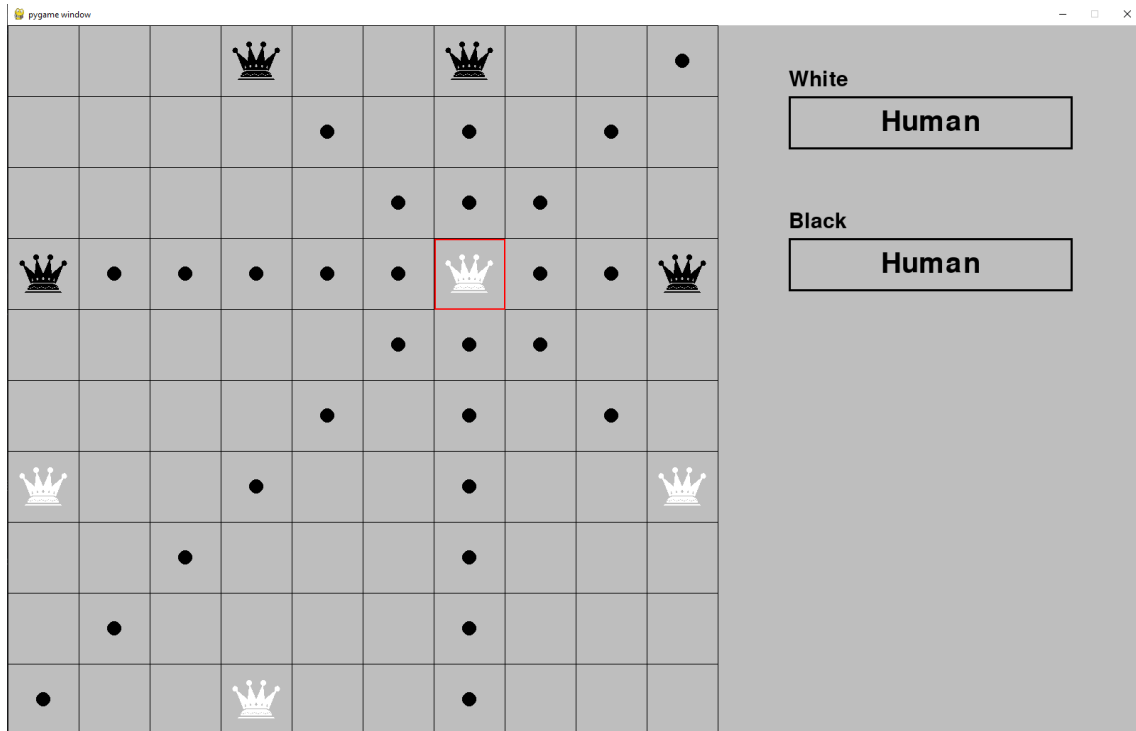


Ilustración 46 Amazona movida

En la Ilustración 47 se puede ver cómo la amazona blanca ha bloqueado una casilla. Esta casilla no podrá ser atravesada durante el resto de la partida.

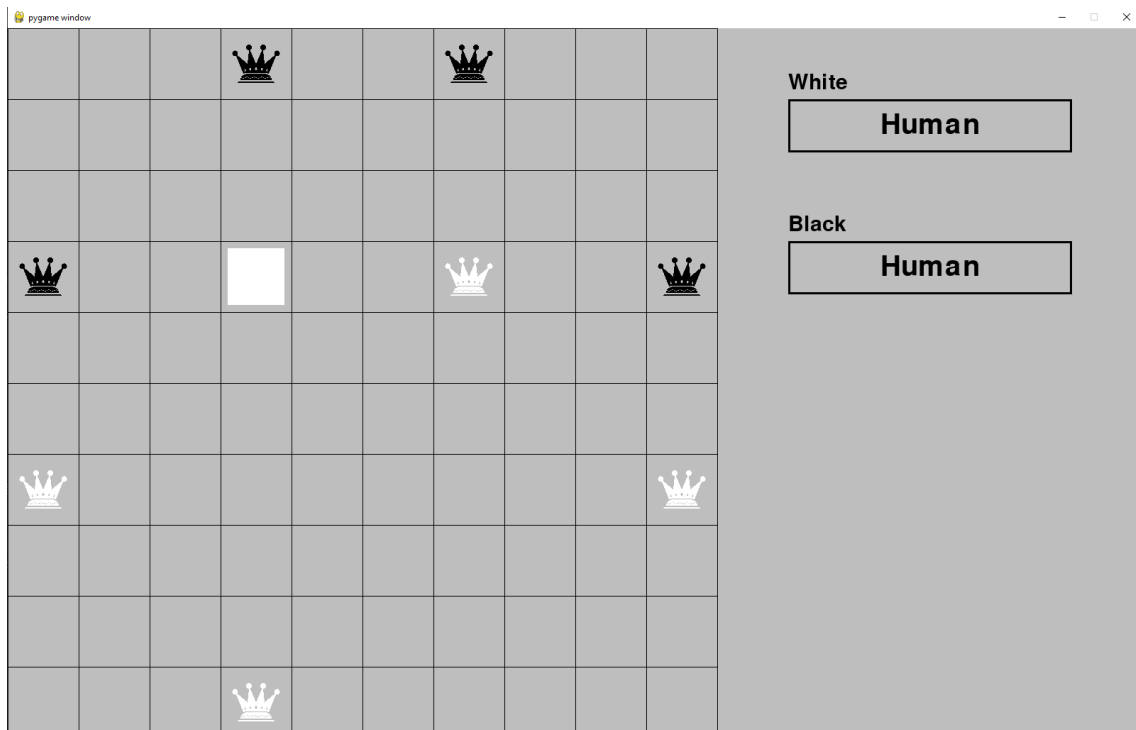


Ilustración 47 Casilla bloqueada

La partida termina cuando uno de los jugadores no tenga movimientos posibles como se puede ver en la Ilustración 48 las blancas no pueden realizar ningún movimiento. Una

vez terminada la partida se pueden volver a seleccionar los jugadores que participarán en la siguiente partida. Una vez se hayan establecido de nuevo los jugadores que participarán, se puede pulsar el botón 'Restart' para empezar una nueva partida como se puede ver en la Ilustración 48.

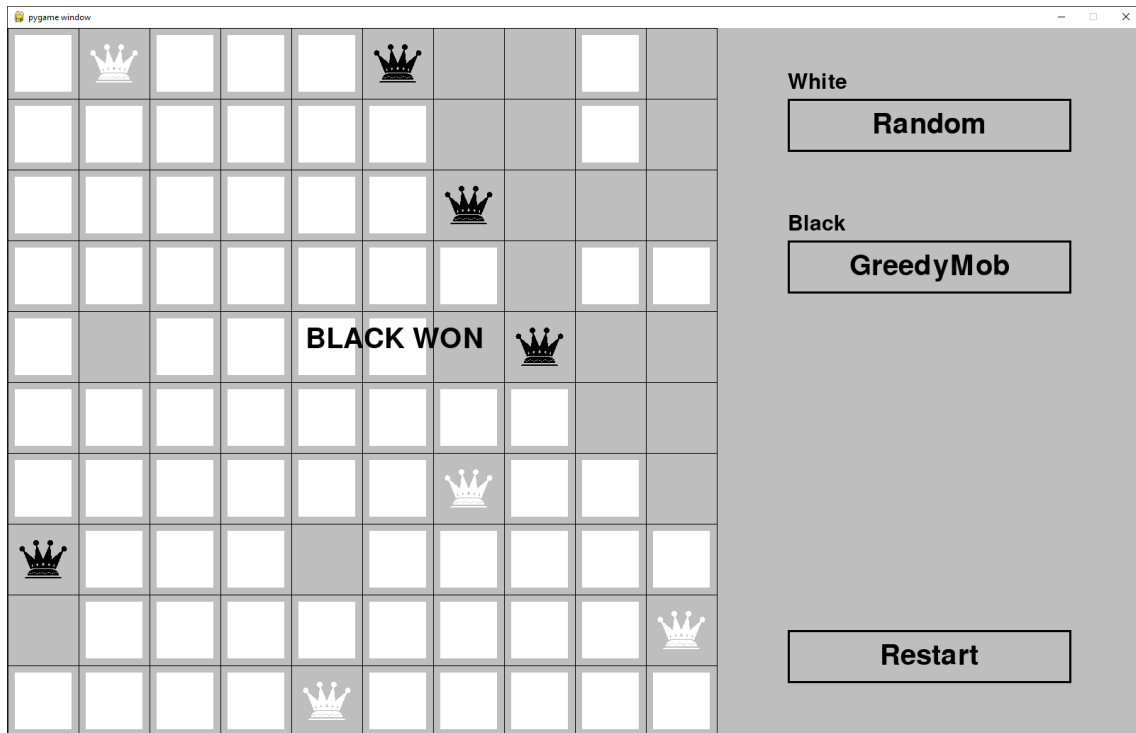


Ilustración 48 Partida terminada

En cualquier momento se puede cerrar el sistema con tan solo pulsar el botón de cerrar la ventana (botón con un símbolo similar a una x) dentro de los botones de control de ventana marcados en la Ilustración 49.

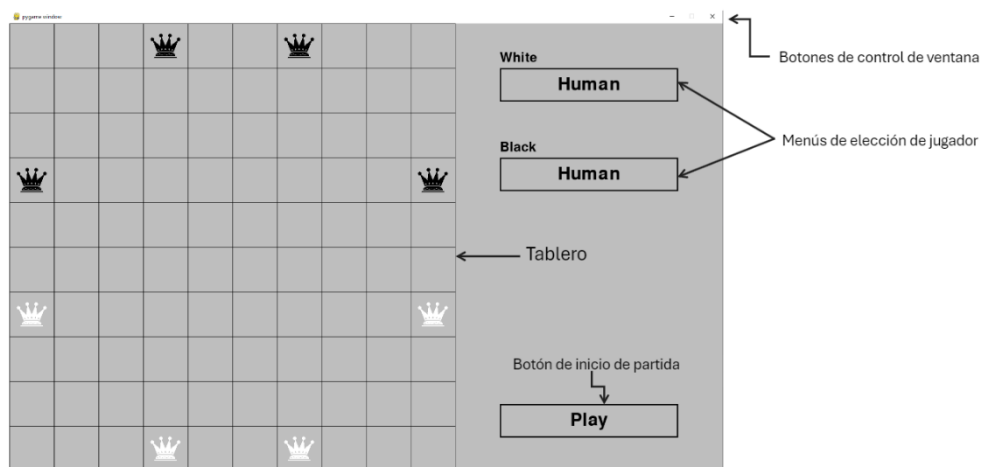


Ilustración 49 Ilustración de los componentes de la ventana y el botón de cerrar

11.4 Manual del Programador

En el manual del programador se incluye la información necesaria para que otros desarrolladores puedan entender cómo funciona internamente el sistema y que sean capaces de modificarlo o incluir nueva funcionalidad.

11.4.1 Paquetes

Como se puede ver en el apartado 6.1.1, el sistema está dividido en varios paquetes. El paquete 'ui' contiene toda la interfaz gráfica. El paquete 'assets' contiene imágenes que usa la interfaz, así como un paquete 'utilities' que contiene utilidades que se usan a lo largo de todo el sistema. El paquete 'amazons' contiene todo lo relacionado con el juego y las inteligencias artificiales. El paquete 'docs' contiene toda la documentación del proyecto. A esta profundidad se encuentran también los 2 ficheros de configuración y el fichero 'main.py'. Este fichero lee los parámetros que se han introducido en la ejecución del sistema y ejecuta el código pertinente.

Dentro del paquete 'amazons' existen varios paquetes. El paquete 'tests' contiene las pruebas unitarias del sistema y el script para entrenar las IAs. El paquete 'players' contiene el jugador humano y el jugador artificial que conecta la interfaz con los algoritmos. El paquete 'logic' contiene la lógica del Juego de las Amazonas. Por último, el paquete 'algorithms' contiene los algoritmos inteligentes implementados en el sistema. Algunos de los algoritmos se han implementado para probar conceptos, pero luego no han sido utilizados ni en la interfaz gráfica ni en el entrenamiento. Estos algoritmos se encuentran en los ficheros 'minimax_algorithm_multi_process.py', 'minimax_algorithm_simple_ordering.py' y 'minimax_algorithm_relative_territory10.py'.

11.4.2 Algoritmos Implementados

Los algoritmos implementados y utilizados en el sistema son 4, sin embargo, 3 de ellos tienen diversas implementaciones que afectan a su eficacia y su comportamiento. Los 4 algoritmos son: aleatorio, voraz, Minimax y árbol de búsqueda de Monte Carlo, se pueden encontrar descripciones más detalladas y sus diferentes implementaciones en las tablas Tabla 81, Tabla 82, Tabla 83 y Tabla 84 respectivamente. El primero solo tiene una implementación, en las tablas para los 3 siguientes se pueden ver las diversas implementaciones.

Algoritmo	Descripción
RandomAlgorithm	Algoritmo que elige movimientos de forma aleatoria

Tabla 81 Algoritmo aleatorio

Algoritmo	Descripción
GreedyAlgorithmMobility	Algoritmo voraz con un heurístico que calcula la movilidad del jugador contrario tras cada posible movimiento, eligiendo el movimiento que minimice el número de movimientos del rival.

GreedyAlgorithmTerritory	Algoritmo voraz con un heurístico que calcula la evaluación territorial tras cada posible movimiento, eligiendo el movimiento que maximice el territorio del jugador y minimice el territorio del rival.
---------------------------------	--

Tabla 82 Algoritmo voraz

Algoritmo	Descripción
MinimaxAlgorithmMobility	Algoritmo Minimax con un heurístico que calcula la movilidad de cada jugador, intentando maximizar la movilidad de un jugador y minimizar la movilidad del contrario.
MinimaxAlgorithmTerritory	Algoritmo Minimax con un heurístico que calcula el territorio de cada jugador, intentando maximizar el territorio de un jugador y minimizar el territorio del contrario. Una casilla pertenece al territorio del que llegue antes.
MinimaxAlgorithmRelativeTerritory	Algoritmo Minimax con un heurístico que calcula el territorio de cada jugador, intentando maximizar el territorio de un jugador y minimizar el territorio del contrario. Cada casilla tiene asociada un entero que representa cuánto antes llega un jugador frente al jugador contrario.
MinimaxAlgorithmRelativeTerritoryMobility	Algoritmo Minimax que combina el heurístico territorial con la movilidad de cada amazona. De esta forma se intenta premiar que al principio de la partida se intente encerrar las piezas rivales y castigar que se encierren las piezas propias.
HistoryTable	Si bien no es un algoritmo, es una forma de mejorar la eficacia de Minimax por medio de la persistencia de evaluaciones de movimientos. De esta forma se pueden guardar evaluaciones para utilizarlas en otras partidas y actualizarlas. Esta estrategia se ha usado para cada uno de los algoritmos de Minimax anteriores.

Tabla 83 Algoritmo Minimax

Algoritmo	Descripción
MCTSAAlgorithmUCB	Algoritmo de búsqueda en árboles de Monte Carlo en su implementación más habitual que utiliza simulaciones y puntuación mediante UCB1.
MCTSAAlgorithmUCBCut	Algoritmo de búsqueda en árboles de Monte Carlo en su implementación más habitual que utiliza simulaciones y puntuación mediante UCB1. La única

	diferencia es que en la expansión de los nodos se expanden solo los 48 más prometedores según una evaluación de movilidad.
MCTSAAlgorithmE	Algoritmo de búsqueda en árboles de Monte Carlo Epsilon voraz. Utiliza distribuciones de probabilidad basadas en los resultados de partidas simuladas (victoria o derrota) y un parámetro épsilon para elegir el mejor movimiento de forma voraz con cierta probabilidad.
MCTSAAlgorithmEMod	Algoritmo de búsqueda en árboles de Monte Carlo Epsilon voraz modificado. Utiliza distribuciones de probabilidad y un parámetro épsilon para elegir el mejor movimiento de forma voraz con cierta probabilidad. Sin embargo, se le otorga una puntuación diferente a los resultados de simulaciones en función de la ventaja final del ganador.

Tabla 84 Algoritmo MCTS

11.4.3 Realizar Modificaciones

Si se desea implementar una nueva inteligencia artificial es tan sencillo como crear una clase, hacer que esta implemente la interfaz Algorithm e implementar el método `make_move`. Una vez creada la nueva inteligencia artificial, se puede introducir en el código del fichero `/game_of_the_amazons_ai/assets/utilities/conf_file_reader.py`, se debe modificar la función `get_algorithm_from_line` para que se obtenga el algoritmo correcto a partir de los ficheros de configuración. Finalmente, se pueden modificar estos ficheros para utilizar la nueva inteligencia artificial.

Si se desea simular partidas para comparar o entrenar una nueva inteligencia artificial, siempre que se haya modificado la función mencionada en el apartado anterior, es sencillo. Solo hay que incluir el texto en el fichero de configuración según el nombre que se ha establecido en la función `get_algorithm_from_line` y los parámetros que tenga.

Si lo que se desea es cambiar la forma de jugar partidas para el entrenamiento o cambiar los datos que se tomen y cómo se guarde la información, se debe modificar el fichero `/game_of_the_amazons_ai/amazons/tests/match_training.py`.

11.4.4 Formato Ficheros de Configuración

El fichero de configuración de la interfaz gráfica contiene los posibles algoritmos que puede seleccionar el usuario para las piezas blancas y negras. El directorio de ejecución ya tiene un fichero de configuración predeterminado para la interfaz, llamado `'gui_conf.txt'`. Si no se indica ningún fichero se utilizará este y también es posible modificarlo. Si se quiere utilizar otro distinto se debe indicar explícitamente. El formato que deberá tener el archivo de configuración de la interfaz es el siguiente:

[algoritmo] [opciones]

[algoritmo] [opciones]

[algoritmo] [opciones]

[algoritmo] [opciones]

[algoritmo] [opciones]

[algoritmo] [opciones]

Es importante tener en cuenta que el máximo de algoritmos a elegir en la interfaz gráfica será como máximo 6. Los posibles algoritmos y sus opciones se muestran en la Tabla 85.

El fichero de configuración del entrenamiento contiene los algoritmos y partidas que se jugarán entre ellos. De forma similar al fichero de configuración de la interfaz, existe un fichero predeterminado llamado 'training_conf.txt' que se utilizará en caso de no indicar uno concreto. El formato del fichero de configuración del entrenamiento es el siguiente:

[algoritmo] [opciones] v [algoritmo] [opciones] : [número de partidas]

[algoritmo] [opciones] v [algoritmo] [opciones] : [número de partidas]

...

En este caso no hay un límite máximo de enfrentamientos entre algoritmos o de número de partidas. Los posibles algoritmos y sus opciones se muestran en la Tabla 85.

Algoritmo	Nombre en el fichero	Opción 1	Opción 2	Opción 3
Humano (sin algoritmo)	human	No tiene	No tiene	No tiene
Aleatorio	random	No tiene	No tiene	No tiene
Voraz (evaluación movilidad)	greedy_mobility	No tiene	No tiene	No tiene
Voraz (evaluación territorial)	greedy_territory	No tiene	No tiene	No tiene
Minimax (evaluación movilidad)	minimax_mobility	Profundidad (entero)	Tiempo (entero)	No tiene
Minimax (evaluación territorial)	minimax_territory	Profundidad (entero)	Tiempo (entero)	No tiene
Minimax (evaluación territorial relativa)	minimax_relative_territory	Profundidad (entero)	Tiempo (entero)	No tiene
Minimax (evaluación movilidad territorial)	minimax_territory_mobility	Profundidad (entero)	Tiempo (entero)	No tiene
Minimax (evaluación movilidad con tabla histórica)	minimax_mobility_table	Profundidad (entero)	Tiempo (entero)	No tiene

Minimax (evaluación territorial con tabla histórica)	minimax_territory_table	Profundidad (entero)	Tiempo (entero)	No tiene
Minimax (evaluación territorial relativa con tabla histórica)	minimax_relative_territory_table	Profundidad (entero)	Tiempo (entero)	No tiene
Minimax (evaluación movilidad - territorial con tabla histórica)	minimax_territory_mobility_table	Profundidad (entero)	Tiempo (entero)	No tiene
MCTS UCB	mcts_ucb	Número de simulaciones (entero)	Tiempo (entero)	Valor de exploración (decimal y opcional)
MCTS UCB recortado	mcts_ucb_cut	Número de simulaciones (entero)	Tiempo (entero)	Valor de exploración (decimal y opcional)
MCTS ϵ-greedy	mcts_epsilon	Número de simulaciones (entero)	Tiempo (entero)	Epsilon (decimal y opcional)
MCTS ϵ-greedy modificado	mcts_epsilon_mod	Número de simulaciones (entero)	Tiempo (entero)	Epsilon (decimal y opcional)

Tabla 85 Guía de nombrado de los algoritmos

Capítulo 12. Conclusiones y Ampliaciones

En este capítulo se presentan las conclusiones y también posibles ampliaciones futuras tras la finalización del proyecto.

12.1 Conclusiones

En este apartado se presentarán las conclusiones. Primero, se dará una vista general del proyecto, destacando los resultados obtenidos y finalmente el autor dará una pequeña reseña personal del proyecto.

Tras la finalización del proyecto se han cumplido los objetivos principales del proyecto. Se ha conseguido implementar diferentes inteligencias artificiales capaces de jugar al Juego de las Amazonas. Durante la realización del proyecto se ha llevado a cabo una investigación sobre artículos científicos relacionados con los objetivos del proyecto. Se han comparado y mejorado para lograr un mejor nivel y extraer información sobre su desempeño. Además, se ha desarrollado una interfaz gráfica que permite al usuario jugar contra inteligencias artificiales o dejar que ellas jueguen sin intervención humana y visualizar las partidas.

El Juego de las Amazonas es computacionalmente muy complejo debido a la gran cantidad de posibles movimientos, sobre todo al principio de una partida. Esto hace que cualquier tipo de poda en un árbol de búsqueda mejore los resultados de forma considerable. Esto se ha percibido experimentalmente a través de todo el desarrollo. El algoritmo Minimax sin poda Alpha-Beta era prácticamente inútil ante un árbol tan ancho. Durante la expansión de los nodos en el árbol de búsqueda de Monte Carlo se ha probado que al expandir solo un número reducido de nodos prometedores se obtienen mejores resultados.

Durante el desarrollo y comparación de los algoritmos se ha visto cómo algunas evaluaciones heurísticas son muy potentes y pueden servir de base para los algoritmos voraces o guiar la búsqueda en Minimax. Este último se ve muy beneficiado por un buen heurístico como se ha visto en los resultados. En cuanto a Minimax las técnicas de ordenación de movimientos como tablas históricas han tenido cierto éxito. Sin embargo, se podrían hacer un número mayor de pruebas y más variadas para entender cómo afectan realmente a la efectividad de Minimax. La evaluación territorial ha probado ser muy útil e incluso permite añadir elementos adicionales que permiten mejorar los resultados, como añadir una evaluación de movilidad individual de las amazonas. De todas formas, las evaluaciones están basadas en conocimiento humano, por lo que tienen limitaciones y llegan hasta cierto límite que no podrán superar. Una buena inteligencia artificial que obtenga conocimiento sin intervención humana puede llegar a superar a cualquier evaluación.

Los algoritmos de árboles de búsqueda de Monte Carlo no han logrado mucho éxito en el juego de las Amazonas. Sin embargo, de forma experimental se ha visto como el método Epsilon-Greedy y la expansión reducida han sido efectivos frente a una implementación clásica de MCTS con UCB1 para el Juego de las Amazonas. La implementación de MCTS con expansión reducida ha ganado 61 de las 100 partidas jugadas contra MCTS con UCB1. Además, MCTS con expansión reducida ha empatado 50 a 50 jugando contra MCTS Epsilon-Greedy, por lo que tienen un rendimiento similar. La modificación propuesta para MCTS con Epsilon-Greedy que evalúa la posición final de una partida simulada no ha tenido más éxito que la versión base de MCTS Epsilon-Greedy, perdiendo 40 a 60. Sin embargo, puede ser interesante considerar la idea de evaluar un final de partida en función a la ventaja final del ganador frente al perdedor en inteligencias artificiales futuras.

El proyecto ha implementado varias inteligencias artificiales presentadas en artículos científicos. Algunas de las inteligencias artificiales desarrolladas como Minimax con la evaluación territorial - movilidad son resultado de pruebas experimentales debido a que no se proveían de implementaciones explícitas en los artículos consultados. Otras son modificaciones que no siguen instrucciones de ningún artículo, como MCTS Epsilon-Greedy modificado. En general se han llegado a resultados experimentales similares durante la implementación de las inteligencias, salvo por el algoritmo MCTS que ha tenido un rendimiento más pobre de lo esperado.

Como valoración personal, este proyecto ha tenido aspectos muy positivos y otros que han podido ser mejores. Empezando por los aspectos positivos, este proyecto me ha enseñado a leer y analizar artículos científicos. También he tenido la oportunidad de aprender sobre los errores que he ido teniendo durante el desarrollo del proyecto y a mejorar el rendimiento de mi código. En cuanto a los aspectos negativos, si bien se han cumplido los objetivos, el sistema presenta un número limitado de implementaciones. Me hubiese gustado desarrollar más inteligencias artificiales y más variadas. Sin embargo, investigar sobre varias versiones diferentes de un mismo algoritmo, desarrollar cada una de ellas y finalmente compararlas entre ellas ha probado ser muy laborioso. Es por eso por lo que la limitación temporal del proyecto no me ha permitido implementar tantas como me hubiese gustado. También me hubiese gustado implementar una interfaz gráfica más elaborada, sin embargo, esto tampoco entraba dentro de los objetivos.

12.2 Ampliaciones

En este apartado se enumerarán y se explicarán en detalle mejoras, cambios o ampliaciones que podrían ser realizadas al sistema en un futuro, así como el motivo o motivos por los que estas ampliaciones no han sido incluidas en el sistema.

12.2.1 Interfaz Gráfica Mejorada

Una de las posibles ampliaciones del sistema es mejorar la interfaz gráfica. Es posible utilizar la lógica y los algoritmos e implementar una interfaz gráfica diferente y más completa, bien sea en Python o en otro lenguaje de programación. También sería

posible convertir el sistema desarrollado en una API (ver 15.1) a la que se le llame desde una interfaz gráfica para obtener los movimientos de los algoritmos. De esta forma se podría implementar juego online con una arquitectura cliente-servidor.

El motivo principal por el que no se ha desarrollada una interfaz gráfica más elaborada o con mayor funcionalidad es que no era el principal objetivo del proyecto. El proyecto busca investigar e implementar diferentes inteligencias artificiales por lo que una mejor interfaz gráfica no entraba dentro del alcance.

12.2.2 Internacionalización de la Interfaz Gráfica

La interfaz gráfica está en inglés y esto puede dificultar el uso del sistema para gente que no lo entienda. Es por esto que puede ser interesante internacionalizar la interfaz y hacerla accesible a un mayor número de usuarios.

Hay 2 motivos principales por los que no se ha llevado a cabo la internacionalización de la interfaz. El primero es que el objetivo del proyecto no se centra en la interfaz gráfica. El segundo es el tiempo limitado de un TFG.

12.2.3 Añadir Nuevas Inteligencias Artificiales

Pese a que se han implementado varias inteligencias artificiales todas ellas son mejoras o modificaciones de 2 modelos de inteligencia artificial, MCTS y Minimax. Es posible implementar en un futuro un mayor número de inteligencias artificiales basadas en otros esquemas. Una de las que más interesantes podría ser es una adaptación de Alpha Zero (ver 3.5) para el Juego de las Amazonas. Hay artículos interesantes que se pueden consultar para facilitar la adaptación de dicha inteligencia artificial como [78], [79] y por supuesto [24], [25], [26] así como un repositorio en GitHub donde se encuentra una adaptación de AlphaZero para Python que se puede encontrar en [80].

El motivo principal por el que no se ha llevado a cabo esta adaptación es el tiempo limitado que tiene un TFG.

12.2.4 Persistencia de Partidas

Otra modificación o ampliación que puede tener sentido es un sistema para guardar las partidas o llevar un registro de estas mientras se están jugando por medio de la interfaz gráfica. También se le podría permitir al usuario recuperar partidas guardadas y seguir jugándolas, de forma similar a lo explicado en 2.3.1.1.3.

De nuevo, no era parte del alcance del proyecto, por lo que no se implementó para el sistema en su versión presentada junto a esta documentación.

Capítulo 13. Planificación del Proyecto y Presupuesto finales

En este capítulo se presentarán la planificación y los presupuestos finales del proyecto. También se comentarán y explicarán las diferencias entre la planificación y los presupuestos finales y los iniciales.

13.1 Planificación Final

La fecha de inicio del proyecto es igual que en la planificación inicial. Sin embargo, el proyecto concluye más tarde de lo que se había planeado inicialmente. De esta forma el inicio es el 23/11/2023 y el final es el 10/07/2024. Los motivos principales por los que el proyecto concluye más tarde de lo previsto son: tener un mayor número de formas de implementar cada algoritmo de lo previsto y de una menor disponibilidad del alumno de lo previsto. Durante la realización del proyecto de media se dedicaron 2 horas diarias durante los días lectivos.

13.1.1 Planificación temporal

De la misma forma que en el apartado 4.1.1, se presenta una tabla con la planificación temporal de las tareas generales y otra tabla con el desglose de cada una de las tareas.

Número de esquema	Nombre de tarea	Duración
1	TFG	341 horas
1.1	Planificación del proyecto	21 horas
1.2	Análisis	44 horas
1.3	Diseño	40 horas
1.4	Desarrollo	119 horas
1.5	Pruebas	42 horas
1.6	Documentación	75 horas

Tabla 86 Tareas principales del proyecto

Número de esquema	Nombre de tarea	Duración
1	TFG	341 horas
1.1	Planificación del proyecto	21 horas
1.1.1	Elección de recursos humanos	5 horas
1.1.2	Estimación de tareas	8 horas
1.1.3	Estimación de costes	8 horas
1.2	Análisis	44 horas

1.2.1	Estudio de alternativas y viabilidad	8 horas
1.2.2	Estudio de artículos científicos	24 horas
1.2.3	Obtención de requisitos	12 horas
1.3	Diseño	40 horas
1.3.1	Diseño de la arquitectura del proyecto	16 horas
1.3.2	Diseño de las clases del proyecto	24 horas
1.4	Desarrollo	119 horas
1.4.1	Desarrollo de la lógica de juego	16 horas
1.4.2	Desarrollo de la interfaz gráfica	24 horas
1.4.3	Desarrollo del algoritmo voraz	12 horas
1.4.3.1	Evaluación movilidad	6 horas
1.4.3.2	Evaluación territorial	6 horas
1.4.4	Desarrollo del algoritmo Minimax	42 horas
1.4.4.1	Algoritmo base	6 horas
1.4.4.2	Poda alpha-beta	6 horas
1.4.4.3	Evaluación movilidad	4 horas
1.4.4.4	Evaluación territorial relativa	8 horas
1.4.4.5	Evaluación territorial	2 horas
1.4.4.6	Evaluación movilidad territorial	10 horas
1.4.4.7	Introducción de tabla histórica	6 horas
1.4.5	Desarrollo del algoritmo de MCTS	25 horas
1.4.5.1	Algoritmo base	6 horas
1.4.5.2	UCB	4 horas
1.4.5.3	UCB reducido	1 hora
1.4.5.4	Epsilon-Greedy	8 horas
1.4.5.5	Epsilon-Greedy modificado	6 horas
1.5	Pruebas	42 horas
1.5.1	Diseño de pruebas	13 horas
1.5.1.1	Unitarias	8 horas
1.5.1.2	Usabilidad	2 horas
1.5.1.3	Accesibilidad	1 hora
1.5.1.4	Rendimiento	2 horas
1.5.2	Desarrollo de pruebas	13 horas
1.5.2.1	Unitarias	6 horas
1.5.2.2	Usabilidad	2 horas
1.5.2.3	Accesibilidad	1 hora
1.5.2.4	Rendimiento	4 horas
1.5.3	Comparación entre los diferentes algoritmos	16 horas
1.6	Documentación	75 horas
1.6.1	Introducción	2 horas
1.6.2	Aspectos teóricos	6 horas
1.6.3	Estudios teóricos	10 horas

1.6.4	Análisis	8 horas
1.6.5	Diseño del sistema	8 horas
1.6.6	Implementación del sistema	8 horas
1.6.7	Desarrollo de las pruebas	6 horas
1.6.8	Manuales del sistema	3 horas
1.6.9	Conclusiones y ampliaciones	6 horas
1.6.10	Planificación del proyecto y presupuestos finales	8 horas
1.6.11	Referencias bibliográficas	2 horas
1.6.12	Apéndices	8 horas

Tabla 87 Desglose de las tareas del proyecto

13.2 Presupuesto Final

De la misma forma que la planificación también se presentará el presupuesto final y se comparará con el inicial. La definición y costes de empresa, así como la facturación de los empleados se mantienen iguales. Lo que cambia son las horas de trabajo de cada empleado debido a duración del proyecto, que ha aumentado en relación con el presupuesto inicial. Las horas han aumentado 4 horas en el análisis, 8 horas en el diseño, 7 horas en el desarrollo y 10 horas en la documentación. Se han reducido 2 horas en las pruebas. En total ha aumentado en 27 horas el proyecto. Lo que resulta en un cambio en el presupuesto final de 447,79 €, 677,28 € incluyendo beneficio (25%) e IVA (21%).

13.2.1 Desarrollo de Presupuesto Detallado (Empresa)

En las siguientes tablas se muestran los costes de cada parte del proyecto en el presupuesto final.

Partida I: Planificación del proyecto									
I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Planificación del proyecto						409,29 €
	01		Elección de recursos humanos					97,45 €	
		01	Jefe del proyecto	5	horas	19,49 €	97,45 €		
	02		Estimación de tareas					155,92 €	
		01	Jefe del proyecto	8	horas	19,49 €	155,92 €		
	03		Estimación de costes					155,92 €	
		01	Jefe del proyecto	8	horas	19,49 €	155,92 €		

Tabla 89 Partida I, planificación del proyecto

Partida II: Análisis									
I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Análisis						643,28 €
	01		Estudio de					116,96 €	

			alternativas y viabilidad						
		01	Analista	8	horas	14,62 €	116,96 €		
	02		Estudio de artículos científicos					350,88 €	
		01	Analista	24	horas	14,62 €	350,88 €		
	03		Obtención de requisitos					175,44 €	
		01	Analista	12	horas	14,62 €	175,44 €		

Tabla 90 Partida II, análisis

Partida III: Diseño									
I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Diseño						584,80 €
	01		Diseño de la arquitectura del sistema					233,92 €	
		01	Analista	16	horas	14,62 €	233,92 €		
	02		Diseño de las clases del proyecto					350,88 €	
		01	Analista	24	horas	14,62 €	350,88 €		

Tabla 91 Partida III, diseño

Partida IV: Desarrollo									
I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Desarrollo						2.136,05 €
	01		Desarrollo de la lógica del juego					287,20 €	
		01	Programador sénior	16	horas	17,95 €	287,20 €		
	02		Desarrollo de la interfaz gráfica					430,80 €	

	01	Programador sénior	24	horas	17,95 €	430,80 €		
	02	Desarrollo del algoritmo voraz					215,40 €	
	01	Programador sénior	12	horas	17,95 €	215,40 €		
	03	Desarrollo del algoritmo minimax					753,90 €	
	01	Programador sénior	42	horas	17,95 €	753,90 €		
	04	Desarrollo del algoritmo de MCTS					448,75 €	
	01	Programador sénior	25	horas	17,95 €	448,75 €		

Tabla 92 Partida IV, desarrollo**Partida V: Pruebas**

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Pruebas						688,80 €
	01		Diseño de pruebas					213,20 €	
		01	Tester	13	horas	16,40 €	213,20 €		
	02		Desarrollo de pruebas					213,20 €	
		01	Tester	13	horas	16,40 €	213,20 €		
	02		Comparación entre los diferentes algoritmos					262,40 €	
		01	Tester	16	horas	16,40 €	262,40 €		

Tabla 93 Partida V, pruebas**Partida VI: Documentación**

I1	I2	I3	Descripción	Cantidad	Unidades	Precio	Subtotal (3)	Subtotal (2)	Total
01			Documentación						1.346,25 €
	01		Introducción					35,90 €	
		01	Programador Senior	2	horas	17,95 €	35,90 €		
	02		Aspectos teóricos					107,70 €	

	01	Programador Senior	6	horas	17,95 €	107,70 €		
03		Estudios teóricos					179,50 €	
	01	Programador Senior	10	horas	17,95 €	179,50 €		
04		Análisis					143,60 €	
	01	Programador Senior	8	horas	17,95 €	143,60 €		
05		Diseño del sistema					143,60 €	
	01	Programador Senior	8	horas	17,95 €	143,60 €		
06		Implementación del sistema					143,60 €	
	01	Programador Senior	8	horas	17,95 €	143,60 €		
07		Desarrollo de las pruebas					107,70 €	
	01	Programador Senior	6	horas	17,95 €	107,70 €		
08		Manuales del sistema					53,85 €	
	01	Programador Senior	3	horas	17,95 €	53,85 €		
09		Conclusiones y ampliaciones					107,70 €	
	01	Programador Senior	6	horas	17,95 €	107,70 €		
10		Planificación del proyecto y presupuestos finales					143,60 €	
	01	Programador Senior	8	horas	17,95 €	143,60 €		
11		Referencias bibliográficas					35,90 €	
	01	Programador Senior	2	horas	17,95 €	35,90 €		
12		Apéndices					143,60 €	
	01	Programador Senior	8	horas	17,95 €	143,60 €		

Tabla 94 Partida VI, documentación

Resumen del presupuesto							
I1	I2	Descripción	Cantidad	Unidades	Precio	Subtotal (2)	Total
01		Proyecto					5.808,47 €
	01	Planificación del proyecto	1	uds.	409,29 €	409,29 €	
	02	Análisis	1	uds.	643,28 €	643,28 €	
	03	Diseño	1	uds.	584,80 €	584,80 €	
	04	Desarrollo	1	uds.	2.136,05 €	2.136,05 €	
	05	Pruebas	1	uds.	688,80 €	688,80 €	
	06	Documentación	1	uds.	1.346,25 €	1.346,25 €	

Tabla 95 Resumen del presupuesto

Capítulo 14. Referencias Bibliográficas

En este capítulo se presentan las referencias consultadas y citadas a lo largo del documento.

14.1 Referencias Consultadas

- [1] “Amazons AI.” Accessed: May 05, 2024. [Online]. Available: <https://www.mindsports.nl/index.php/dagaz/873-amazons-ai>
- [2] “Game of the Amazons.” Accessed: May 05, 2024. [Online]. Available: <https://game-of-amazons.vercel.app/>
- [3] “Comprar Game of the Amazons - Microsoft Store es-ES.” Accessed: May 05, 2024. [Online]. Available: <https://www.microsoft.com/es-es/p/game-of-the-amazons/9p6tdst1mm09?activetab=pivot:overviewtab>
- [4] F. Zehra, M. Javed, D. Khan, and M. Pasha, “Comparative Analysis of C++ and Python in Terms of Memory and Time,” Dec. 2020, doi: 10.20944/PREPRINTS202012.0516.V1.
- [5] “Basic Memory Management in C - Systems Encyclopedia.” Accessed: Jun. 03, 2024. [Online]. Available: <https://systems-encyclopedia.cs.illinois.edu/articles/c-memory-management/>
- [6] “Máquina virtual Java virtual machine - Documentación de IBM.” Accessed: Jun. 03, 2024. [Online]. Available: <https://www.ibm.com/docs/es/i/7.3?topic=platform-java-virtual-machine>
- [7] “Kotlin vs Java Para el Desarrollo Móvil y Web - Kinsta®.” Accessed: Jun. 04, 2024. [Online]. Available: <https://kinsta.com/es/blog/kotlin-vs-java/>
- [8] “Kotlin vs Java: Comparativa en profundidad | OpenWebinars.” Accessed: Jun. 04, 2024. [Online]. Available: <https://openwebinars.net/blog/kotlin-vs-java/>
- [9] “GlobalInterpreterLock - Python Wiki.” Accessed: Jun. 03, 2024. [Online]. Available: <https://wiki.python.org/moin/GlobalInterpreterLock>
- [10] “Amazons - Chessprogramming wiki.” Accessed: May 26, 2024. [Online]. Available: <https://www.chessprogramming.org/Amazons>
- [11] “Game of the Amazons - Wikipedia.” Accessed: Feb. 08, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Game_of_the_Amazons
- [12] “Greedy Algorithms | Brilliant Math & Science Wiki.” Accessed: May 26, 2024. [Online]. Available: <https://brilliant.org/wiki/greedy-algorithm/>

- [13] H. Bi, Z. Yang, and M. Wang, "The Performance of Different Algorithms to Solve Traveling Salesman Problem," *Proceedings - 2021 2nd International Conference on Big Data and Artificial Intelligence and Software Engineering, ICBASE 2021*, pp. 153–156, 2021, doi: 10.1109/ICBASE53849.2021.00036.
- [14] J. Bang-Jensen, G. Gutin, and A. Yeo, "When the greedy algorithm fails," *Discrete Optimization*, vol. 1, no. 2, pp. 121–127, Nov. 2004, doi: 10.1016/J.DISOPT.2004.03.007.
- [15] "Artificial Intelligence | Mini-Max Algorithm - Javatpoint." Accessed: May 26, 2024. [Online]. Available: <https://www.javatpoint.com/mini-max-algorithm-in-ai>
- [16] "Minimax: Juegos con adversario." Accessed: May 26, 2024. [Online]. Available: <https://www.cs.us.es/~fsancho/Blog/posts/Minimax.md>
- [17] "Minimax - Wikipedia, la enciclopedia libre." Accessed: Feb. 10, 2024. [Online]. Available: <https://es.wikipedia.org/wiki/Minimax>
- [18] "Poda alfa-beta - Wikipedia, la enciclopedia libre." Accessed: Jul. 10, 2024. [Online]. Available: https://es.wikipedia.org/wiki/Poda_alfa-beta
- [19] "Monte-Carlo Tree Search", doi: 10.26481/dis.20100930gc.
- [20] N. Metropolis and S. Ulam, "The Monte Carlo Method," *J Am Stat Assoc*, vol. 44, no. 247, pp. 335–341, 1949, doi: 10.1080/01621459.1949.10483310.
- [21] "General Game-Playing With Monte Carlo Tree Search | by Michael Liu | Medium." Accessed: May 26, 2024. [Online]. Available: <https://medium.com/@quasimik/monte-carlo-tree-search-applied-to-letterpress-34f41c86e238>
- [22] "(PDF) Comparison of Selection Strategies in Monte-Carlo Tree Search for Computer Poker." Accessed: May 05, 2024. [Online]. Available: https://www.researchgate.net/publication/228826332_Comparison_of_Selection_Strategies_in_Monte-Carlo_Tree_Search_for_Computer_Poker
- [23] "(4481) AlphaGo - The Movie | Full award-winning documentary - YouTube." Accessed: May 25, 2024. [Online]. Available: <https://www.youtube.com/watch?v=WXuK6gekU1Y>
- [24] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature 2016 529:7587*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, doi: 10.1038/nature16961.
- [25] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature 2017 550:7676*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, doi: 10.1038/nature24270.
- [26] D. Silver *et al.*, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," 2017.

- [27] “Stockfish - Strong open-source chess engine.” Accessed: May 26, 2024. [Online]. Available: <https://stockfishchess.org/>
- [28] “elmo (shogi engine) - Wikipedia.” Accessed: May 26, 2024. [Online]. Available: [https://en.wikipedia.org/wiki/Elmo_\(shogi_engine\)](https://en.wikipedia.org/wiki/Elmo_(shogi_engine))
- [29] P. P. L. M. Hensgens, J. W. H. M. Uiterwijk, E. O. Postma, and I. E. C. D. Van Der Werf, “A Knowledge-based Approach of the Game of Amazons,” 2001.
- [30] T. A. Marsland, “AREVIEW OF GAME-TREE PRUNING †,” *Final draft: ICCA Journal*, vol. 9, no. 1, 1986.
- [31] M. Buro, “Simple amazons endgames and their connection to Hamilton circuits in cubic subgrid graphs”.
- [32] “5th Computer Olympiad - Chessprogramming wiki.” Accessed: May 28, 2024. [Online]. Available: https://www.chessprogramming.org/5th_Computer_Olympiad
- [33] “6th Computer Olympiad - Chessprogramming wiki.” Accessed: May 28, 2024. [Online]. Available: https://www.chessprogramming.org/6th_Computer_Olympiad
- [34] J. Lieberum, “An evaluation function for the game of amazons,” *Theor Comput Sci*, vol. 349, no. 2, pp. 230–244, Dec. 2005, doi: 10.1016/J.TCS.2005.09.048.
- [35] I. Khalil, E. Neuhold, A. M. Tjoa, L. Da Xu, and I. You, Eds., *Comparative Study of Monte-Carlo Tree Search and Alpha-Beta Pruning in Amazons*, vol. 9357. in *Lecture Notes in Computer Science*, vol. 9357. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-24315-3.
- [36] C. Tian, “Monte-Carlo tree search with Epsilon-Greedy for game of amazons,” *Applied and Computational Engineering*, vol. 6, no. 1, pp. 775–780, Jun. 2023, doi: 10.54254/2755-2721/6/20230956.
- [37] N. Hariharan and P. G. Anand, “A Brief Study of Deep Reinforcement Learning with Epsilon-Greedy Exploration,” *International Journal of Computing and Digital Systems*, vol. 11, no. 1, pp. 541–551, 2022, doi: 10.12785/IJCDS/110144.
- [38] “Solving the Multi-Armed Bandit Problem | by Anson Wong | Towards Data Science.” Accessed: Jun. 08, 2024. [Online]. Available: <https://towardsdatascience.com/solving-the-multi-armed-bandit-problem-b72de40db97c>
- [39] “Aprendizaje por refuerzo (RL) — Capítulo 3: Multi-armed bandit— Parte 1: Epsilon-greedy y epsilon-decreasing | by Joan Cerretani | Medium.” Accessed: Jun. 10, 2024. [Online]. Available: <https://medium.com/@joancerretanids/aprendizaje-por-refuerzo-rl-cap%C3%ADtulo-3-multi-armed-bandit-parte-1-epsilon-greedy-y-99aadba10cdf>
- [40] “Epsilon-Greedy Algorithm in Reinforcement Learning - GeeksforGeeks.” Accessed: Jun. 10, 2024. [Online]. Available: <https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/>

- [41] F. Liu, L. Viano, and V. Cevher, “Understanding Deep Neural Function Approximation in Reinforcement Learning via ϵ -Greedy Exploration”.
- [42] “Comprar PyCharm Professional: Precios y licencias, Descuentos - Suscripción a JetBrains Toolbox.” Accessed: May 25, 2024. [Online]. Available: <https://www.jetbrains.com/es-es/pycharm/buy/?section=commercial&billing=yearly>
- [43] “Create estimate: Configure Amazon WorkSpaces.” Accessed: May 25, 2024. [Online]. Available: <https://calculator.aws/#/createCalculator/WorkSpaces>
- [44] “Why Software Design Is Important.” Accessed: Jun. 13, 2024. [Online]. Available: <https://www.computer.org/resources/importance-of-software-design-is-important>
- [45] “pickle — Python object serialization — Python 3.12.4 documentation.” Accessed: Jun. 15, 2024. [Online]. Available: <https://docs.python.org/3/library/pickle.html>
- [46] “5. Data Structures — Python 3.12.4 documentation.” Accessed: Jun. 15, 2024. [Online]. Available: <https://docs.python.org/3/tutorial/datastructures.html#more-on-lists>
- [47] “What is Accessibility Testing? (Examples).” Accessed: Jun. 26, 2024. [Online]. Available: <https://www.guru99.com/accessibility-testing.html#sample-test-cases-accessibility-testing>
- [48] “PEP 8 – Style Guide for Python Code | peps.python.org.” Accessed: Jun. 15, 2024. [Online]. Available: <https://peps.python.org/pep-0008/>
- [49] “About the Unified Modeling Language Specification Version 2.5.1.” Accessed: Jun. 15, 2024. [Online]. Available: <https://www.omg.org/spec/UML/>
- [50] “math — Mathematical functions — Python 3.12.4 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3/library/math.html>
- [51] “time — Time access and conversions — Python 3.12.4 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3/library/time.html>
- [52] “abc — Abstract Base Classes — Python 3.12.4 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3/library/abc.html>
- [53] “Pygame Front Page — pygame v2.6.0 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://www.pygame.org/docs/>
- [54] “threading — Thread-based parallelism — Python 3.12.4 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3/library/threading.html>
- [55] “multiprocessing — Process-based parallelism — Python 3.12.4 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3/library/multiprocessing.html>

- [56] “random — Generate pseudo-random numbers — documentación de Python - 3.12.4.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/es/3/library/random.html>
- [57] “os.path — Common pathname manipulations — Python 3.12.4 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3/library/os.path.html>
- [58] “sys — System-specific parameters and functions — Python 3.12.4 documentation.” Accessed: Jun. 26, 2024. [Online]. Available: <https://docs.python.org/3/library/sys.html>
- [59] “unittest — Unit testing framework — Python 3.12.4 documentation.” Accessed: Jun. 17, 2024. [Online]. Available: <https://docs.python.org/3/library/unittest.html>
- [60] “The Python Profilers — Python 3.12.4 documentation.” Accessed: Jun. 26, 2024. [Online]. Available: <https://docs.python.org/3/library/profile.html#module-cProfile>
- [61] “Sphinx — Sphinx documentation.” Accessed: Jun. 26, 2024. [Online]. Available: <https://www.sphinx-doc.org/en/master/>
- [62] “PyInstaller Manual — PyInstaller 6.9.0 documentation.” Accessed: Jul. 10, 2024. [Online]. Available: <https://pyinstaller.org/en/stable/>
- [63] “collections — Container datatypes — Python 3.12.4 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3/library/collections.html>
- [64] “Cola doblemente terminada - Wikipedia, la enciclopedia libre.” Accessed: Jun. 16, 2024. [Online]. Available: https://es.wikipedia.org/wiki/Cola_doblemente_terminada
- [65] “copy — Shallow and deep copy operations — Python 3.9.19 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3.9/library/copy.html>
- [66] “csv — CSV File Reading and Writing — Python 3.12.4 documentation.” Accessed: Jun. 16, 2024. [Online]. Available: <https://docs.python.org/3/library/csv.html>
- [67] “Compare PyCharm Professional vs. PyCharm Community - JetBrains IDE.” Accessed: Jun. 05, 2024. [Online]. Available: <https://www.jetbrains.com/products/compare/?product=pycharm&product=pycharm-ce>
- [68] “Visual Studio Code - Code Editing. Redefined.” Accessed: Jun. 05, 2024. [Online]. Available: <https://code.visualstudio.com/>
- [69] “draw.io.” Accessed: Jun. 15, 2024. [Online]. Available: <https://app.diagrams.net/>
- [70] “Visual Paradigm - Suite de Productividad Online.” Accessed: Jun. 15, 2024. [Online]. Available: <https://online.visual-paradigm.com/es/>

- [71] "Git." Accessed: Jun. 16, 2024. [Online]. Available: <https://www.git-scm.com/>
- [72] "GitHub." Accessed: Jun. 16, 2024. [Online]. Available: <https://github.com/>
- [73] "Comprehensive Guide to Selecting Git Provider." Accessed: Jun. 16, 2024. [Online]. Available: <https://www.daytona.io/dotfiles/guide-selecting-git-provider>
- [74] "Mendeley Reference Manager | Mendeley." Accessed: Jun. 16, 2024. [Online]. Available: <https://www.mendeley.com/reference-management/reference-manager/>
- [75] "7-Zip." Accessed: Jul. 03, 2024. [Online]. Available: <https://www.7-zip.org/>
- [76] "Soporte WinRAR - Sitio oficial WinRAR en español." Accessed: Jul. 03, 2024. [Online]. Available: <https://www.winrar.es/>
- [77] "Download Python | Python.org." Accessed: Jul. 03, 2024. [Online]. Available: <https://www.python.org/downloads/>
- [78] G. Zhang *et al.*, "Mastering the Game of Amazons Fast by Decoupling Network Learning," in *Proceedings of the International Joint Conference on Neural Networks*, Institute of Electrical and Electronics Engineers Inc., Jul. 2021. doi: 10.1109/IJCNN52387.2021.9534274.
- [79] H. Huang and S. Li, "The application of reinforcement learning in Amazons," *Proceedings - 2019 International Conference on Machine Learning, Big Data and Business Intelligence, MLBDDBI 2019*, pp. 369–372, Nov. 2019, doi: 10.1109/MLBDDBI48998.2019.00083.
- [80] "suragnair/alpha-zero-general: A clean implementation based on AlphaZero for any game in any framework + tutorial + Othello/Gobang/TicTacToe/Connect4 and more." Accessed: Jun. 20, 2024. [Online]. Available: <https://github.com/suragnair/alpha-zero-general>
- [81] "¿Qué es una API y cómo funciona?" Accessed: Jun. 20, 2024. [Online]. Available: <https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces>
- [82] "What Is a Mockup? | Coursera." Accessed: Jun. 01, 2024. [Online]. Available: <https://www.coursera.org/articles/what-is-mockup>
- [83] "Videoconferencia, reuniones, llamadas | Microsoft Teams." Accessed: Jul. 06, 2024. [Online]. Available: <https://www.microsoft.com/es-es/microsoft-teams/group-chat-software>

Capítulo 15. Apéndices

En este capítulo se presentan los apéndices del proyecto que incluyen el glosario y diccionario de datos, el resumen del contenido entregado en el archivo adjunto, el resumen del código fuente y las actas de las reuniones.

15.1 Glosario y Diccionario de Datos

- **API (Application Programming Interface):** conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones [81].
- **Checklist:** lista preparada para llevar control y registrar tareas o acciones a realizar dentro de un proceso para lograr un objetivo de forma sistemática y planeada.
- **Heurístico:** regla o estrategia para encontrar una solución a un problema de forma simple. No garantiza una solución óptima, pero la aproxima.
- **IDE (Integrated Development Environment):** entorno de desarrollo integrado, plataforma de software que facilita el desarrollo de software.
- **Mockup:** visualización o diseño de una aplicación, página web o producto que ilustra cómo podrá ser el resultado final [82].
- **Profiling:** proceso de análisis del rendimiento de un programa o aplicación informática por medio de herramientas software.

15.2 Contenido Entregado en el Archivo Adjunto

En este capítulo se explicará la estructura y contenido del archivo adjunto, es decir, el archivo entregable de este proyecto.

15.2.1 Contenidos

En este apartado se explicará el contenido del entregable del proyecto. Todo el contenido se explica en la Tabla 96. En esta se indican los directorios y archivos contenidos en el entregable.

Directorio	Contenido
./	Es el directorio raíz del archivo adjunto y a partir del cual se referenciarán el resto de los directorios del entregable. Contiene un fichero leeme.txt explicando la estructura del entregable y un fichero reglas.txt que contiene las reglas del Juego de las Amazonas.

./game_of_the_amazons_ai	Contiene toda la estructura de directorios del proyecto para el desarrollo.
./game_of_the_amazons_ai/docs	Contiene toda la documentación del código. Para poder visualizarla correctamente se puede abrir, dentro del directorio mencionado, el archivo <code>_docs/build/html/index.html</code> .
./game_of_the_amazons_ai/ui	Contiene todo el código fuente relacionado con la interfaz gráfica.
./game_of_the_amazons_ai/assets	Contiene 2 carpetas, una que contiene las imágenes que se utilizan en la interfaz gráfica y otra con utilidades para el sistema.
./game_of_the_amazons_ai/assets/images	Contiene las imágenes que se utilizan en la interfaz gráfica.
./game_of_the_amazons_ai/assets/utilities	Contiene código con utilidades para el sistema. Estas son un fichero Python que guarda las partidas en un fichero pkl y otro que lee los ficheros de configuración para obtener los algoritmos.
./game_of_the_amazons_ai/amazons	Contiene todo el código fuente relacionado con el juego de las amazonas y los tests correspondientes.
./game_of_the_amazons_ai/amazons/algorithms	Contiene todos los algoritmos inteligentes implementados en el sistema. En este mismo directorio se encuentra la interfaz Algorithm, el algoritmo aleatorio y las carpetas greedy, minimax y mcts que contienen las diferentes implementaciones de los algoritmos voraz, Minimax y MCTS respectivamente.
./game_of_the_amazons_ai/amazons/logic	Contiene toda la lógica del Juego de las Amazonas.
./game_of_the_amazons_ai/amazons/players	Contiene los jugadores que pueden participar en una partida. Estos son el jugador humano y jugador artificial. El primero obtiene los movimientos del usuario y el segundo simula el comportamiento humano durante un movimiento.

./game_of_the_amazons_ai/amazons/tests	Contiene los tests de los algoritmos y la lógica, así como un script para realizar el entrenamiento de las inteligencias artificiales.
./comparacion_algoritmos	Contiene las carpetas donde se incluyen los resultados de las comparaciones de los algoritmos. Cada subcarpeta contiene la comparación de 2 algoritmos e incluye los csv con los resultados y los ficheros pkl que guardan las partidas.
./instalacion	Ficheros utilizados para la instalación y posterior ejecución del proyecto.
./instalacion/build	Directorio que contiene los documentos necesarios para la ejecución del sistema. No contienen nada relevante para entender o ejecutar el sistema.
./instalacion/dist	Directorio que contiene el ejecutable del proyecto en formato exe junto con los archivos de configuración por defecto y las imágenes utilizadas por la interfaz.
./instalacion/dist/assets/images	Directorio que contiene las imágenes utilizadas por la interfaz gráfica durante la ejecución.
./documentacion	Contiene toda la documentación asociada al proyecto, incluyendo este documento.
./documentacion/diagramas	Directorio que contiene las imágenes y diagramas presentados a lo largo de la documentación.
./documentacion/planificación_presupuesto_inicial	Contiene el fichero xlsx con el presupuesto inicial y el fichero mpp con la planificación inicial.
./documentacion/planificación_presupuesto_final	Contiene el fichero xlsx con el presupuesto final y el fichero mpp con la planificación final.
./documentacion/resultados_test	Contiene los resultados de las pruebas presentadas en la documentación, en concreto las pruebas de rendimiento extendidas en formato txt y una hoja xlsx con las tablas utilizadas para generar los diagramas.

Tabla 96 Contenido del entregable

15.2.2 Código Ejecutable e Instalación

El ejecutable básico del sistema es `/instalacion/dist/game_of_the_amazons_ai.exe`. Este ejecutable está hecho para ser ejecutado en Windows 10 o superior. En el directorio `/instalacion/dist` también se encuentran los ficheros de configuración que serán descritos en el siguiente apartado y también se encuentra el directorio `/instalacion/dist/assets`. Este directorio contiene una carpeta con las imágenes que se muestran en el tablero para representar las piezas y las casillas bloqueadas.

15.2.3 Ficheros de Configuración

Los ficheros de configuración necesarios para poder ejecutar el sistema son 2. Uno para la configuración de la interfaz gráfica, que por defecto se llama `'gui_conf.txt'` y otro para la configuración del entrenamiento, que por defecto se llama `'training_conf.txt'`.

15.3 Código Fuente

El código fuente del proyecto se encuentra en el directorio `/game_of_the_amazons_ai`. El código está dividido en paquetes, descritos en el apartado 6.1.1. Directamente en el directorio `/game_of_the_amazons_ai` se encuentra el fichero `main.py` y los dos ficheros de configuración, uno para la interfaz gráfica y otro para el entrenamiento. Luego existen otros 4 directorios. El directorio `/game_of_the_amazons_ai/ui` contiene el código necesario para la interfaz gráfica del sistema. El directorio `/game_of_the_amazons_ai/docs` contiene toda la documentación del sistema. Para ver de forma sencilla toda la documentación y navegar a través de ella se puede acceder al archivo `/game_of_the_amazons_ai/docs/_build/html/index.html`. El directorio `/game_of_the_amazons_ai/assets` contiene por una parte las imágenes que se pueden ver en la interfaz gráfica y por otra parte utilidades que se usan través de todo el sistema. El directorio `/game_of_the_amazons_ai/amazons` es el más importante y contiene otros 4 directorios.

Dentro de `/game_of_the_amazons_ai/amazons` se encuentra el directorio `/game_of_the_amazons_ai/amazons/logic` que contiene toda la lógica del sistema. El directorio `/game_of_the_amazons_ai/amazons/players` contiene el jugador humano y el artificial que se encargan de obtener los movimientos de los algoritmos. El directorio `/game_of_the_amazons_ai/amazons/algorithms` contiene los propios algoritmos desarrollados y utilizados en el sistema. Por último, el directorio `/game_of_the_amazons_ai/amazons/tests` contiene las pruebas unitarias y el código encargado de realizar la comparación de las inteligencias artificiales.

15.4 Actas de reuniones

A continuación, se presentarán las actas de cada reunión llevada a cabo por el tutor del TFG y el alumno durante su elaboración. Todas las reuniones han sido realizadas de forma telemática por medio de la aplicación Microsoft Teams [83]. El acta de cada

reunión incluye la fecha y un pequeño resumen de los aspectos tratados durante la misma.

15.4.1 Acta Reunión 1

Fecha: 23/11/2023

Resumen:

Se ha explicado el proceso de elaboración del trabajo fin de grado.

Se han presentado herramientas interesantes al alumno para la realización del proyecto.

Se han compartido TFGs de otros años con el alumno a modo de ayuda.

Se han explicado tareas que tiene que hacer el alumno:

- Crear un github para el proyecto

- Crear la documentación

- Compartir la documentación con el tutor

15.4.2 Acta Reunión 2

Fecha: 14/12/2023

Resumen:

Se han respondido dudas del alumno sobre la documentación.

Se han respondido dudas del alumno sobre el orden de realización de tareas dentro del proyecto.

15.4.3 Acta Reunión 3

Fecha: 18/01/2024

Resumen:

Se ha comentado el formato de la planificación inicial.

Se han comentado las herramientas más adecuadas para realizar la planificación inicial y el diagrama de GANTT.

15.4.4 Acta Reunión 4

Fecha: 01/02/2024

Resumen:

Se ha comentado cómo se debería estructurar la interfaz gráfica.

Se empezarán a redactar los requisitos.

15.4.5 Acta Reunión 5

Fecha: 16/02/2024

Resumen:

Se han comentado diferentes problemas con la duración del algoritmo Minimax.

Se ha comentado la posibilidad de implementar diferentes evaluaciones.

Se han comentado diferentes técnicas para ordenar los movimientos de Minimax.

El alumno ha preguntado cómo se debería realizar el entrenamiento, barajando dos opciones distintas:

Hacerlo desde la interfaz gráfica

Hacerlo sin interfaz gráfica

Se ha decidido realizar el entrenamiento sin interfaz gráfica para reducir el tiempo de entrenamiento.

Se ha comentado la posibilidad de utilizar una GPU o hilos de ejecución.

15.4.6 Acta Reunión 6

Fecha: 29/02/2024

Resumen:

Se han comentado aspectos acerca de la documentación como qué introducir en el abstract y en las keywords.

Se han comentado los contenidos que deberían aparecer en los aspectos teóricos.

15.4.7 Acta Reunión 7

Fecha: 14/03/2024

Resumen:

Se ha presentado una primera implementación de la interfaz gráfica.

Se ha dado acceso al alumno a una máquina virtual donde realizar el entrenamiento de los algoritmos.

15.4.8 Acta Reunión 8

Fecha: 11/04/2024

Resumen:

Se han decidido los algoritmos voraces a utilizar.

Se han comentado qué tipo de entrenamiento y comparación de algoritmos se puede realizar.

15.4.9 Acta Reunión 9

Fecha: 24/04/2024

Resumen:

Se han empezado a ejecutar partidas en la máquina virtual para comparar diferentes implementaciones y elegir los mejores parámetros para las inteligencias artificiales.

15.4.10 Acta Reunión 10

Fecha: 09/05/2024

Resumen:

Se han comentado pros y contras de diferentes evaluaciones y técnicas para mejorar el rendimiento de los algoritmos de Minimax.

Se ha decidido qué evaluaciones implementar sobre el algoritmo Minimax en el sistema.

15.4.11 Acta Reunión 11

Fecha: 23/05/2024

Resumen:

Se han comentado apartados de documentación que hay que adaptar, como añadir un apartado sobre artículos científicos consultados y modificar el apartado de SGBD por un apartado sobre la persistencia del sistema.

15.4.12 Acta Reunión 12

Fecha: 06/06/2024

Resumen:

Se han comentado pequeños cambios que se han realizado sobre la interfaz gráfica.

Se han discutido problemas encontrados en diferentes artículos científicos.

Se han comentado correcciones realizadas sobre los algoritmos Minimax y MCTS.

15.4.13 Acta Reunión 13

Fecha: 18/06/2024

Resumen:

Se ha decidido reducir el alcance y no adaptar Alpha Zero.

Se han comentado cambios y posibles mejoras en la documentación.

Se ha decidido incluir el diagrama de secuencia para el caso de uso de entrenar algoritmos.

Se han comentado posibles formas de realizar las pruebas de accesibilidad y usabilidad.

15.4.14 Acta Reunión 14

Fecha: 04/07/2024

Resumen:

Se han comentado correcciones a la documentación.

Se ha comentado el contenido del entregable.

Se ha establecido como fecha de la última corrección el lunes 08/07/2024.

El tutor ha explicado los siguientes pasos a seguir tras la entrega, la forma y formato de la entrega y los márgenes de tiempo para realizar la presentación.