



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE COMPUTADORES

Ajedrez Cuántico y su evaluación usando Stockfish

Realizado por:

Alejandro Martín Auden

Dirigido por:

Prof. Dr. José Ramón Portillo Fernández

Sevilla, Junio de 2023



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DE COMPUTADORES

TRABAJO FIN DE GRADO
CURSO ACADÉMICO [2022-2023]

RESUMEN:

En este proyecto vamos a abordar características de la física cuántica y cómo atraer estos conceptos a la población. Estudiaremos el ajedrez cuántico, su historia, sus características y principales contribuyentes en la materia. Podremos explorar una solución con un profundo planteamiento, desarrollo e implementación, y seremos capaces de recrear el proceso de creación de un proyecto informático en el que programaremos un ajedrez cuántico. Encontraremos los problemas y dificultades que siempre aparecen en el camino del desarrollo, junto con las soluciones implementadas. También usaremos las herramientas más punteras en el cálculo ajedrecístico para terminar obteniendo nuestras propias conclusiones sobre los diferentes elementos cuánticos. Entenderemos otras alternativas, otros algoritmos capaces de ofrecernos otra visión sobre dichas evaluaciones. Y podremos consultar una guía para el uso del software que hemos desarrollado.

En nuestro proyecto, al final, encontraremos un programa para jugar al ajedrez cuántico de libre uso y posibles métodos de evaluación de elementos cuánticos basados en Stockfish, el motor de ajedrez más potente del mercado.

TÉRMINOS CLAVE:

Ajedrez cuántico; Stockfish; Aplicaciones didácticas en la cuántica;

ÍNDICE

1	INTRODUCCIÓN, CONTEXTO Y MOTIVACIÓN	7
2	OBJETIVOS DEL PROYECTO	11
2.1	ALCANCE DEL PROYECTO Y DEL PRODUCTO	12
2.1.1	Requisitos mínimos funcionales	12
2.1.2	Otros objetivos dentro del alcance del proyecto	13
3	ANÁLISIS DE REQUISITOS	14
4	ANÁLISIS DE ANTECEDENTES Y APORTACIÓN REALIZADA	16
5	NORMAS DEL AJEDREZ CUÁNTICO	19
5.1	REGLAS CUÁNTICAS AÑADIDAS	19
6	ANÁLISIS TEMPORAL Y DE COSTES	22
7	DISEÑO E IMPLEMENTACIÓN	26
7.1	Preparación	26
7.2	PRIMERA ITERACIÓN	28
7.3	SEGUNDA ITERACIÓN	31
7.4	TERCERA ITERACIÓN	34
7.5	CUARTA ITERACIÓN	41
7.6	QUINTA ITERACIÓN	43
8	PRUEBAS	46
9	EVALUANDO CON STOCKFISH	51
9.1	Un método para evaluar estados cuánticos:	53

9.2	Las infinitas soluciones de las posiciones cuánticas.....	55
10	MANUAL DE USUARIO.....	58
11	BIBLIOGRAFÍA.....	63

Relación de Figuras

Figura 1: Movimientos clásicos	19
Figura 2: Estructura de Desglose de Trabajo	22
Figura 3: Diagrama de Gantt, Análisis temporal de tareas	23
Figura 4: Diagrama de clases	27
Figura 5: Resultado de mover cuánticamente dos veces una misma pieza	30
Figura 6: Primera representación visual del tablero	33
Figura 7: Árbol binario de posiciones. Idea original	35
Figura 8: Árbol binario de posiciones. Posiciones abstractas	37
Figura 9: Algoritmo para la definición de posiciones cuánticas	38
Figura 10: Unión de estados, método para posiciones abstractas	38
Figura 11: Unión de estados, método para posiciones cuánticas	39
Figura 12: Representación definitiva del tablero	40
Figura 13: Algoritmo para la evaluación de una posición	44
Figura 14: Test Jugada cuántica y definición	46
Figura 15: Definición de una pieza con más de dos superposiciones	47
Figura 16: Definición de una pieza entrelazada cuánticamente. Opción de captura A	47
Figura 17: Definición de una pieza entrelazada cuánticamente. Opción de captura B	47
Figura 18: Movimientos complementarios, el enroque y la captura al paso	48
Figura 19: Ejemplo de finalización de partida	48
Figura 20: Dos damas resultantes de movimientos cuánticos	50
Figura 21: Ejemplo evaluación	52
Figura 22: Movimiento cuántico y propuesta de valoración	54
Figura 23: Problemas de la evaluación	55
Figura 24: Pantalla de inicio. Selección de oponente	58
Figura 25: Inicio del juego	59
Figura 26: Comando HELP	59
Figura 27: Comando RULES	60
Figura 28: Comando HINT	61

Relación de Tablas

Cuadro 1: Análisis temporal de tareas	23
---	----

1 INTRODUCCIÓN, CONTEXTO Y MOTIVACIÓN

No somos conscientes del impacto que tiene hoy día la mecánica cuántica en nuestras vidas. Sin embargo, la comprensión de estos fenómenos que pasan desapercibidos en nuestro día a día no está al alcance de todos. En la vida de un amplio porcentaje de la población, la cuántica es solo una palabra que suena lejana y no ha ocupado un puesto de relevancia en la educación de muchos. Sin embargo, no es descabellado pensar que esto puede cambiar en un breve lapso de tiempo. Si bien es cierto que no es objetivo de este proyecto hacer que la física cuántica se entienda en profundidad, sí que lo es poder traer algunas de las características más especiales y poco intuitivas. Y divulgar estos fenómenos de modo que puedan ser reconocidas por un mayor espectro de la población. Pero, ¿por qué es este objetivo tan necesario en la sociedad actual?

Nos encontramos a las puertas de lo que podría ser la nueva revolución, con energías renovables más eficientes, tecnologías capaces de descentralizar instituciones, máquinas que piensan como personas y, el tema que aquí nos concierne, una física cuántica que sería capaz de reinventar la computación tal y como la conocemos.

Es necesario para que la humanidad evolucione que lo haga mediante pasos firmes y uniformes. Es contraproducente desarrollar hiper tecnologías si el usuario medio no es capaz de sacar ventaja de estas mismas. No es disparatado decir que una sociedad bien educada será más próspera y más capaz de adaptarse.

Hoy día ese es el reto al que muchos mayores se enfrentan cuando observan esta sociedad digital. Es el objetivo que muchas plataformas contra la brecha digital promueven, con eslóganes tan conocidos como “Soy mayor, pero no idiota”.

Para prevenir la brecha intergeneracional, la educación es la herramienta más poderosa. Debemos educarnos a nosotros mismos como seres siempre dispuestos a aprender más y a adaptarnos a los cambios que son tan necesarios en nuestra vida.

En la educación cabe destacar el ajedrez como potenciador de la capacidad mental. Sus numerosos beneficios son innegables, y es usado en multitud de ambientes como ejercicio para prevenir diversas enfermedades relacionadas con el envejecimiento de la mente.

El ajedrez es un juego que, en sus cientos de años de historia, ha evolucionado junto con sus jugadores para dar paso a formas más novedosas y llamativas de practicar este deporte casi infinito. En su más reciente historia, hemos podido contemplar los avances

milagrosos que la computación nos ha proporcionado, cambiando la forma de la que esta disciplina se practica y se mejora.

Pero hubo una época en la que, el hecho de que una máquina fuera capaz de superar a un humano en un juego de estrategia, parecía tan solo insospechable. Fue con la llegada de Deep Blue (1996) que la computación pudo cambiar las mentes escépticas de una mayoría (Computer Chess, 2023).

Deep Blue fue una supercomputadora desarrollada por el gigante americano IBM para enfrentar al vigente campeón del mundo, Gary Kaspárov. La capacidad de computación de este superordenador fue una de las más potentes del mundo en la época. En el encuentro programado a 6 partidas que tuvo comienzo el 10 de febrero de 1996, una máquina fue capaz por primera vez de derrotar en una partida lenta a un vigente campeón del mundo. Sin embargo, en el total de las 6 partidas, Kaspárov se impuso por 4 a 2 venciendo 3 partidas y obteniendo 2 tablas.

No fue hasta el año siguiente en el que una versión mejorada de Deep Blue, Deeper Blue, pudo tomar la revancha y vencer por primera vez en un encuentro al ritmo de un torneo profesional, al vigente campeón del mundo (por 3,5 a 2,5).

En los últimos años, adicionalmente, dicha disciplina ha recibido una importante suma de practicantes y de popularidad debido, en gran medida a las limitaciones en cuanto a la experiencia del entretenimiento durante la pandemia y al creciente flujo de contenido en la red con relación a este tema.

Stockfish (Wikipedia, 2023) es un motor de ajedrez de código abierto, líder entre los motores ajedrecísticos del momento. Creado como idea de Tord Romstad y desarrollado por Marco Costalba en su versión 1.0, hoy en día se caracteriza por su profundidad de cálculo gracias a la exhaustiva purga de movimientos en la evaluación a altas profundidades. En su estudio nos basaremos para diagnosticar evaluaciones a la superposición de elementos clásicos como los movimientos o las posiciones.

El ajedrez cuántico nace como idea del profesor Selim Akl (2010), director de la Queen's School of Computing en Queen's University, Canadá. Entre sus investigaciones destacan varios artículos cuyos temas se centran en la computación en paralelo y los algoritmos computacionales. Ha realizado también investigaciones relacionadas con el mundo cuántico entre las que destaca el artículo "On the importance of being quantum".

En dicho artículo propone diferentes variantes al ajedrez clásico en las que se aplican características típicas de la mecánica cuántica. El principal objetivo de la investigación era divulgativo, pero también venía motivada por los recientes logros de las máquinas en los juegos de estrategia. Hablamos de Deep Blue y sus partidas contra Kaspárov. Akl fue inspirado ante la resignación de un compañero al ver que las máquinas "ya

habían resuelto el ajedrez”. Debería centrarse en resolver otros juegos como el “piedra papel o tijeras”. Así lo describe en “The Quantum Chess Story” (Selim G. Akl, 2016). Y añade que esa broma formulada por un compañero le hizo ver que añadir un elemento aleatorio daría una nueva vuelta de tuerca al juego.

Otra versión del ajedrez cuántico es la diseñada por Christopher Cantwell (2017). En su artículo “Quantum Chess: Making Quantum Phenomena Accessible” ofrece una solución distinta, pero aplicando todavía los mismos aspectos de la mecánica cuántica. Su posterior investigación, “Quantum Chess: Developing a Mathematical Framework and Design Methodology for Creating Quantum Games” (Christopher Cantwell, 2019) ofrece una posible descripción del tablero y una serie de modelos que ofrecen una solución matemática a, por ejemplo, el estado del tablero o los movimientos de las piezas.

Christopher Cantwell es un graduado en ingeniería electrónica y en física, y máster en física por la University of Southern California. Creador de la versión más aceptada de ajedrez cuántico y CEO y fundador de Quantum Realm Games. Esta empresa además colabora mucho en la difusión del fenómeno mediante varias campañas, destacando el corto cinematográfico *Paul Rudd explores the Quantum Realm with Stephen Hawking* (2016).

La física cuántica como inspiración a todos estos investigadores es la unión de muchas particularidades, descubrimientos y principios que han dado lugar a la física tal y como la conocemos.

Esta inspiración la podemos resumir en los principios más básicos de la física cuántica. Principios sencillos pero esenciales para el entendimiento del trabajo. Para entender la física cuántica hacemos primero referencia a la metáfora del científico Erwin Schrödinger (1887-1961). En este famoso símil, el físico austríaco compara a los elementos cuánticos con un experimento mental. En este, existe un hipotético gato encerrado dentro de una caja. En esa caja se encuentra también un matraz con veneno que, de ser liberado, mataría al gato. Según Schrödinger, si un evento subatómico aleatorio pudiera tener una probabilidad de romper el matraz matando al gato, y otra probabilidad de no romperlo manteniéndolo con vida, entonces el gato se encuentra en un estado de superposición, vivo y muerto a la vez.

Estos estados se encuentran superpuestos hasta que interactúan con el mundo exterior, o son observados desde el mundo exterior. Del mismo modo, la física dice que un electrón existe al mismo tiempo en todos sus estados posibles.

La física cuántica describe comportamientos como éste, que escapan al entendimiento de la gran mayoría de la población. Sobre todo, porque en su mayoría tan solo describe cálculos probabilísticos.

Otro gran hito para el progreso de la física y también muy relacionado a la metáfora de Schrödinger fue el principio de incertidumbre.

La Relación de indeterminación de Heisenberg, comúnmente llamada principio de incertidumbre, establece que ciertos pares magnitudes observables y complementarias no puedan ser medidos y conocidos con precisión arbitraria. Afirma basándose en este principio que no es posible conocer al mismo tiempo la posición y el momento lineal (movimiento) de una partícula. También se conoce que estas experimentaciones no son debidas a la imprecisión de los medios utilizados o a nuestra incapacidad de realizar medidas a esas escalas, sino a la superposición de estados existentes.

También conocemos la posibilidad del entrelazamiento cuántico. Pensemos en el gato de Schrödinger como otra partícula cuántica, por ejemplo, una partícula M (matraz) y una partícula G (gato). El estado de la partícula M es indeterminado, no lo hemos medido todavía. Pero sabemos que hay un estado para M en el que G se ve afectado. La medición de cualquiera de las dos partículas afectará al estado del otro. Ambos deben estar siempre ligados, pues no existe una posibilidad en la que el matraz haya derramado el componente químico y el gato siga vivo.

Estos conceptos y características abren un mundo de posibilidades, aunque la computación compite por ser uno de los campos que se pueden llegar a beneficiar más de todos los avances de las últimas décadas. La cuántica está en el objetivo de investigadores, de científicos y de educadores que saben que el campo traerá grandes novedades en el futuro próximo

2 OBJETIVOS DEL PROYECTO

El objetivo principal de este proyecto se concentra en traer las características esenciales que observamos en la física cuántica y las plasma bajo las normas de un juego. Con ello seremos capaces de entender aspectos de la física cuántica exponiendo fenómenos de forma observable.

Adicionalmente, el sistema hará uso del motor de ajedrez de libre uso más potente del mercado, Stockfish. Su uso consistirá en convertir parámetros como objetos indeterminados o cuantificados (afectados por las variaciones y las normas de la física cuántica) y tratarlos de forma que el motor sea capaz de interpretarlos a través de las normas del ajedrez clásico. Todo ello con el objetivo de tratar de nuevo los resultados obtenidos para obtener resultados indeterminados o cuantificados.

Dicho de otro modo, el presente proyecto pretende crear una interfaz entre el motor de Stockfish y las normas cuánticas.

Los juegos cuánticos tienen como característica común incluir movimientos que indeterminan la posición de las piezas. Es decir, colocar una pieza en dos o más casillas, indeterminando su posición, y dejando que el azar decida dónde se ubica definitivamente en el momento en el que ésta se define. En otros juegos cuánticos, como el tres en raya cuántico, en lugar de colocar una ficha en una casilla como norma el clásico juego, la colocamos en dos casillas. Estos movimientos intentan emular el comportamiento de la física a niveles en los que no reinan las leyes de la física clásica. En el caso nuestro del ajedrez cuántico podemos encontrar objetos definidos y objetos no definidos o indeterminados. Estos últimos son entidades cuánticas.

Queremos diseñar un sistema que implemente las características cuánticas ya descritas, hacer del juego una atractiva forma de traer las leyes de la cuántica a la población, mientras mantiene una interfaz y una funcionalidad sencilla de interpretar.

2.1 ALCANCE DEL PROYECTO Y DEL PRODUCTO

Las características del proyecto y su modelo incremental nos permitieron describir unos objetivos ambiciosos que fueron evaluados durante todo el proceso. La definición más básica de sus principales metas sería alcanzada con una máxima prioridad. Mientras la falta de tiempo y presupuesto dejaría muchos de los objetivos iniciales fuera de las especificaciones finales del producto en su versión 1.0. Sin embargo, dicho modelo permitirá importantes actualizaciones en fases venideras de su desarrollo.

2.1.1 Requisitos mínimos funcionales

Definimos objetivos principales como aquellos que serán los necesarios para la cumplimentación de los requisitos mínimos del sistema. Estas son:

1. Desarrollar un sistema que simule las características cuánticas más destacables y las implemente con unas normas definidas dentro de una interfaz sencilla y funcional.

2. Implementar dichas reglas y validaciones, proyecciones de las normas cuánticas siguiendo un sentido físico mínimamente riguroso. Entre los objetivos se encuentran:

- a. Definir la superposición cuántica: proveer de un sentido lúdico al carácter azaroso de estos sistemas. Como principio fundamental de la mecánica cuántica, sostiene que un sistema existe, parcialmente, en todos sus estados posibles.

- b. Dar lugar al entrelazamiento cuántico: este fenómeno supone una derivación de lo que podemos observar que es resultado de la superposición cuántica. Teóricamente es definido como la propiedad de un conjunto, cuyas partículas se encuentran entrelazadas a no poder definir sus partículas como individuos con estados definidos, sino como un único sistema. Con ello queremos decir que la definición de cada uno de ellos.

3. Hacer uso de la herramienta Stockfish para comprobaciones sencillas de estados del tablero clásico.

2.1.2 Otros objetivos dentro del alcance del proyecto

Otros objetivos dentro del alcance de este proyecto son:

1. Usar el motor de ajedrez Stockfish para realizar cálculos en posiciones clásicas para obtener un resultado que sea posible interpretar en estados cuánticos.
2. Crear una interfaz atractiva e intuitiva para el uso de la aplicación.
3. Que el usuario pueda solicitar ayuda de la interfaz para el uso de la misma.
3. Permitir el desarrollo con normalidad de una partida de ajedrez clásico entre dos jugadores (si no se realiza en ningún momento un movimiento cuántico), escribiendo sus movimientos a través de la línea de comando¹.
4. Almacenar toda la información necesaria para el transcurso efectivo y óptimo del juego. Para ello en muchas ocasiones será necesario saber si un peón ha dado dos pasos para permitir comer al paso, saber si las torres se han movido para permitir el enroque, o guardar el turno del jugador que corresponda.
5. Permitir que una pieza se mueva a dos sitios a la vez y almacenar todos los movimientos futuros que tengan conexión con esta indeterminación para definirlos cuando se produzca una colisión.
6. Obviar los movimientos ilegales inducidos por el jaque. El fin de una partida de ajedrez cuántico consiste en capturar al rey enemigo, no hacer jaque mate.
7. Mejorar el sistema de evaluación de posiciones, evaluación de movimientos y evaluación de superposiciones, mediante el uso de Minimax, Montecarlo u otros algoritmos de cálculo.
8. Mejorar el sistema de evaluación a través de Stockfish. Cálculo de la mejor-peor jugada, Opción segura. Ver más en el capítulo 9: Evaluando con Stockfish.

¹ Sin embargo, el juego no terminaría con el jaque mate, hemos pensado que las normas más adecuadas son aquellas que continúan la partida hasta que el rey es capturado.

3 ANÁLISIS DE REQUISITOS

El presente análisis de requisitos tiene como objetivo definir de forma clara los requerimientos funcionales y no funcionales para el desarrollo de un sistema que pueda trabajar conjuntamente con el motor Stockfish para hacer cálculos relacionados a la variante del ajedrez conocida como Ajedrez Cuántico. El sistema permitirá al usuario jugar una partida normal, humano contra humano o humano contra máquina. Además, permitirá obtener una evaluación o una sugerencia de movimiento dada una posición específica.

Requisitos Funcionales:

- Funciones de partida:
 - El sistema requiere la definición de unas normas que rijan con rigurosidad y se apliquen para las partidas entre humanos o que involucren a la computadora.
 - El sistema debe reconocer los movimientos introducidos por comando y evaluar la validez de los mismos, siguiendo las normas del ajedrez clásico, en unión con las normas del ajedrez cuántico.
 - El sistema debe saber gestionar posiciones complejas formadas por muchas otras posiciones, y evaluar cada una de sus funcionalidades recursivamente dentro de cada una de ellas.
 - El sistema debe reconocer cuando las normas del juego requieren de la definición de los estados cuánticos y encontrar el resultado de esta unión haciendo uso del azar y de la lógica al mismo tiempo.
 - El sistema debe gestionar todo el conjunto de banderas que habilitan el correcto uso de ciertos movimientos especiales del ajedrez clásico (el enroque y la captura al paso)
- Funciones contra la máquina:
 - El sistema debe obtener evaluaciones de la superposición de estados del juego.
 - El sistema debe recomendar una posible mejor jugada que optimice el valor de la evaluación de la posición resultante.

- El sistema debe ser capaz de evaluar los movimientos clásicos más prometedores y de calcular a partir de ellos los movimientos cuánticos más favorables.
- Funciones globales:
 - El sistema debe ofrecer al usuario jugar una partida humano contra humano, humano contra máquina o máquina contra máquina. En su defecto debe ser capaz de ofrecer en paralelo las jugadas que la máquina crea convenientes para que el usuario las juegue por ella.
 - El sistema debe ser capaz de representar de forma intuitiva el estado del tablero, algunas veces resultado de la superposición, y el estado de las piezas implicadas en el fenómeno.
 - El sistema debe proporcionar una guía de su consola para el correcto manejo por parte del usuario y el conjunto de reglas del juego.
 - El sistema debe ofrecer la posibilidad en todo momento de acabar la partida, y de reiniciarla si fuera necesario.

Requisitos no funcionales:

- Usabilidad
 - La interfaz debe ser intuitiva.
 - La interfaz debe ofrecer ayuda en todo momento.
 - La interfaz debe recordar las normas en caso de ser necesario.
- Fiabilidad
 - El sistema debe estar probado contra errores y fallos comunes, pero también contra aquellos específicos del caso particular.
 - El sistema debe estar pensado para la detención en caso de error fatal, y ofrecer un mensaje explicativo del error en cuestión.
- Escalabilidad
 - El sistema debe estar pensado para alcanzar mejoras de manera continua, actualizando la heurística implementada con el fin de desarrollar sus capacidades.
 - El sistema debe admitir la implementación de otras soluciones como alternativas a la propuesta por el desarrollo.

Estos requisitos tienen como objetivo consolidarse como los hitos de nuestro modelo. Modelo que aplicaremos de manera iterativa incremental a lo largo del proceso de desarrollo para terminar obteniendo un sistema capaz de resolver los problemas de accesibilidad y de cálculo que supone el ajedrez cuántico.

4 ANÁLISIS DE ANTECEDENTES Y APORTACIÓN REALIZADA

Las primeras menciones conocidas al ajedrez cuántico se le atribuyen a Selim Akl (2010), un reconocido investigador, experto en computación cuántica, quien dirige la Queen's School of Computing en Queen's University, Canadá. Sin embargo, la propuesta que incluye en su artículo "On the Importance of Being Quantum", describe una serie de variantes completamente distintas a la que nosotros hemos trabajado. La aleatoriedad de los sucesos se encuentra mucho más presente, lo que lo aleja más de la concepción clásica de ajedrez y dificulta la evaluación de sus estados.

Es más, el objetivo de está en que sea aleatorio. Selim Akl quiso crear un juego de estrategia en el que los ordenadores no tuvieran una ventaja significativa contra los humanos. A este científico se le atribuye el mérito en el enrevesado concepto que detalla en su artículo. Pero fue una alumna suya, Alice Wismath (2012), quien desarrolló estos conceptos en un modelo informático. Para ello, Alice programó y confeccionó diferentes versiones entre las propuestas por Selim Akl. Entre ellas, una inteligencia también capaz de jugar.

En una de sus variantes, la superposición de estados típica en la física cuántica se refleja en el tipo de pieza. Antes de seleccionar la pieza que vamos a mover no conocemos de cual se trata, y esto se debe a que cada pieza es la superposición de varias a la vez. De este modo, una pieza puede ser el resultado de superponer el caballo izquierdo con la torre derecha. Una pieza queda definida cuando decidimos moverla, es decir, cuando la observamos y puede volver a ser cuántica si aterriza en una casilla negra. De lo contrario, en una casilla blanca mantiene su estado definido.

Basado en estas versiones del juego, la Queen's University organizó el primer torneo de Ajedrez Cuántico, en el cual también se observaron partidas contra ordenadores (programados también por Alice). El torneo concluyó con las máquinas venciendo a todos sus rivales novatos y amateurs, sin mucha experiencia en el ajedrez ni en esta nueva modalidad. Sin embargo, el sorprendente hito fue descubrir que jugadores más experimentados podían llegar a ganar hasta tantas partidas contra las máquinas como estas contra ellos. Al fin y al cabo, ese era el objetivo del juego.

Esta alocada variante del ajedrez ha caído en el olvido, y hoy día pocos recuerdan los auténticos comienzos del ajedrez cuántico como concepto. Otras versiones se han hecho hueco en el panorama comercial y científico, pero la versión de Alice todavía se puede encontrar entre las bases de datos de la Queen's University.

El ajedrez cuántico del modo que lo conocemos es diseñado por Christopher Cantwell en sus artículos “Quantum Chess: Making Quantum Phenomena Accessible” (2017) y “Quantum Chess: Developing a Mathematical Framework and Design Methodology for Creating Quantum Games” (2019). Su investigación ofrece una posible descripción del tablero y una serie de modelos que ofrecen una solución matemática a, por ejemplo, el estado del tablero o los movimientos de las piezas.

A diferencia de las soluciones desarrolladas por Selim Akl y Alice Wismath, la superposición cuántica propone un cambio sustancial en el planteamiento. Y es que no se trata de la superposición de tipos de piezas, sino de la superposición del estado de su ubicación. Esto conlleva a que, si en algún momento decidimos hacer cuántica una pieza, su posición quedaría indeterminada. A efectos prácticos, la pieza tiene una probabilidad de encontrarse en un sitio y otra probabilidad complementaria a la anterior de encontrarse en otro.

En este aspecto, Cantwell define unas normas que dan mucha libertad al jugador para hacer con los estados cuánticos lo que el jugador desee. En cualquier momento, puede crear esos estados de superposición o revertirlos definiendo su posición.

La notación propuesta por Cantwell utiliza un esquema del tipo “origen + destino” para movimientos normales, “origen + \wedge + destino + destino” para movimientos de superposición cuántica (separación de piezas) y “origen + origen + \wedge + destino” para movimientos de unión cuántica.

La solución de Cantwell cogió fuerza tras el corto *Paul Rudd explores the Quantum Realm with Stephen Hawking* protagonizado por Paul Rudd (actor que da vida a Antman) y el famoso físico Dr. Stephen Hawking. En este, los dos protagonistas se enfrentaban en una partida de ajedrez cuántico.

Desde entonces también han tenido lugar otros eventos con el ajedrez cuántico como actor principal. Un claro ejemplo son los torneos Q2B organizados por Quantum Realm Games, que fueron organizados en 2020 y 2021.

La presente investigación nos ha hecho entender que el esfuerzo y el trabajo de muchas personas ya se ha volcado en esta materia para dar lugar a teorías extraordinarias. A todo este estudio hemos querido aportar trayendo nuestro trabajo en el que:

- Analizaremos en profundidad el desarrollo del proyecto para conocer en profundidad características de la variante propuesta. El resultado será un programa con una interfaz sencilla y su licencia será para el libre uso de la comunidad.

- Estudiaremos los posibles algoritmos de medida y evaluación, junto con la posible inclusión de la librería de Stockfish para la métrica de elementos clásicos y su extrapolación a los elementos cuánticos.
- Nuestra investigación explorará temas sobre los que todavía no se tiene registro, la evaluación de juegos cuánticos, haciendo uso de las más potentes herramientas para el cálculo ajedrecístico.

Nuestra aportación será simbólica comparada con los grandes avances de la ciencia de la computación cuántica o la innovación en la educación sobre estos temas, pero ambiciosa. Mas no podemos considerarlo un punto y final, pues esperamos poder seguir desarrollando esta herramienta y perfeccionándola para acercar la física cuántica a toda la población.

5 NORMAS DEL AJEDREZ CUÁNTICO

Las normas de esta variante están conformadas por un grupo de normas procedentes del ajedrez clásico sumadas a otro conjunto de normas específicas del modo de juego.

Primero enumeraremos las normas del ajedrez clásico empleadas en el ajedrez cuántico:

- Los movimientos de las piezas siguen siendo los mismos a los posibles en el ajedrez clásico.

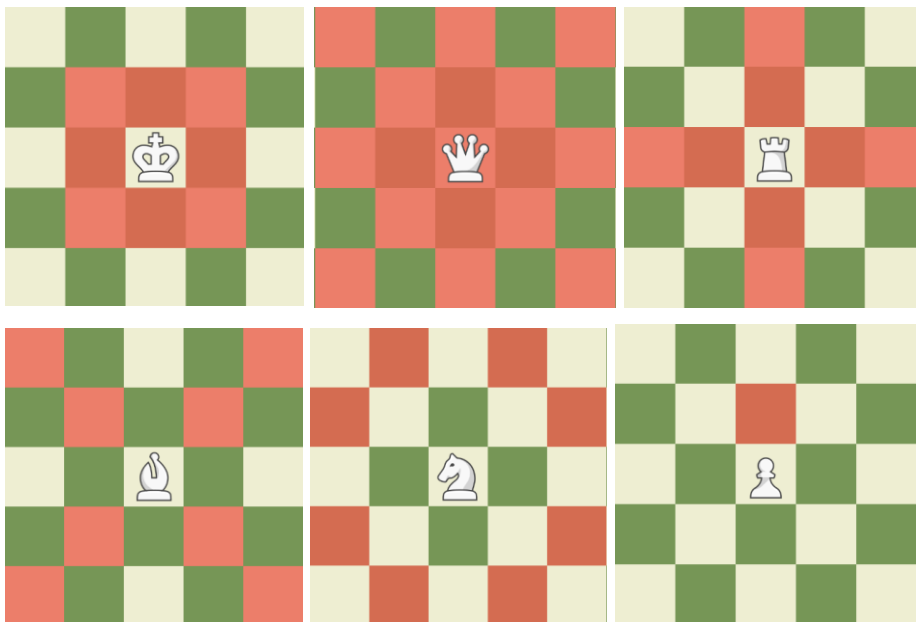


Figura 1: Movimientos clásicos

- Los movimientos de peón al comienzo dando dos pasos, la captura en diagonal del peón, la captura al paso y el enroque del rey son válidos.
- La coronación permite convertir un peón que haya llegado a la última fila en una pieza de su elección.
- A diferencia del ajedrez clásico, los movimientos en jaque son todos legales. Cuando nuestro rey se encuentra bajo ataque enemigo, ya no es obligatorio defenderlo, aunque sí recomendable. Perder el rey supone el final de la partida.
- La partida no finaliza con un jaque mate, sino capturando el rey.

5.1 REGLAS CUÁNTICAS AÑADIDAS

Las siguientes normas y posibilidades son añadidas a la variante:

- Entidades cuánticas: Son entidades cuánticas aquellas cuyas características no están completamente definidas. En ellas se encuentra una superposición de estados (varios estados a la vez) que indetermina parte de la información que esta entidad nos proporciona. Las entidades cuánticas pueden describirse como la aleatoriedad entre dos (o más) estados. Existe una cierta probabilidad de que sea descrito con un estado A y otra probabilidad complementaria de estar descrito con un estado B siendo $A \neq B$.
 - Piezas cuánticas: Es el resultado de indeterminar la posición de una pieza. La pieza puede estar o no estar en el estado descrito, pero debe existir en alguna parte del tablero.
 - Movimientos cuánticos: Es la acción que provoca la indeterminación de una pieza. Ocurre al mover cuánticamente una misma pieza a dos casillas diferentes. Se compone de dos estados que son dos movimientos clásicos. En esta variante, ambos tienen la misma probabilidad de suceder.
 - Posiciones cuánticas: Es el resultado de tener piezas cuánticas en el tablero. Los posibles estados del tablero son todas las combinaciones de todos los estados posibles de las piezas, dando un valor de como mucho $2^{\frac{n}{2}}$ posiciones posibles, siendo n el número total de estados posibles de las piezas.
- Podemos elegir realizar tanto movimientos cuánticos como movimientos clásicos.
- Si el movimiento resulta en la coexistencia de dos piezas en una misma casilla en superposición, el movimiento no es válido bajo ningún concepto.
- Una pieza cuántica puede ser atravesada dando lugar al entrelazamiento cuántico. El resultado será que en un estado la pieza es posible que pueda hacer el movimiento y en otro es posible que no. Al realizar ese movimiento la pieza queda en superposición y su estado estará ligado al de la pieza que obstaculizaba su movimiento.
- Se define como observación de las entidades cuánticas las capturas de material: Si una pieza cuántica captura o es capturada, se considerará una observación y se verá obligada a ser definida.
- Observación: Al suceder una captura entre piezas cuánticas, se calcula de forma aleatoria el estado definido de las piezas implicadas.
- El movimiento de captura se realiza independientemente de si la pieza capturada resulta estar en la casilla objetivo, a excepción de:

- Una pieza cuántica causando una interferencia se defina siendo un obstáculo para el movimiento de captura.
 - La pieza que realiza el movimiento se define en un estado distinto al que realiza el movimiento.
- En el caso del peón, el movimiento de captura en diagonal se realizará, aunque la definición de los estados determine que no haya ninguna pieza que pueda capturar.
- El rey puede eludir un mate realizando un movimiento cuántico. Al indeterminar su posición, el rival deberá capturar uno de sus subestados para tener una oportunidad aleatoria de ganar la partida.

6 ANÁLISIS TEMPORAL Y DE COSTES

La distribución del trabajo no pudo ser distribuida exhaustivamente para completar un modelo que requería del testeo durante las iteraciones para marcar una nueva hoja de ruta con cada nueva implementación. Sin embargo, había algunas pautas marcadas desde el inicio del proyecto.

Teníamos varios campos marcados sobre los que pretendíamos hacer avances conjuntos para no dejar ningún aspecto del proyecto por detrás.

En la Figura 1 podemos observar una Estructura de Desglose de Trabajo.

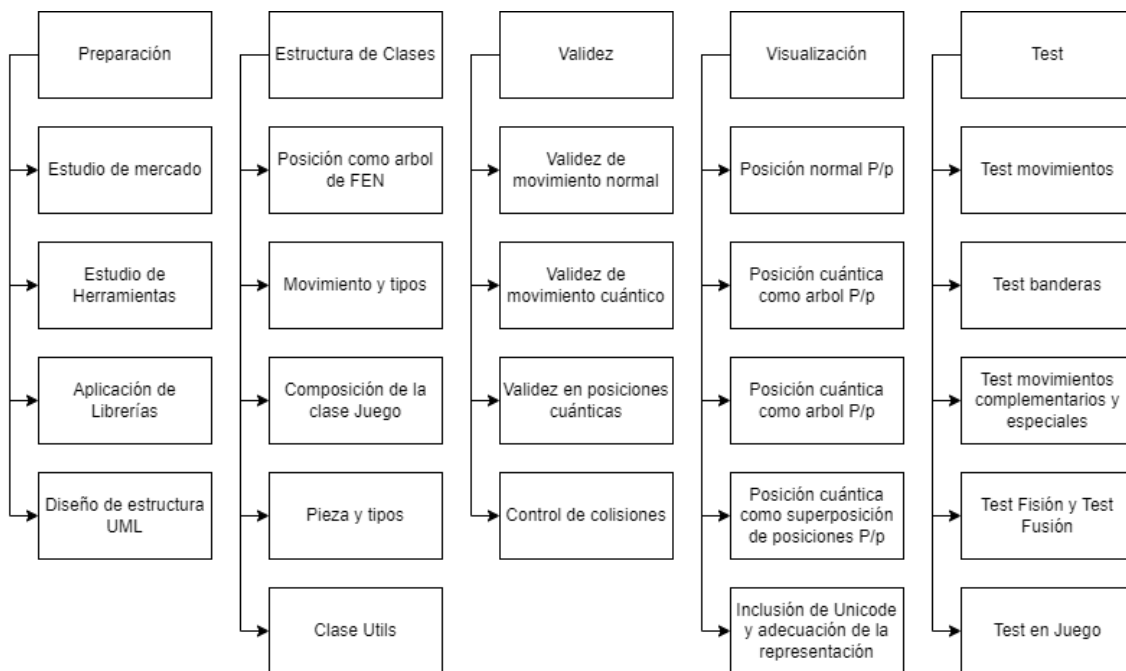


Figura 2: Estructura de Desglose de Trabajo

Este desglose nos ayudaría a separar las diferentes categorías de las tareas y llevar un mejor seguimiento del tiempo de implementación de cada apartado.

A partir de las tareas propuestas, el flujo de trabajo siguió un patrón descrito a continuación en el siguiente diagrama de la Figura 2. Inmediatamente después podemos ver con más rango de detalle los inicios, duraciones y dependencias de las tareas en el Cuadro 1.

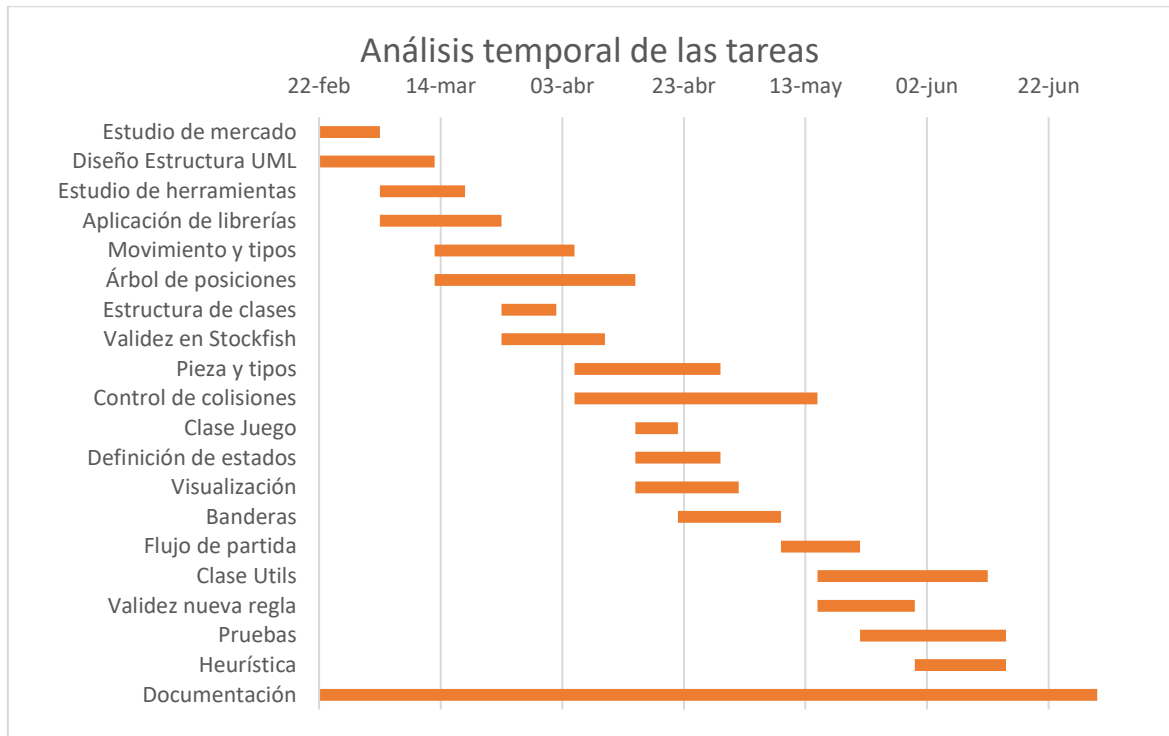


Figura 3: Diagrama de Gantt, Análisis temporal de tareas

	Fecha Inicio	Duración	Fecha Fin	Requisitos	
Estudio de mercado	22/02/2023	10	04/03/2023		
Diseño Estructura UML	22/02/2023	19	13/03/2023		
Estudio de herramientas	04/03/2023	14	18/03/2023	Estudio de mercado	
Aplicación de librerías	04/03/2023	20	24/03/2023	Estudio de mercado	
Movimiento y tipos	13/03/2023	23	05/04/2023	Diseño Estructura UML	
Árbol de posiciones	13/03/2023	33	15/04/2023	Diseño Estructura UML	
Estructura de clases	24/03/2023	9	02/04/2023	Aplicación de librerías	Diseño Estructura UML
Validez en Stockfish	24/03/2023	17	10/04/2023	Aplicación de librerías	
Pieza y tipos	05/04/2023	24	29/04/2023	Movimiento y tipos	
Control de colisiones	05/04/2023	40	15/05/2023	Pieza y tipos	
Clase Juego	15/04/2023	7	22/04/2023	Árbol de posiciones	
Definición de estados	15/04/2023	14	29/04/2023	Árbol de posiciones	Estructura de clases
Visualización	15/04/2023	17	02/05/2023	Árbol de posiciones	
Banderas	22/04/2023	17	09/05/2023	Clase Juego	
Flujo de partida	09/05/2023	13	22/05/2023	Banderas	
Clase Utils	15/05/2023	28	12/06/2023	Control de colisiones	
Validez nueva regla	15/05/2023	16	31/05/2023	Control de colisiones	
Pruebas	22/05/2023	24	15/06/2023	Flujo de partida	Visualización
Heurística	31/05/2023	15	15/06/2023	Validez nueva regla	Clase Utils
Documentación	22/02/2023	128	30/06/2023		

Cuadro 1: Análisis temporal de tareas

En el Cuadro 1, los requisitos son tareas que se deben haber completado para poder proceder con la tarea. Los tiempos han sido ajustados entre la expectativa inicial del análisis temporal y el flujo actual del trabajo.

El tiempo estimado para la finalización del proyecto eran 300 horas. Su distribución fue desigual a lo largo del desarrollo de iteraciones y tareas, pero ahora podemos marcar una estimación de cómo se han distribuido esas horas.

- Primera iteración:

Lo que dicha iteración requiso del equipo fue el tiempo necesario para plantear diferentes propuestas de diseño y estructuración del programa. Fue necesaria investigación previa y referencias. Aparte, requerimos una pequeña proporción de ese tiempo para el desarrollo de las clases.

Estimamos que la duración fue de 30 horas.

- Segunda iteración:

Los primeros pasos en el desarrollo de la interfaz, estructura de clases y primeros métodos. El apartado más sencillo del proyecto requirió tan solo que nos pusiéramos manos a la obra en el código.

Estimamos que la duración fue de 20 horas.

- Tercera iteración:

Por los cambios introducidos en la estructura de las clases y las adaptaciones en los métodos, junto a la corrección de bugs y errores fatales, la tercera iteración fue la más larga. Requirió no solo tiempo depurando código, pero también buscando posibles soluciones y alternativas.

La duración estimada fue de 80 horas.

- Cuarta iteración:

La siguiente iteración buscaba perfeccionar todos los aspectos que comenzó y quedaron incompletos durante la tercera iteración. Con muchos frentes abiertos, la cuarta iteración requiso también de muchas adaptaciones, depuración y correcciones. Esta fase incluye muchas iteraciones de testeo.

Estimamos que la duración fuera de otras 80 horas.

- Quinta iteración:

Los objetivos principales ya estaban casi todos cumplidos, quedaba completar de testear el sistema en los distintos casos planteados. Para acabar, finalizar las últimas funcionalidades y ajustar la interfaz de usuario.

Estimamos que esta fase durara 50 horas.

A este tiempo le añadimos el tiempo para documentación, investigación y recogida de datos, a lo que le estimamos 40 horas.

El apartado temporal ha sido menos minucioso que otros pues hemos contado con la libertad de poder distribuir el tiempo de manera flexible y eficaz. La metodología aplicada nos ha permitido gestionarlo de dicha manera, permitiéndonos seleccionar el alcance

del proyecto para los distintos plazos. Consideramos que el empleo de dicho recurso ha sido satisfactorio y ha proporcionado un resultado que se encuentra a las alturas de las expectativas formadas en el comienzo.

7 DISEÑO E IMPLEMENTACIÓN

Para formar una base sólida de lo que se implementaría en un futuro necesitamos generar la estructura necesaria, el esqueleto de nuestro sistema.

Las tareas han sido diseñadas para seguir un modelo iterativo incremental. Empezando por tareas sencillas, el programa irá realizando poco a poco más funciones dentro de las especificaciones finales deseadas.

Pero antes de comenzar el proyecto, fueron necesarias una serie de preparaciones que habilitaran al equipo y lo formaran con todas las herramientas que pudieran estar a su alcance. El objetivo consistió en capacitar para la posterior toma de decisiones.

7.1 Preparación

El ajedrez cuántico, una variante del clásico juego de mesa, es una modalidad que incluye comportamientos de la mecánica cuántica en aspectos del juego.

Para poder diseñar un proyecto con el objetivo de crear una aplicación para el ajedrez cuántico, serían necesarias las debidas fuentes de inspiración.

- Artículos escritos por Christopher Cantwell.
- Vídeos explicativos o divulgativos, como *Reto a una Gran Maestra al Ajedrez Cuántico*, del canal de Youtube Quantum Fracture (2021) o *Paul Rudd explores the Quantum Realm with Stephen Hawking*, de Institute for Quantum Information and Matter (2016)
- Vídeos sobre competiciones (Quantum Chess Tournament | Final | Q2B20, 2020)
- Otros ejemplos de juegos cuánticos, como el tres en raya.

Seguidamente, cobraron una gran importancia, sobre la investigación previa al diseño, el uso de las librerías adecuadas. Ser capaces de conocer las funcionalidades y limitaciones de las librerías más propensas a ser usadas en este proyecto contribuyó a la comprensión de los resultados obtenidos. Entre estas librerías destacamos Stockfish, la cual forma parte de los requisitos del sistema y tiene un papel determinante entre los objetivos de este proyecto, así como su alcance en el futuro.

- Librería Stockfish para Python (Stockfish Pypi, 2022): El estudio de la librería Stockfish consistió en una serie de pruebas de uso en situaciones cotidianas de ajedrez, así como algunos cálculos un poco más complejos sobre posiciones

cuánticas, o la evaluación de movimientos cuánticos. Durante las primeras iteraciones, sirvió también con el objetivo de evaluar la validez de los movimientos, pero este uso tuvo que ser descartado y reprogramado. Más información en el apartado de la *cuarta iteración*. Así mismo era también objetivo del estudio que Stockfish fuera capaz de evaluar posiciones con piezas cuánticas y movimientos cuánticos.

- Librería Chess: Una decisión aparentemente obvia, el objetivo de esta inclusión fue poder representar de forma visual los resultados obtenidos en los primeros ciclos de vida del programa. Ahora nos sirve para detectar ataques, pero sobre todo para detectar cuando el rey se encuentra amenazado y puede ser perdido.
- El uso de otras librerías adicionales propias de Python (como math, random o re) han sido necesarias para llevar a cabo el conjunto de funcionalidades deseada.

La organización ha permitido la correcta distinción de objetivos a todos los plazos por parte del equipo, y por ello ha sido en todo momento de vital importancia mantener un registro del estado y los progresos dentro de nuestra hoja de ruta. No obstante, es clave aludir a la importancia de la flexibilidad que nos ha ofrecido seguir un modelo iterativo incremental. Durante el proceso de desarrollo, importantes puntos de inflexión han requerido del equipo tomar decisiones complicadas que contravenían las pautas iniciales definidas entre los requisitos.

Consideramos que es de gran importancia subrayar estos cambios en la estrategia, y de este modo hemos documentado cada variación en los planteamientos.

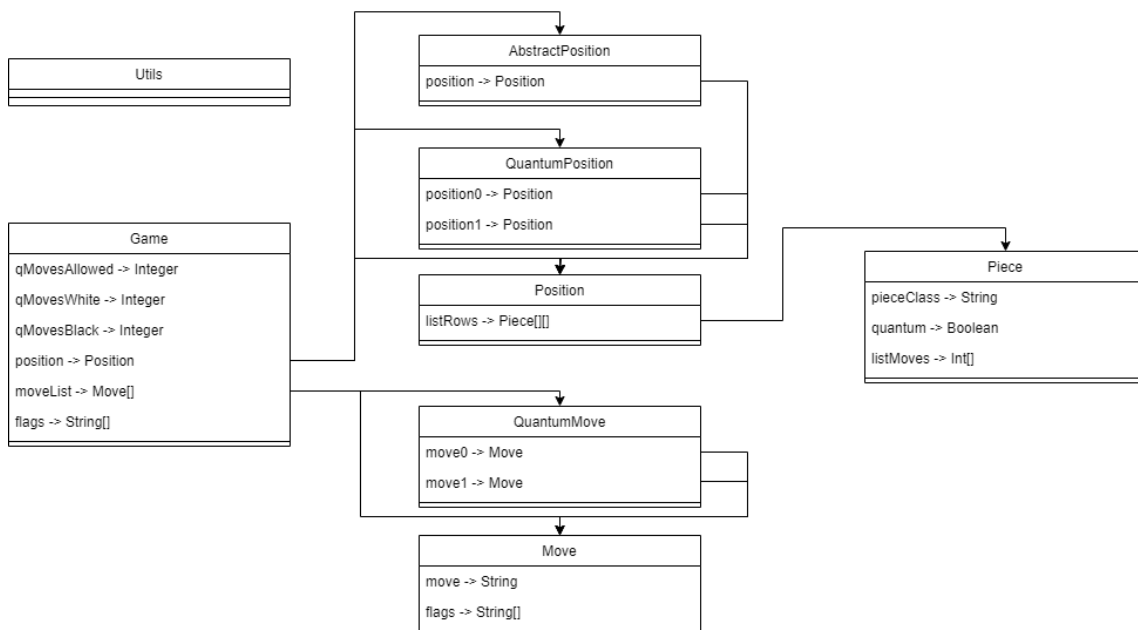


Figura 4: Diagrama de clases

Para reflejar una idea de las clases involucradas en el proyecto, realizamos el debido diagrama de clases. Con las iteraciones fue modificado hasta resultar en el diagrama de la Figura 4. En él, podemos observar que la clase juego será la clase principal encargada de guardar todas las variables de la partida. En ella iremos añadiendo posiciones y movimientos. Estas posiciones o movimientos pueden ser clásicas o complejas. Las posiciones o movimientos complejos están formados por una o dos posiciones o movimientos clásicos. Las posiciones clásicas están definidas como una matriz del objeto piezas.

Para desarrollar el proyecto siguiendo un modelo incremental, definimos primero los objetivos de una primera iteración, pasos a seguir y que deseamos que sean capaces de superar.

7.2 PRIMERA ITERACIÓN

En una primera iteración marcamos los siguientes objetivos:

- Creamos las clases
- Creamos posiciones simuladas
- Utilizamos Stockfish para comprobar el resultado que obtenemos y si se asemeja a el resultado que esperamos

Los test comprobaron que, en efecto, Stockfish puede valorar movimientos válidos para una posición que nosotros le otorguemos.

Para recrear la posición decidimos en principio usar la notación FEN. Una posición normal tendrá un atributo FEN que sería suficiente para poder definirla. En futuras iteraciones pudimos advertir que sería necesario otros objetos que guardaran más información, las piezas.

Cuando el usuario decide hacer un movimiento cuántico, está llevando a una sola pieza a dos casillas distintas (la unión de dos movimientos normales). Esto conlleva que el tablero se encuentre en dos estados a la vez, uno en el que la pieza ha realizado el primer movimiento descrito por el jugador, y otro en el que la pieza ha realizado el segundo.

Lo que hacemos con esta información es, a partir de estos dos movimientos, crear dos posiciones normales, calculamos cual es el FEN resultante de esos movimientos, y guardamos estos dos objetos Position como atributos de un nuevo objeto QuantumPosition. De este modo, con cada movimiento cuántico creamos un árbol binario de posiciones.

Consideramos que la evaluación correspondiente a una posición cuántica, es la de la media de todas sus posiciones hijas. De este modo, si en una de las dos posiciones un jugador tiene mucha ventaja y en otra posición es el oponente quien se ve favorecido, la media será próxima a 0. Esto es totalmente coherente porque es cuestión de azar, 50% de definirse el tablero en el estado favorable de cada jugador. La partida se encontraría equilibrada.

Por ello hemos llegado a la conclusión de que una función recursiva tendría una forma muy cómoda de evaluar una posición cuántica si siguiera el pseudocódigo descrito a continuación:

Evalúa(posición)

Si posición cuántica:

Retornar (evalua subposición 1 + evalua subposición 2) / 2

Si no:

Stockfish evalúa posición

Retornar evaluación

Para estos algoritmos usamos los siguientes comandos de Stockfish:

```
stockfish.set_fen_position(fen)
stockfish.is_fen_valid(fen)
stockfish.make_moves_from_current_position(moveList)
stockfish.get_evaluation().get("value")/100.0
stockfish.is_move_correct(move.move)
```

En este último, `get_evaluation()`, nosotros recibimos un objeto tipo dictionary y tenemos que coger el valor de "value". Además, la puntuación se encuentra en centipeones, lo que significa que cada 100 puntos acumulan al valor de un peón. Para ver el valor en peones, tan solo tenemos que dividirlo entre 100.

Usamos `is_fen_valid()` e `is_move_correct()` para el control de errores y de posiciones conflictivas. Información más detallada en el Capítulo 9: Evaluando con Stockfish.

Todos los métodos tendrán que diferenciar si son posiciones de tipo cuánticas o no cuánticas, para poder aplicar siempre el algoritmo adecuado para estos, ya que sus atributos no son iguales.

Los movimientos son objetos creados para almacenar toda la información relacionada con estas acciones dentro del juego. Necesitamos diferenciar si son cuánticos o no, y almacenan una o dos cadenas de caracteres con el movimiento en UCI².

Para crear los objetos QuantumPosition necesitamos que haya un movimiento cuántico. Estos vendrán escritos también mediante UCI igual que los movimientos normales.

- Método makeMove(Move): Realizará los movimientos clásicos en el tablero
- Método makeMove(QuanticMove): Creará las posiciones cuánticas resultantes de los dos movimientos clásicos descritos como parámetros.

El método descrito en la clase Position será el encargado de alterar al objeto mismo para convertirlo en el resultado de los movimientos, y será responsable de distinguir las acciones correspondientes en el caso de que el movimiento sea cuántico o sea clásico.

Para evaluar una jugada, hicimos uso del método creado para hacer movimientos y el método para evaluar posiciones. En este caso tan solo hubo que crear una copia de la posición actual, ejecutar el movimiento y evaluar la posición resultante. Sin embargo, habría ocasiones en las que los movimientos no fueran válidos para una de las posiciones. Esto se debe a que, si el usuario decide mover una pieza cuántica, esta pieza solo existe en una de las ramas de nuestro árbol binario de posiciones. Por lo cual, al evaluar el movimiento debemos devolver la valoración de la posición sin ningún movimiento, pero cambiando de turno al otro jugador.

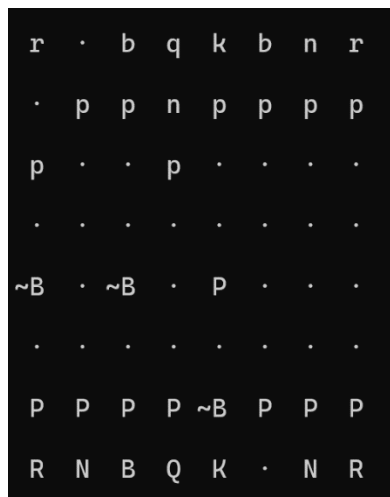


Figura 5: Resultado de mover cuánticamente dos veces una misma pieza

² Universal Chess Interface (UCI) es un protocolo abierto de comunicación para la comunicación entre las interfaces y los motores de ajedrez. Usa entre cuatro y cinco caracteres para describir el movimiento, puesto que cuatro caracteres son los necesarios para definir una casilla de origen y una casilla de destino. Todos los movimientos del ajedrez se pueden definir de esta manera. El quinto carácter, que es opcional, es para definir en qué pieza se desea coronar cuando los peones coronan.

En la Figura 5 podemos contemplar una representación del tablero (a partir del modelo desarrollado en la segunda iteración) en el que podemos ver una pieza tras dos movimientos cuánticos. La pieza cuántica se muestra con el símbolo “~”. La “B” mayúscula nos informa de que es el un alfil blanco. La figura mostrada es el resultado de mover cuánticamente el alfil de casillas blancas desde f1 a e2 y a b5 a la vez, y después mover el alfil desde b5 a a4 y a c4 a la vez. Nuestro sistema solo permite dividir una pieza en dos piezas iguales con la misma probabilidad, pero cada una de ellas se puede volver a subdividir en los siguientes movimientos resultando en estados menos probables. Una pieza con una probabilidad del 50% al dividirse resultara en dos estados que, cada uno, represente el 25%, sumando así todos los posibles estados: $50\% + 25\% + 25\% = 100\%$.

Con esto habríamos concluido los objetivos de una primera iteración. Un ciclo sencillo con objetivos que no requieran de una compleja estructura para poder sentar las bases de lo que se construiría más adelante.

7.3 SEGUNDA ITERACIÓN

La siguiente iteración traería consigo objetivos más ambiciosos:

- Construir la clase Juego
- Crear y administrar las banderas necesarias
- Probar combinaciones de movimientos cuánticos y clásicos
- Observar el resultado dentro del árbol de posiciones
- Crear un método que nos proporcione una representación visual de los estados cuánticos y clásicos

La clase juego almacenaría la cantidad de movimientos cuánticos permitidos en total y la cantidad de éstos realizados hasta el momento durante la partida. Guardaría por un lado la posición de las piezas, como una variable tipo posición, el turno, la posibilidad o no de enroque en cada caso (corto o largo, del blanco o del negro) y, por último, una lista de los movimientos que en esta partida se han jugado.

Uno de los retos que traía consigo esta segunda iteración sería evaluar la validez de cada movimiento. Habíamos decidido, en un principio, que esto sería evaluado usando Stockfish. Siguiendo un algoritmo similar al que encontramos en la evaluación de posiciones, si ejecutamos el método en la clase movimiento, podemos obtener algo como lo descrito a continuación.

```
esVálido(movimiento, posición)
```

```
Si posición cuántica:
```

```
Retornar (esVálido(movimiento, sub-posición 1) O esVálido(movimiento, sub-posición 2))
```

```
Si no:
```

```
Stockfish, ¿es válido movimiento en posición?
```

```
Retornar resultado
```

Queremos usar los métodos que nos ofrece Stockfish de modo que haremos uso de los mismos métodos que anteriormente para evaluar. También es posible evaluar la validez de un movimiento con la librería Chess.

En posiciones cuánticas, el resultado es válido si el movimiento lo es en la posición 1 o en la posición 2. Esta operación es disyuntiva, debido a que el movimiento solo se requiere que sea válido en al menos uno de los escenarios posibles.

Y en caso de que el movimiento sea cuántico, tenemos que comprobar tan solo si ambos movimientos de forma individual ya son válidos. Esta operación, en contraposición, es conjuntiva. Pues al hacer dos movimientos a la vez requerimos que ambos sean legales.

Algo muy favorable a la notación a partir de UCI es que, en casi la totalidad de posibles movimientos, las únicas casillas afectadas por este son las que están descritas por dicha interfaz como casilla origen y casilla destino. A esta norma se escapan dos tipos de movimientos. Estos son el enroque y la captura al paso.

- El enroque es un movimiento que requiere de unas condiciones específicas para poder ser ejecutado. Ni el rey ni la torre con la que se ejecuta el movimiento se han debido mover en ningún momento antes en la partida. No puede haber ninguna pieza situada entre ellas dos y, además en el ajedrez clásico, si la casilla del rey o alguna de las casillas por las que el rey pase se encuentran atacadas por el enemigo, el enroque no es legal. El movimiento consiste en mover el rey dos pasos en dirección hacia la torre y que la torre salte por encima del rey y se sitúe a su otro lado. Es comúnmente usado para esconder al rey y resguardarlo. En la notación UCI solo es necesario usar la casilla origen y destino del rey.
- La captura al paso (captura de peón lateral o en passant) es un movimiento exclusivo de los peones para capturar otros peones si se dan las condiciones adecuadas. El peón que captura debe estar en la quinta fila (o en la cuarta si es negro). El peón capturado debe estar en una columna adyacente al peón que va a realizar la captura, y se ha de mover dos casillas desde su posición inicial. El peón que realizará la captura lo debe hacer en el movimiento inmediatamente

siguiente al doble salto del peón capturado, y lo hará como si el anterior hubiera dado un solo paso.

Estos movimientos son únicos en el ajedrez pues modifican casillas y piezas que no se encuentran en las casillas origen ni destino. Hasta ahora los movimientos habían sido sencillos porque las casillas que modificaban ya venían definidas, pero para estos dos casos particulares, hemos tenido que describir lo que hemos llamado “movimiento complementario”, llamando así a la función que calcularía y modificaría estas casillas en caso de que esto fuera necesario.

Seguidamente necesitamos una manera de representar el tablero que pudiera hacer más fácil comprobar que las jugadas se ejecutaban correctamente. En una primera versión del método `__str__` predeterminado de Python para los objetos `Position`, en los casos en los que estas posiciones fueran no cuánticas, transformamos la notación FEN en una matriz de caracteres que representaban piezas. Al imprimirlas alineadas y hacer un pequeño cambio en los huecos vacíos, formaban lo que podíamos intuir como un tablero.

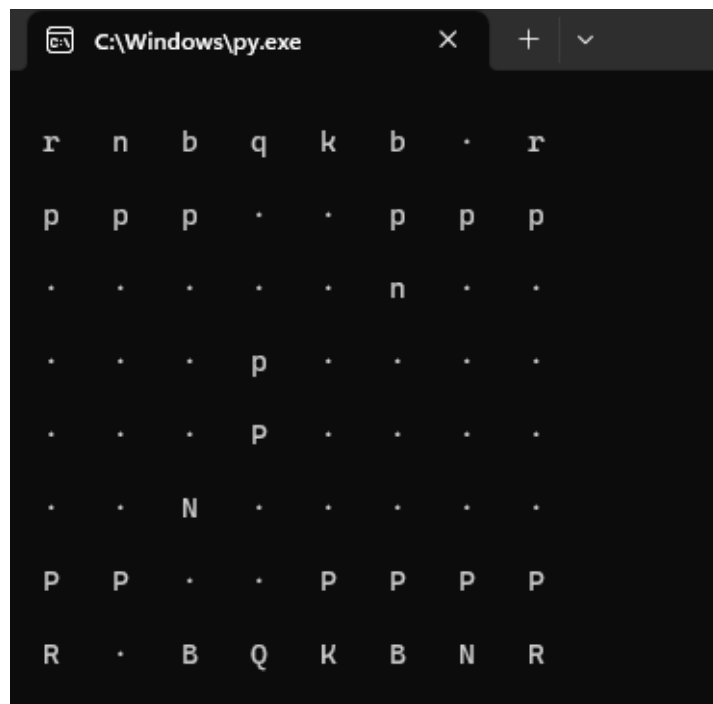


Figura 6: Primera representación visual del tablero

En esta Figura 6 encontramos una primera representación del tablero, con las letras mayúsculas para representar piezas blancas y las minúsculas para las negras. Las posiciones cuánticas añadían una ligera dificultad, pero en la primera versión hicimos lo mismo que todos los métodos para las posiciones cuánticas hacen, y lo ejecutamos para sus subposiciones. Esto dio como resultado que, si en una partida se hacían 3 jugadas cuánticas, se imprimían hasta 2^3 posiciones posibles.

Como conclusión de esta segunda iteración, realizamos las siguientes pruebas:

- Movimientos normales
- Movimientos cuánticos
- Comprobación de las estructuras creadas

El movimiento entrelazado fue una característica que encontramos como consecuencia de los movimientos cuánticos. Estos ocurren cuando una pieza pretende hacer un movimiento que es interrumpido por una pieza cuántica en alguno de sus posibles estados. En el caso de que la pieza estuviera en dicho estado, el movimiento no sería posible. Pero si no fuera ese el caso, el movimiento se ejecutaría sin problema. Dado que un movimiento solo necesita ser válido en al menos una de las posiciones posibles, este se ejecuta en dicho escenario mientras que, en caso de no ser posible, la pieza se mantiene en su posición original. Hemos cuantificado la pieza con un movimiento clásico, pero no hemos aumentado el número de posibles estados del tablero. Se puede decir que el estado de estas dos piezas está ligado pues el resultado del estado de una depende directamente del otro.

Sin embargo, esto daría muchos problemas en iteraciones posteriores como podremos ver más adelante.

7.4 TERCERA ITERACIÓN

Objetivos de la tercera iteración:

- Añadir la definición de estos estados cuánticos.
 - Clase Pieza
 - Clase AbstractPosition

Según hemos descrito en apartados anteriores, la definición de las piezas cuánticas significa la unión de sus estados en uno de ellos dos de forma azarosa. También hemos definido que esto debía ocurrir cuando una pieza cuántica realizara una captura o fuera capturada.

Para eso también hemos usado la librería de Stockfish. Generalmente, detectar una captura es sencillo, pero el caso de la “captura al paso” lo hace ligeramente más complicado. Consultar a una librería que directamente nos provee con la respuesta a nuestra pregunta nos pareció la solución más práctica.

Una vez aclarado cuándo tiene que tener lugar una definición de una superposición cuántica, nos queda llegar a la implementación. Este es uno de los principales inconvenientes del método de desarrollo especificado, y es que a veces son necesarios cambios profundos en el planteamiento de la implementación.

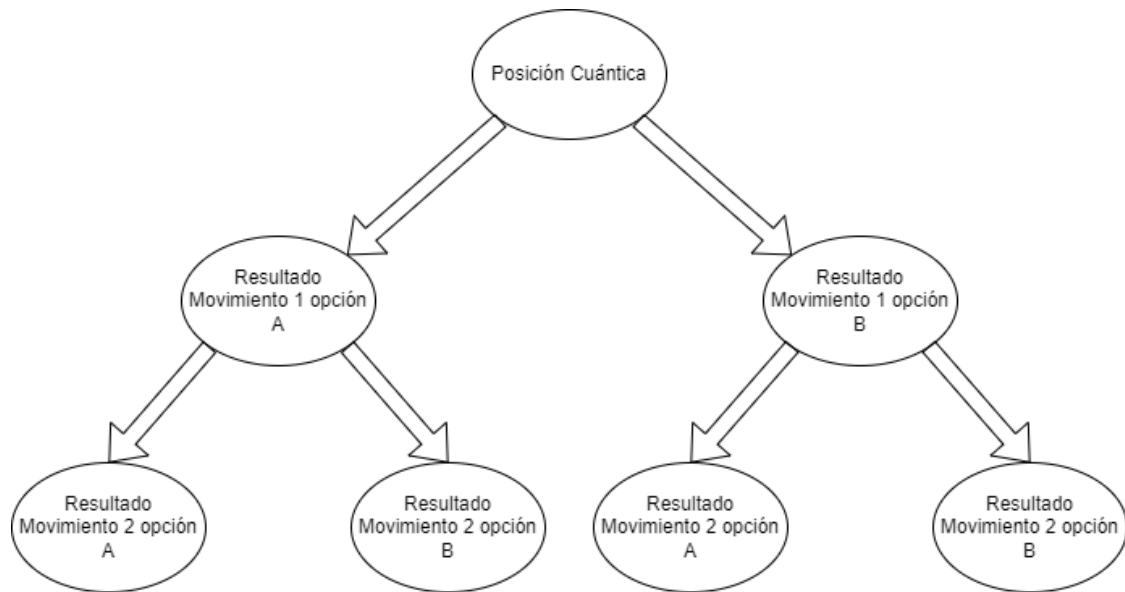


Figura 7: Árbol binario de posiciones. Idea original

Habíamos implementado una estructura de árbol binario para las posiciones cuánticas dando lugar a una representación similar a la que encontramos en la Figura 7. Cuando realizamos un movimiento cuántico, dividimos la actual posición en dos posiciones más, una posición resultada del primer movimiento y otra resultada del segundo. Pero cuando seguimos realizando movimientos cuánticos sobre estas posiciones, distinguirlas se vuelve más complicado.

Los movimientos se ejecutan siempre sobre las hojas del árbol. Esto quiere decir que los nodos más profundos son resultado de los movimientos más recientes. Cuando realizamos un movimiento cuántico, nuestra posición se bifurca en dos hijos, derecho e izquierdo para las posiciones resultantes del movimiento 1 y 2 superpuestos en una única posición cuántica. Al realizar más movimientos cuánticos creamos más nodos. Pero todos van a seguir siempre la misma regla, derecho para el movimiento 1 e izquierdo para el movimiento 2.

Si queremos colapsar el estado de una pieza que entró en superposición en el quinto movimiento cuántico, tan solo tenemos que buscar en el árbol el quinto nivel y seleccionar el hijo derecho o izquierdo de cada rama y sustituir al padre por el hijo dentro del árbol.

Esto suponía varios problemas. Primero, teníamos que guardar en algún sitio el momento en el que una pieza entraba en estado de superposición. Después, este algoritmo resultaba problemático, pues después de colapsar posiciones, de repente el árbol era de profundidad $x-1$. Si íbamos a guardar la profundidad del árbol para seleccionar en qué nivel se tenía que definir la posición, teníamos que estar

constantemente cambiando todos esos punteros que hacían referencia a una profundidad cambiante.

7.4.1 Primer inconveniente, guardar los índices

Para ello creamos la clase Piece y cambiamos la definición de la posición como matriz de piezas. La clase pieza almacena en sí el tipo de pieza que es, el hecho de si es cuántica o no y, en el caso de que esté relacionada con algún suceso cuántico, en qué profundidad del árbol puedo encontrar una referencia al evento. El atributo para el tipo de pieza (pieceClass) sigue la misma notación que en FEN por lo cual es muy sencillo crear un método que nos construya una posición FEN a partir de la matriz de piezas. Pero la notación no podía caer en desuso porque era necesaria para la evaluación y la validez de los movimientos, por lo que desarrollamos un método para calcular la notación FEN a partir de la matriz de piezas.

7.4.2 Segundo inconveniente, profundidades cambiantes

Con respecto a nuestro segundo problema, no es necesario cambiar referencias a profundidades cambiantes si dichas profundidades las volvemos estáticas. Con esta intención cambiamos la estructura del árbol de posiciones para aceptar una nueva clase heredada llamada AbstractPosition. Ahora el árbol sería un árbol binario de 1 o 2 hijos. Los nodos con 1 hijo serían posiciones abstractas, ya que no representan ningún cambio en la estructura de la posición. Los nodos con 2 hijos serían posiciones cuánticas. Por último, los nodos hoja serían posiciones clásicas. Las posiciones abstractas se crean mediante los movimientos normales o mediante la definición de una superposición, mientras las posiciones cuánticas se siguen creando a partir de los movimientos cuánticos.

Al final obtenemos un árbol de N profundidades, siendo N el número de movimientos jugados. Los nodos de profundidad N nunca cambia y si en el futuro, tras M jugadas, queremos comprobar si en la jugada N tuvo lugar una superposición cuántica, podemos acceder a ellos.

También hemos creado un método que nos devuelve la posición descrita según el antiguo esquema sin posiciones abstractas, pues pensamos que podría ser útil en un futuro.

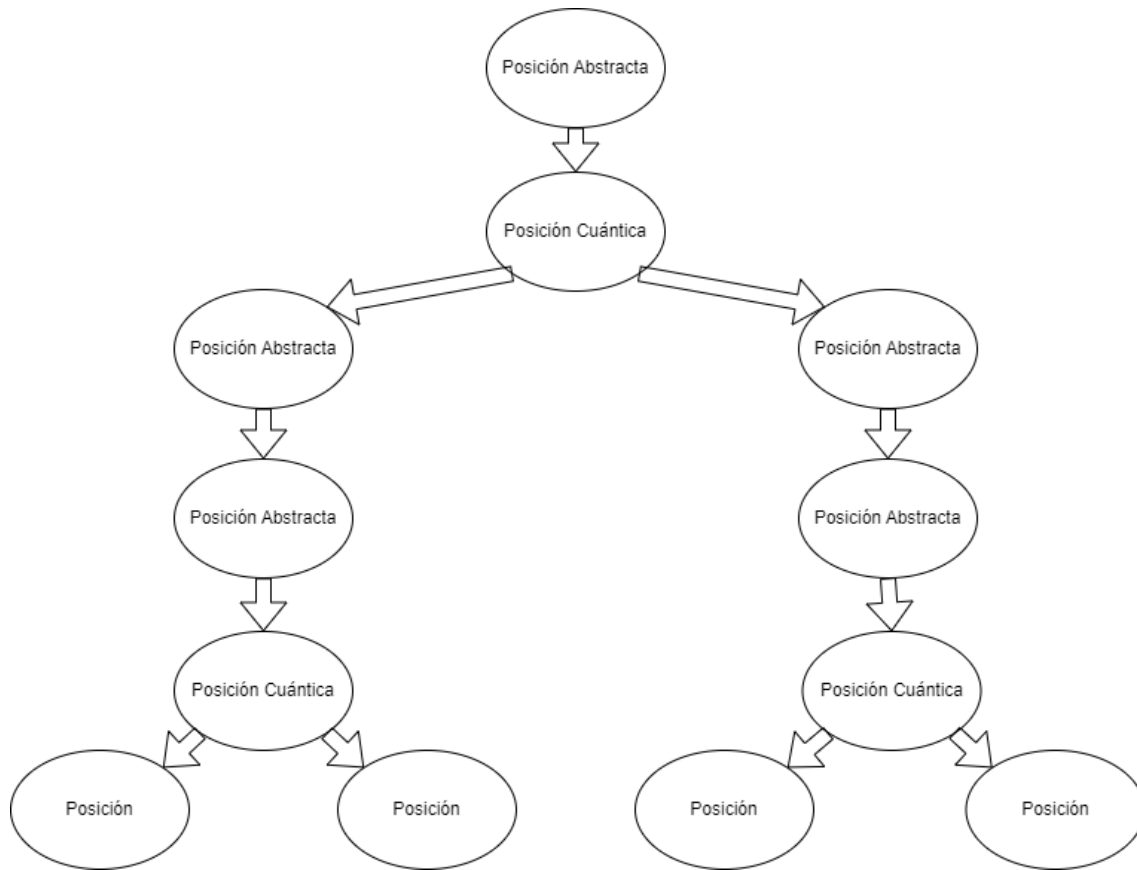


Figura 8: Árbol binario de posiciones. Posiciones abstractas

En la Figura 8 podemos observar el resultado del esquema del árbol de posiciones una vez implementada la posición abstracta. Las posiciones abstractas solo tienen un hijo, las cuánticas tienen dos, y las normales no tienen ninguno.

La nueva estructuración del árbol de posiciones exigía replantear todos los métodos existentes para implementarlos en las posiciones abstractas. En su mayoría tan solo era necesario una línea de código que redirigiera la petición al hijo de la posición abstracta.

Es importante no olvidar en este punto que fue necesario actualizar el método que realizaba el movimiento cuántico para que dejara el sello de la superposición, actualizara la pieza como cuántica y el nodo al que en un futuro el sistema tendría que acudir para resolver la posición.

Continuando con el siguiente proceso, la solución de estados de superposición: Una vez detectadas las colisiones necesarias para que la definición ocurra y conociendo los estados que debemos solventar. Cabe recordar que no es posible crear un estado de superposición a la vez que se resuelve uno. Por lo tanto, no permitimos movimientos cuánticos que realicen capturas. Esto se debe a que contravendría una de las bases de nuestra solución a la variante del ajedrez cuántico, y es que no puede haber dos piezas en una misma casilla en ningún tipo de estado.

```

if move.isCapture(self.position):
    capturer = piece
    captured = self.position.getWhatIsOnSquare(move.move[2:4])
    interferencers = ChessUtils.getInterference(self.position, move.move)

    for indexMove in interferencers:
        choice = random.randint(0, 1)
        self.position = self.position.mergePosition(indexMove, choice)
        if self.moveList[indexMove].move0.flags[0] == "w":
            self.QMovesWhite -= 1 # When merging, we give back the possibility to make more quantum moves
        else:
            self.QMovesBlack -= 1

    if capturer.isQuantic():
        for indexMove in capturer.listMoves:
            choice = random.randint(0, 1)
            self.position = self.position.mergePosition(indexMove, choice)
            if self.flagList[0] == "w":
                self.QMovesWhite -= 1
            else:
                self.QMovesBlack -= 1

    if captured == None: ### ONLY HAPPENS WHEN EN PASSENT
        if move.move[3] == "6" and move.move[1] == "5":
            captured = self.position.getWhatIsOnSquare(move.move[2] + str(5))
        if move.move[3] == "3" and move.move[1] == "4":
            captured = self.position.getWhatIsOnSquare(move.move[2] + str(4))
    if captured.isQuantic():
        for indexMove in captured.listMoves:
            choice = random.randint(0, 1)
            self.position = self.position.mergePosition(indexMove, choice) # (list(reversed(captured.listMoves)))
            if self.flagList[0] == "b":
                self.QMovesWhite -= 1
            else:
                self.QMovesBlack -= 1

```

Figura 9: Algoritmo para la definición de posiciones cuánticas

En la Figura 9 encontramos el fragmento de código que, en caso de detectar una colisión con una pieza cuántica, recorre los índices que le hemos marcado dentro del árbol de posiciones. Por cada índice que tenga guardado deberá fusionar una pareja de posiciones cuánticas. Con la captura al paso ocurre algo especial y es que, aunque el movimiento sea una captura, la casilla objetivo está vacía.

Habiendo detectado la colisión de una pieza en superposición cuántica, obtendremos los índices que nos indicarían en qué punto del árbol de posiciones se encuentra el evento cuántico que dio lugar a esa pieza. Iteraremos por el árbol por todas sus ramas. Al alcanzar la profundidad deseada cambiaremos la posición cuántica resultado de su superposición por una abstracta. De este modo no alteraremos la profundidad total del árbol.

```

def mergePosition(self, move, choice):
    # Input
    #     Integer with an index, representing the depth inside the tree where the position was split
    #     Integer with value 1 or 0 to represent the decision between children
    # Output
    #     Abstract position result of the merge of the nodes in the selected index
    if move == 0:
        raise Exception("You tried to merge an abstract position")
    else:
        return AbstractPosition(copy.deepcopy(self.position).mergePosition(move - 1, choice))

```

Figura 10: Unión de estados, método para posiciones abstractas


```
def mergePosition(self, move, choice):
    # Input
    # Integer with an index, representing the depth inside the tree where the position was split
    # Integer with value 1 or 0 to represent the decision between children
    # Output
    # Abstract position result of the merge of the nodes in the selected index
    if move == 0:
        listPos = [self.position0, self.position1]
        return AbstractPosition(listPos[choice])
    else:
        return QuanticPosition(copy.deepcopy(self.position0.mergePosition(move - 1, choice)), copy.deepcopy(self.position1.mergePosition(move - 1, choice)))
```

Figura 11: Unión de estados, método para posiciones cuánticas

En las Figuras 10 y 11 podemos observar los métodos recursivos que analizan el árbol de posiciones hasta encontrar la posición en la que nos detendremos a unir los estados en una sola posición abstracta.

Durante la unión de estados es muy importante tener en cuenta las siguientes consideraciones:

- Una pieza puede ser “separada” en más de dos superposiciones. Por lo que es posible que en la pieza se definan dos ubicaciones para la resolución. En ese caso, ejecutaremos la búsqueda del siguiente nodo de profundidad para alcanzar este segundo evento y todos los que procedan. Esto se debe ejecutar tanto para la pieza que realiza la captura como para la pieza capturada. Una vez definido el estado, el movimiento de captura se computará si procede.
- Existe la posibilidad de que la pieza en superposición no se encuentre en la casilla que esperábamos como resultado de la definición de su estado. Y es por ello que a veces no tenemos una referencia para modificar el estado de la misma pieza en otros puntos del tablero. Al resolver el estado de esta pieza la debemos marcar como no cuántica de nuevo y los punteros a los eventos de superposición deben desaparecer para que se puedan volver a guardar nuevos eventos en próximos movimientos cuánticos.
- También es necesario comprobar con cada movimiento si el rey se ha movido, si una torre se ha movido o si un peón se encuentra disponible para ser capturado al paso. Es por ello que al final de cada movimiento realizamos una comprobación del estado de todas las piezas del tablero. También guardamos y actualizamos todas las banderas que necesitamos para el correcto transcurso de la partida. Para ello usamos los métodos `deQuantify()` y `updateFlags()` de la clase `Game`.

Conforme el proyecto iba cogiendo forma, era necesario una forma más sencilla de visualizar el tablero, algo inspirado en las referencias conocidas del ajedrez cuántico en las que el tablero no era representado como la múltiple representación de sus subestados, sino como una superposición en sí. Para ello debíamos diferenciar las piezas que se encontraban en superposición con aquellas que no. También era necesario incluir algo más visual que no la inicial de cada pieza.

Decidimos añadir delante de las piezas cuánticas un símbolo que denotara la incertidumbre que el estado de esa casilla y esa pieza suponía. Dentro de los caracteres ASCII, el que encontramos más apropiado fue “~”. Ubicándolo delante de cada una de las piezas cuánticas, nuestro objetivo es marcar piezas cuyo estado no se encuentra definido.

Seguidamente, para hacer una representación de los varios estados de una posición, iteraremos recursivamente por el árbol de posiciones para encontrar qué pieza se encuentra en cada casilla de la matriz. En algunos subestados podíamos encontrar que no había nada, cuando en otros encontrábamos una pieza (piezas cuánticas). El resultado de iterar nos daba una cadena similar a la anterior pero tan solo un tablero en vez de los 2^n tableros representados anteriormente.

Fue un avance importante en la legibilidad de la interfaz consolidar las casillas del tablero delineándolas y añadiendo otros caracteres que complementarían la visualización.

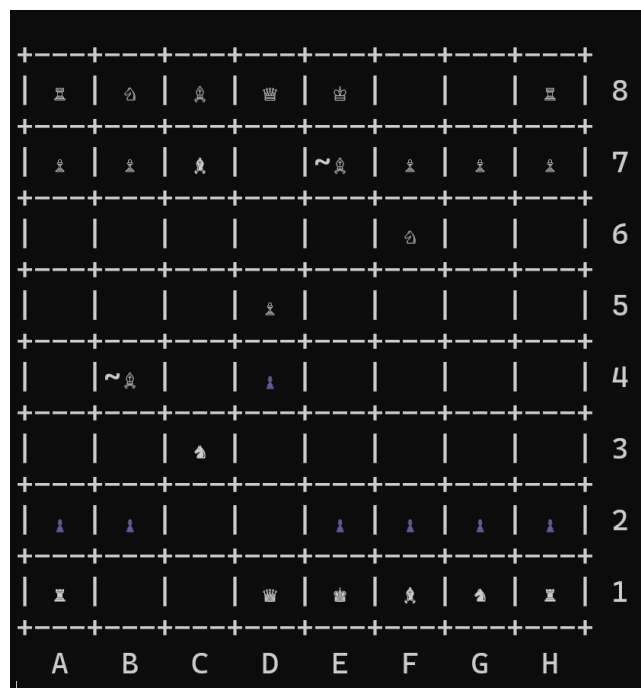


Figura 12: Representación definitiva del tablero

Finalmente, añadimos caracteres Unicode³ para que, en vez de letras, pudiéramos ver las piezas del tablero. En la Figura 12 podemos observar el resultado. Hemos delimitado las casillas y marcado las filas y columnas para la correcta identificación de las casillas.

³ Unicode es un estándar de codificación de caracteres que asigna un número único (llamado punto de código) a cada carácter utilizado en la mayoría de los sistemas de escritura del mundo. Incluye todo tipo de caracteres y símbolos, entre ellos, piezas de ajedrez.

Teniendo una interfaz más sencilla donde podamos visualizar los resultados de nuestras operaciones también facilitaría mucho el trabajo de testeo y depuración de código.

Sin embargo, los problemas durante la fase de test en esta iteración no fueron pocos. Con una estructura cada vez más compleja, y con llamadas cada vez más profundas, la inclusión de las posiciones abstractas supuso un significativo incremento en la dificultad para la depuración de código.

7.5 CUARTA ITERACIÓN

Entre sus objetivos se encontraban marcados:

- Solucionar el entrelazamiento cuántico
- Programar el final de la partida

Éramos conscientes de que la conclusión del juego daría muchos problemas a la hora de validar movimientos, y era una incógnita desde el momento en el que se comenzó el desarrollo. No existía manera alguna de que Stockfish nos diera por válido un movimiento en el que se capturaba un rey. También era complicado que diera por válido los movimientos que capturaban reyes cuánticos.

Por el otro lado, el entrelazamiento suponía un reto igual o mayor. El entrelazamiento se produce cuando una pieza cuántica supone un obstáculo para el desplazamiento de otra pieza. La pieza consta de un estado en el que es un obstáculo, pero también de otro estado en el que no. Esto nos indica que hay un estado en el que podemos pasar y otro en el que no. La pieza puede realizar el movimiento entrelazado para superponerse en dos estados, uno en el que realiza el movimiento y otro en el que no. Y la definición de su estado estará siempre supeditada a la definición de la primera pieza cuántica, y viceversa.

Para que se produjera un entrelazamiento, requeríamos identificar esa pieza que producía la interferencia. Como objetivo, obtener los índices de los eventos que la habían hecho cuántica para asignárselos también a la nueva pieza entrelazada.

Esto resultaba imposible de computar siguiendo el modelo desarrollado hasta el momento en el que las piezas pasaban de estar en un origen para estar en un destino sin marcar un camino. El desplazamiento era tan solo validado por la librería de Stockfish y ésta tan solo nos informa si el movimiento es legal o no. En ningún caso nos informa si el movimiento no era legal debido a una interferencia con otra pieza, por un jaque o por que la pieza no existía en esa casilla para ese subestado de la superposición.

El enlace cuántico se realizaba correctamente a partir de los comportamientos descritos con anterioridad en las primeras tres iteraciones. Y era posible unir los estados correspondientes. Pero era un gran obstáculo identificar qué pieza causaba la interferencia y cuándo se había creado la primera superposición causante del evento.

El único método para encontrar qué obstáculo se situaba en medio durante un movimiento era calcular iterativamente por todas las casillas por las que la pieza ha pasado. A su vez, pensamos que una función que validase los movimientos a partir de su desplazamiento sería la solución perfecta también para los problemas que el final del juego pudiera conllevar.

Finalmente desarrollamos dos procesos distintos, computando en el primero la validez de los movimientos. El método calculaba si el movimiento no era nada fuera de lo estándar, que tuviera un carácter de la 'a' a la 'h', que los números fueran del 1 al 8 o que la pieza no aterrizara en una casilla donde ya hay una pieza del mismo color.

Seguidamente usaba un submétodo específico para cada tipo de pieza y comprobar si su movimiento era correcto, así como que en el desplazamiento no se encontrara con ningún obstáculo.

Su desarrollo fue sencillo, y en poco tiempo pudimos sustituir la función `is_move_correct()` de Stockfish.

De un modo muy similar a los métodos específicos de los movimientos, diseñamos una función de interferencia. Su objetivo era evaluar las casillas que hay entre el origen y destino con un simple bucle. Es importante excluir de este método todos los movimientos de caballo. Son también redundantes los saltos unitarios. Es decir, si entre casilla y casilla solo hay un paso, no puede haber una pieza en medio haciendo interferencia. De este modo podríamos decir que todos los movimientos del rey son redundantes y que no deberían ser computados, pero existe una excepción.

Tan solo hay una casuística en la que el rey puede hacer un movimiento en el que una pieza atraviesa una pieza. Durante el enroque el rey se puede mover dos casillas, pero no puede saltar ninguna pieza porque la casilla que deja en medio es el lugar donde irá colocada la torre. De lo contrario tendríamos dos piezas en una misma casilla y eso contraviene las normas.

Del mismo modo, la torre en el enroque corto salta dos casillas, pero solo dejando hueco en medio para el rey. Es en el enroque largo cuando la torre debe saltar tres casillas, y hay una casilla que, de estar ocupada por una pieza cuántica, podría dar lugar a un entrelazamiento (la casilla b1 para las blancas o b8 para las negras).

Finalmente se creó el archivo `__init__.py` que se ejecutaría al inicio de la aplicación. En él creamos el juego a partir de la clase `Game` e inicializamos variables para el entorno. Describimos un bucle consultando si el juego ha terminado, y durante ese bucle consultaremos qué jugada queremos realizar.

Entre los métodos que diseñamos durante la iteración anterior, recorriamos después de cada jugada si habíamos hecho movimientos que afectaran de algún modo a las banderas. En ellas recorriamos todo el set de piezas. Aprovechamos este bucle para incluir una búsqueda de reyes. Puede darse que haya más de un rey de cada color, pero si alguno de los dos no se repite al menos una vez, significa que ha sido capturado y que ese jugador ha perdido. En ese caso, el bucle alza la bandera que da por finalizado el juego y anota quién es el vencedor.

La cuarta iteración finalizó con resultados altamente positivos. Ya se podían hacer partidas que nos ofrecieran casos de prueba más reales y extensos que los diseñados por el equipo.

7.6 QUINTA ITERACIÓN

En esta quinta iteración nos centramos en:

- Desarrollar las aplicaciones de Stockfish
 - Evaluación de los elementos cuánticos
 - Juego contra Stockfish
- Pincelada final a las funcionalidades de la interfaz de usuario
- Descripción de las normas y comandos de la interfaz.

Durante la primera iteración diseñamos unos algoritmos sencillos de evaluación de la posición y del movimiento. Pero estos pasos fueron tan solo una toma de contacto con la librería de Stockfish y no contemplaban una amplia mayoría de casos que no era capaz de evaluar correctamente.

- Stockfish da error: si el motor de Stockfish no acepta una posición a la que hemos llegado haciendo jugadas válidas definidas por nosotros, es porque el rey está en jaque y es turno del otro jugador. Stockfish lanza una excepción si se encuentra con estas posiciones. Esto significa que el jugador puede capturar el rey y debe devolver un valor muy alto de ventaja.
- La evaluación se hace en cada hoja del árbol de posiciones, y se va rescatando haciendo media aritmética entre las posiciones cuánticas para obtener una noción global del conjunto.

```

def evalPosition(position, flags):
    if isinstance(position, Position.AbstractPosition):
        return ChessUtils.evalPosition(position.position, flags)
    if isinstance(position, Position.QuanticPosition):
        resPos0 = ChessUtils.evalPosition(position.position0, flags)
        resPos1 = ChessUtils.evalPosition(position.position1, flags)
        return round((resPos0 + resPos1) / 2, 2)
    if isinstance(position, Position.Position):
        fen = position.boardToFen(flags)
        try:
            stockfish.set_fen_position(fen)
            result = stockfish.get_evaluation().get("value")/100.0
            return round(result, 2)
        except:
            if flags[0] == "w":
                return 100.0
            else:
                return -100.0

```

Figura 13: Algoritmo para la evaluación de una posición

En la Figura 13 observamos un método actualizado para la nueva estructura del proyecto y del árbol de posiciones. Valora que una posición FEN ilegal es desventajoso para el jugador que hizo el último movimiento. Para asumir esto se basa en que los movimientos legales valorados por Stockfish y los valorados por nuestro sistema son los mismos exceptuando los jaques. Si la posición no es correcta ha de ser por que el sistema ha realizado una jugada que Stockfish considera ilegal, por ende, ha ignorado un jaque y se ha dejado capturar el rey. Entonces la ventaja es para el jugador al que le toca mover.

Por el otro lado, la evaluación del movimiento es ligeramente más compleja, requiriendo más comprobaciones para que se acomodara a la nueva estructura.

Todos los cálculos se ejecutan en las hojas y también está provisto de control para penalizar en caso de que la posición no sea válida. Pero en este caso también debe controlar que el movimiento no sea válido para Stockfish, pero sí para nuestro sistema. Puede ser que no sea válido por que hay un jaque. En ese caso volvemos a penalizar. De no ser así es porque el movimiento (dentro de una posición cuántica) no es válida para esta subposición. En ese caso, para esta subposición es como si perdiéramos el turno. Entonces evaluaremos la posición en caso de que le toque al rival sin nosotros hacer ningún cambio.

Uno de los objetivos más interesantes del proyecto será obtener una recomendación por parte de Stockfish para jugar. Este paso nos acercará a poder jugar una partida contra el motor.

Stockfish cuenta con el método `get_top_moves(int)` en el cual le podemos pasar un entero positivo como atributo y recibir una lista de diccionarios en los que tenemos una recomendación de movimiento y su evaluación.

Estos métodos se ejecutan sobre posiciones clásicas, y su evaluación solo es respecto a la posición clásica de la que proviene. Para poder sacar el máximo provecho de esta información la debemos de tratar. Lo primero es definir el método recursivo y obtener los mejores n movimientos para cada posición. Nosotros hemos definido $n=3$. Una vez extraídos los acumulamos y evaluamos si existe alguna pareja de movimientos recomendados que tengan una misma casilla de origen. Estos son los posibles movimientos cuánticos. Y ya poseemos un método que, dado un movimiento nos devuelve la evaluación a lo largo de todo el árbol. De este modo podemos conseguir una lista de los movimientos más prometedores dentro de la posición cuántica.

Hemos desarrollado para la interfaz las siguientes funcionalidades:

- Hemos añadido a la interfaz las funcionalidades más necesarias.
- Hemos definido una selección de rival, en caso de querer jugar contra un humano o contra la máquina.
- Hemos descrito un comando para seleccionar el color que deseamos jugar.
- Hemos visto acuciante la necesidad de implementar un comando para ayuda de comandos e interfaz y otro para las normas del juego.
 - Comando “Over” para acabar la partida en cualquier momento.
 - Comando “Reset” para reiniciar la partida.

Para cualquiera de estas acciones nos pide una confirmación con el objetivo de no perder por descuido una partida que pueda ser interesante.

- Podemos pedirle a la máquina que nos evalúe la posición o un movimiento.
- Podemos pedirle que directamente nos recomiende un movimiento.
- Por último, detectamos que, cuando introducimos un movimiento, éste sea en el formato correcto y concuerde con una expresión regular descrita previamente para la notación UCI.

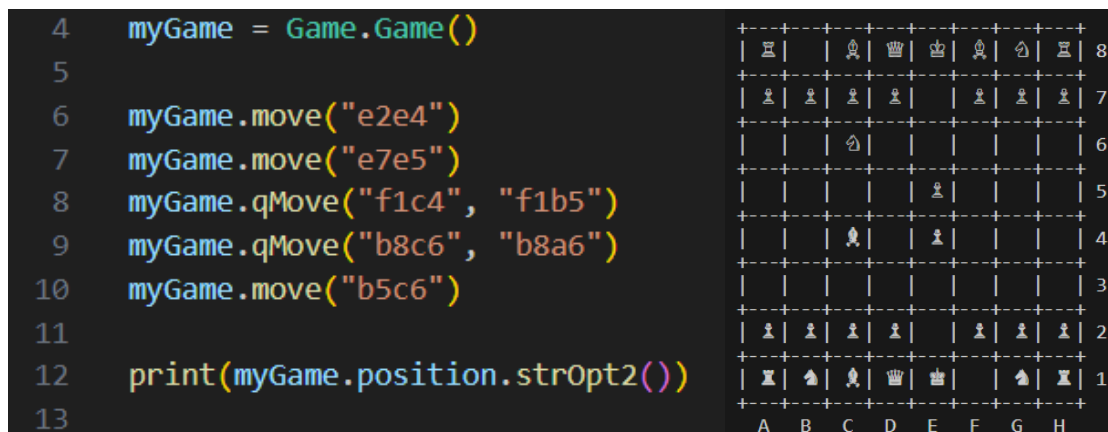
No se conoce registro alguno de proyectos que ostenten hacer uso de un motor de ajedrez para el cálculo de heurísticas en los juegos cuánticos, y este proyecto plantea una primera solución. Sin embargo, muchas de las posibilidades dentro del desarrollo quedaron sin implementar. Algunas de las ideas para la continuación del desarrollo pueden encontrarlas en el capítulo 9 Evaluando con Stockfish.

8 Pruebas

Durante este apartado conoceremos los test empleados para verificar el correcto funcionamiento del sistema. Las pruebas realizadas durante cada iteración se han ido actualizando para ser válidos en todas las versiones y para las últimas implementaciones del sistema.

En el intento de probar todas las combinaciones de entrada posibles, realizamos los siguientes test:

-Figura 14: Se requiere que el sistema pueda recibir jugadas normales y jugadas cuánticas. Además, debe de ser capaz de detectar una colisión y definir el estado de las piezas cuánticas. Debemos contemplar la posibilidad de que tanto la pieza capturada como la que captura sean cuánticas.



```

4  myGame = Game.Game()
5
6  myGame.move("e2e4")
7  myGame.move("e7e5")
8  myGame.qMove("f1c4", "f1b5")
9  myGame.qMove("b8c6", "b8a6")
10 myGame.move("b5c6")
11
12 print(myGame.position.strOpt2())
13

```

8	♔		♚	♖	♗	♘	♙	♜
7	♞	♞	♞	♞		♞	♞	♞
6			♟					
5					♞			
4			♞		♞			
3								
2	♞	♞	♞	♞		♞	♞	♞
1	♜	♙	♚	♗	♖	♘	♙	♜
	A	B	C	D	E	F	G	H

Figura 14: Test Jugada cuántica y definición

-Figura 15: Se requiere poder realizar dos movimientos cuánticos con una misma pieza antes de definirla. La probabilidad de que salga un estado u otro debe corresponder al orden en el que se dividen.


```

4  myGame = Game.Game()
5
6  myGame.move("e2e4")
7  myGame.move("e7e5")
8  myGame.qMove("f1c4", "f1b5")
9  myGame.move("b8c6")
10 myGame.qMove("b5e2", "b5a4")
11 myGame.move("d7d6")
12 myGame.move("a4c6S")
13
14
15 print(myGame.position.strOpt2())

```

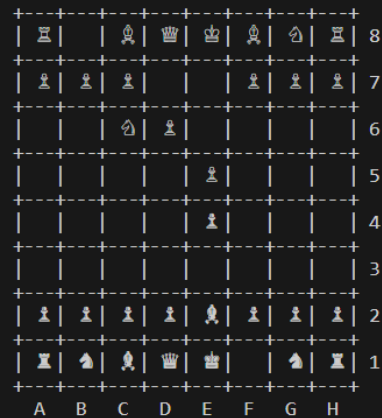


Figura 15: Definición de una pieza con más de dos superposiciones

-Figura 16 y 17: Se requiere poder entrelazar piezas cuánticamente mediante una jugada clásica. Ambas piezas deben ser definidas en caso de que cualquiera de ellas colisione.

```

4  myGame = Game.Game()
5
6  myGame.move("e2e4")
7  myGame.move("e7e5")
8  myGame.qMove("d2d4", "d2d3")
9  myGame.move("d7d6")
10 myGame.move("f1c4")
11 myGame.move("e5d4")
12
13
14 print(myGame.position.strOpt2())

```

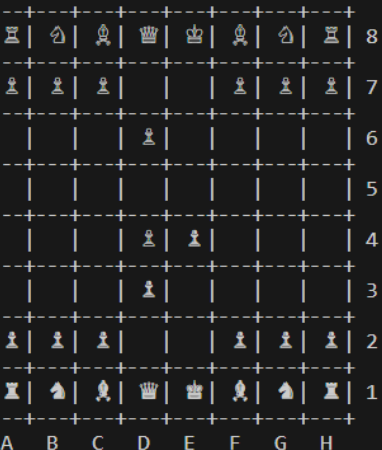


Figura 16: Definición de una pieza entrelazada cuánticamente. Opción de captura A

```

4  myGame = Game.Game()
5
6  myGame.move("e2e4")
7  myGame.move("e7e5")
8  myGame.qMove("d2d4", "d2d3")
9  myGame.move("c7c6")
10 myGame.move("f1b5")
11 myGame.move("c6b5")
12
13
14 print(myGame.position.strOpt2())

```

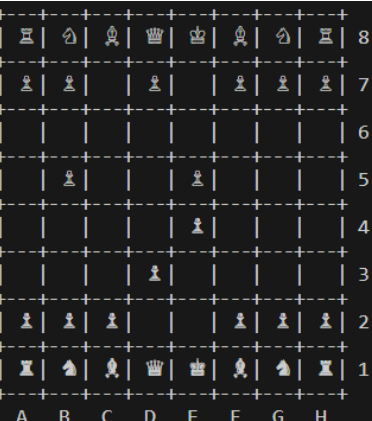


Figura 17: Definición de una pieza entrelazada cuánticamente. Opción de captura B

Figura 18: Se requiere el uso de los llamados movimientos complementarios y su correcta interacción con los eventos cuánticos.

```

16  myGame.move("h7h6")
17  myGame.move("g1f3")
18  myGame.qMove("a7a6", "a7a5")
19  myGame.qMove("e1c1", "e1g1")
20  myGame.move("a5a4")
21  myGame.move("b2b4")
22  myGame.move("a4b3")
23
24  print(myGame.position.strOpt2())
25

```

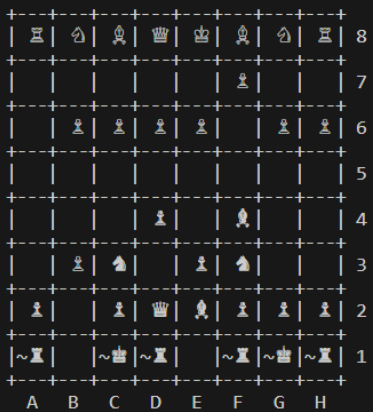


Figura 18: Movimientos complementarios, el enroque y la captura al paso

-Figura 19: Se requiere un sistema de detección eficaz para la finalización de la partida. Es necesario un disparador que finalice el juego si el rey es capturado. El sistema debe informar del jugador vencedor.

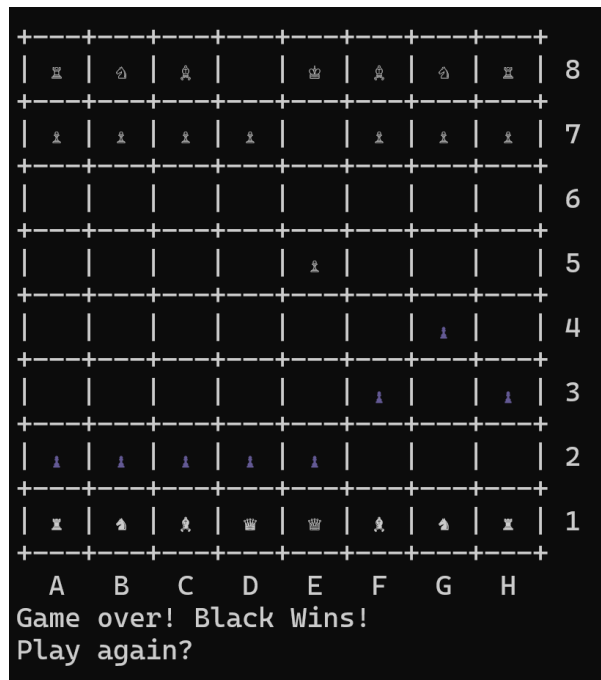


Figura 19: Ejemplo de finalización de partida

Eran muchos los errores que obteníamos al principio por aspectos que escapaban de nuestro control. Cabe mencionar un error ciertamente curioso que tuvo lugar como consecuencia de la primera implementación de la captura al paso. Cuando una captura al paso se realiza, primero valoramos si es válido el movimiento usado Stockfish. Debíamos asegurarnos de que fuera una captura según el motor. Después realizábamos el movimiento de forma recursiva a través del árbol de posiciones. Al finalizar el movimiento, realizamos el movimiento complementario descrito en la iteración.

La única condición para realizar el movimiento era que sucediera en determinadas casillas, en la fila 3 o 6, que la pieza que realizara el movimiento fuera un peón, y que la casilla objetivo estuviera vacía. Al fin y al cabo, antes habíamos comprobado que fuera una captura y las únicas capturas a casillas vacías son las capturas al paso. Nuestra sorpresa fue mayúscula durante una prueba de un caso real, jugando una partida en Beta, al observar el insospechado fenómeno.

El jugador con negras había colocado un caballo cuántico en la casilla f3. Al capturarlo con el peón de g2, el peón de f2 apareció cuantificado. A la conclusión a la que pudimos llegar era que la captura, por alguna razón había sido considerada como una captura al paso, eliminando la pieza detrás de sí. Esto se debe a que, dentro del árbol de posiciones, había subposiciones que cumplían con las tres condiciones previas. La pieza era un peón, se encontraba en las casillas determinadas y su objetivo era una casilla vacía. En realidad, no estaba vacía solo que, en esa subposición, el caballo no se encontraba allí.

Otro error habitual era capturar una pieza resultado de un enlace cuántico y perder en algún momento los índices que nos indicaban cuando unir las piezas en el árbol de posiciones. En algunos momentos se perdían al sacar copias de objetos sin cuidar todas las variables, en otros casos era por que la función de detección de interferencia tenía pequeños fallos.

Estos últimos eran los más difíciles de detectar porque en muchos casos funcionaba correctamente y detectaba la colisión con piezas cuánticas. Pero en casos muy aislados y en movimientos muy específicos, estos movimientos no hacían interferencia y no guardaban los índices para luego ser definidos.

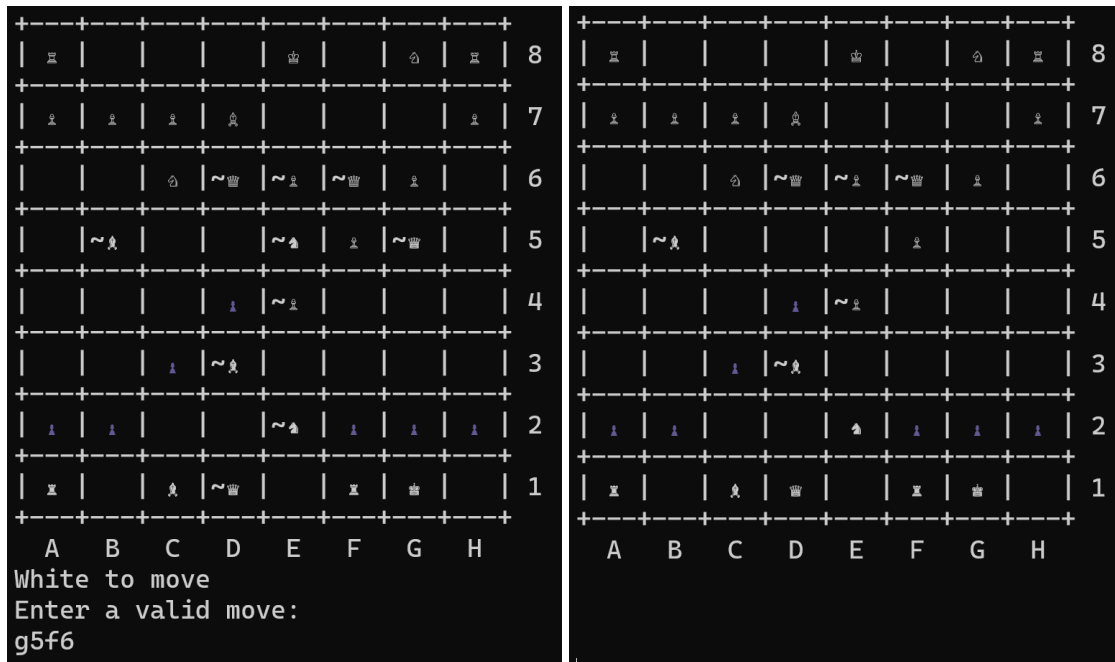


Figura 20: Dos damas resultantes de movimientos cuánticos

En la Figura 20 hemos insertado el movimiento “g5f6” en el que la dama cuántica blanca captura la dama cuántica negra. Las dos han surgido resultado de un entrelazamiento cuántico y están ligadas a otra pieza. Mas la dama blanca se define sin problema mientras la negra no sufre ningún cambio tras la observación. Tras una debida depuración, concluimos con que la detección de interferencia durante el movimiento horizontal de la dama no estaba correctamente descrita. Sin embargo, para los movimientos diagonales sí, y eso hacía que una dama se comportara diferente a la otra.

Somos conscientes de que el testeo de un producto es el más complicado de probar, pues incluso cuando no se encuentran fallos, todavía pueden aparecer más en casos muy aislados. Por ello hemos realizado decenas de pruebas de partidas entre desarrolladores y “testers”, con el fin de encontrar el máximo número posible de excepciones que salgan del control del programa. Los más significativos los hemos podido plasmar aquí, pero otros muchos no han recibido la misma atención pues han sido resueltos en minutos.

9 EVALUANDO CON STOCKFISH

En este capítulo no solo se pondrá sobre la mesa la viabilidad del uso de Stockfish y los métodos empleados, sino también opciones alternativas y posibilidades para un futuro desarrollo. Entre las alternativas, muchas fueron descartadas por la falta de tiempo o de recursos pero quedarán pendientes en próximas versiones del programa.

La herramienta para la evaluación de posiciones en el ajedrez clásico que supone Stockfish ha demostrado en innumerables ocasiones ser una de las más fuertes del mercado. El hecho de ser un motor de libre uso abre muchas posibilidades para trastear en la línea de fuego, en lo más avanzado y potente del cálculo ajedrecístico.

En el panorama del estudio de mecánicas de juego, en el intento de resolver los juegos de estrategia, el uso de algoritmos de optimización es bien extendido. Entre las opciones valoradas por el equipo se encuentran los algoritmos de Minimax y Monte Carlo Tree Search.

Minimax es un método de decisión centrado en minimizar la pérdida máxima que el oponente pueda causar. Para su uso debemos suponer que el juego es de suma cero e información perfecta. Suma cero quiere decir que la ganancia o pérdida de un jugador puede mantener el juego equilibrado si el rival gana o pierde en igual medida. Un juego con información perfecta es aquel que en todo momento los jugadores tienen acceso completo y perfecto a toda la información relevante del mismo.

Este último requisito es devastador para la correcta implementación de Minimax en el contexto actual. Es indispensable conocer todos los posibles movimientos y evaluarlos. En el ajedrez clásico ya es complicado obtener todos los movimientos posibles. Pero en el ajedrez cuántico, la dificultad es exponencialmente mayor dada las elevadas posibilidades combinacionales de los movimientos cuánticos. Añadido a lo anterior, se encuentra el aspecto azaroso de las superposiciones cuánticas.

Existen variantes de Minimax que no exigen información perfecta. Dos ejemplos pueden ser el Minimax con corte limitado (Limited Depth Minimax) o Minimax con probabilidad (Minimax with Chance). Minimax con corte limitado no finaliza la búsqueda en profundidad, pero continúa requiriendo una completa información de todos los movimientos posibles. Minimax con probabilidad es perfecto para juegos de azar y sería capaz de solucionar el aspecto azaroso de las superposiciones.

Por el otro lado, Monte Carlo realiza pequeñas evaluaciones de los estados antes de continuar buscando en profundidad. Puede ser un buen objeto de investigación para futuras implementaciones, pero el uso de Stockfish ofrecía resultados más rápidos frente a la falta de tiempo para el desarrollo de nuevas heurísticas.

Durante la fase de diseño se plantearon varias estrategias que pudieran aprovechar la fiereza de la capacidad de cálculo del motor. De este planteamiento surgieron las siguientes dudas:

- ¿Cómo evaluar una posición cuántica?
- ¿Cómo evaluar un movimiento cuántico?
- ¿Qué hacer para que un movimiento cuántico sea valorado por encima de los movimientos clásicos?

Como hemos explicado durante el desarrollo, el escenario sería una posición cuántica en la que tiene lugar la superposición de una pieza en dos diferentes estados. Existe un 50% de posibilidades de que se encuentre en un estado y otro 50% para el complementario. Por ello, en un caso general existe una posibilidad de que la ventaja este definida por el primer estado y otra posibilidad en la que la ventaja está definida por un segundo estado.

Nuestro desarrollo se ha basado en ese principio, pero nuestras mentes han ido más allá. ¿Es posible que esa norma no se cumpla? Para demostrarlo tan solo es necesario encontrar una situación en la que se cumpla.

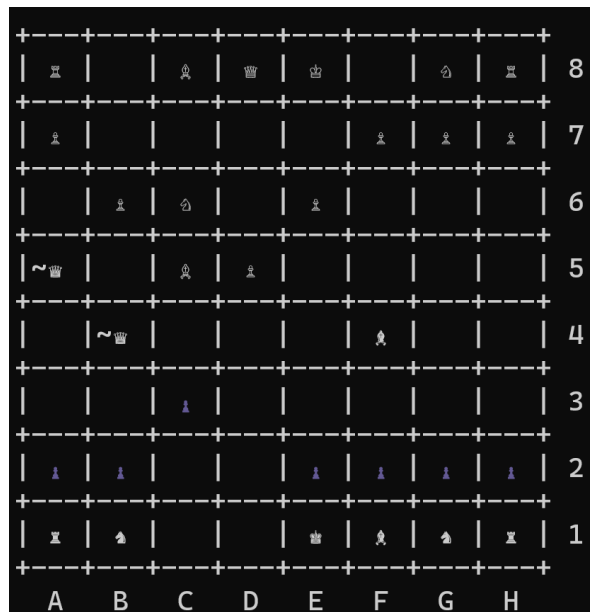


Figura 21: Ejemplo evaluación

En la posición mostrada en la figura 21, las blancas han superpuesto la posición de su dama en dos diferentes estados. Las dos damas se encuentran bajo ataque y el blanco muy seguramente querrá quitarla de allí.

Sin embargo, no va a ser capaz de retirar ambas damas en un solo turno. El blanco se encuentra en problemas, pues tiene un 50% de probabilidades de perder la dama.

Si evaluamos las posiciones por separado ambas nos dan ligera ventaja negra. Pero nada que ver con la ventaja que dan si no retiramos la dama del peligro.

Este tipo de peligros son indetectables por el método aplicado. Una posible solución sería evaluar la posición a partir de las mejores jugadas para cada posición cuántica. La evaluación del ajedrez clásico de Stockfish nos da una orientación de la valoración en el caso de que ambos jugadores jueguen al máximo nivel. Por ello la evaluación solo se corresponde a si hacemos la mejor jugada. Si jugamos al ajedrez cuántico, es posible que la mejor jugada para un subestado no sea buena en otros subestados.

9.1 Un método para evaluar estados cuánticos:

El ajedrez en su más alto nivel lo podemos definir como la búsqueda del fallo del oponente. Está probado que un juego sin errores debe irrefutablemente conllevar a las tablas. Y en este aspecto, los ordenadores dejan poco margen de maniobra. Si una jugada es, por insignificante que sea el detalle, peor que las recomendadas, sabrán sacar provecho del factor desequilibrante. De este modo, durante el turno de cada jugador, la evaluación de la posición no puede inclinarse a su favor, solo en su contra. Esto, por supuesto, es distinto en la práctica, pero no muy alejado de la realidad. Si nosotros nos encontramos con una ventaja en el tablero, es siempre porque el oponente se ha equivocado en algún punto.

Una jugada cuántica nunca va a ser la elección favorita de Stockfish. Hasta ahora, nosotros hemos definido la evaluación de una jugada cuántica como la media entre la evaluación de una jugada A y una jugada B. Si la jugada A tiene puntuación X y la jugada B tiene puntuación Y, la media de las dos puntuaciones nunca será superior a ambos movimientos individuales.

Pero, ¿en qué caso una jugada cuántica es buena?

Primero, dado que el juego no deja de ser ajedrez y se rige por sus normas básicas, el movimiento cuántico debe estar formado por dos jugadas que sean buenas independientemente. Será difícil obtener un movimiento cuántico que mejore la evaluación de dos jugadas bien escogidas por Stockfish.

Pero el juego no deja de depender del azar. En el siguiente escenario usaremos un movimiento cuántico hipotético AB por el jugador 1, creado a partir de los movimientos clásicos A y B. $F(P_A)$ será la puntuación del movimiento A y $F(P_B)$ la del movimiento B.

La posición resultante es la superposición de los estados P_A y P_B . Tras realizar el movimiento A, Stockfish recomienda al jugador 2 jugar el conjunto $H_A[]$. Y tras realizar el movimiento B, recomienda $H_B[]$. Podemos entonces deducir que la posición P_A mantendrá el valor $F(P_A)$ si el oponente juega alguno de los valores de $H_A[]$. Sin embargo, esta afirmación no está asegurada si el jugador 2 juega algún elemento del conjunto $H_B[]$. El valor no puede ceder más en favor del jugador 2 durante su propio turno, como hemos explicado antes. Por lo cual toda variación en la evaluación es a favor del jugador 1.

En conclusión, una jugada cuántica AB es buena si para la posición resultante de A son malas las opciones que el oponente obtiene para la posición B y viceversa.

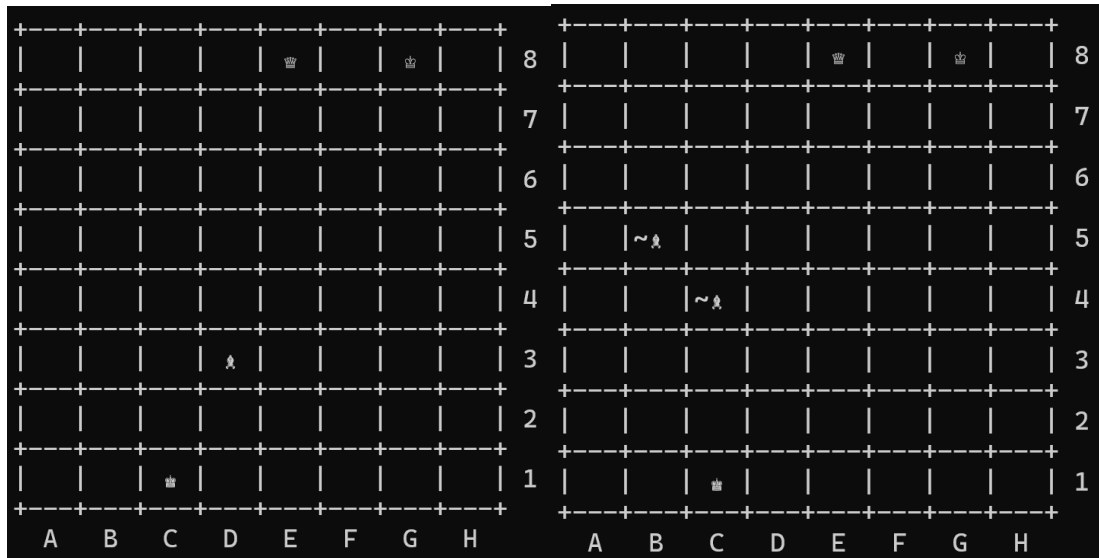


Figura 22: Movimiento cuántico y propuesta de valoración

La Figura 22 muestra un claro ejemplo del caso. La evaluación para el blanco en el caso de que el alfil esté en cualquiera de las dos posiciones es mala, y Stockfish computará que está abocado a perder. Sin embargo, el movimiento cuántico le brinda al rey blanco una última oportunidad.

El negro ahora se enfrenta al dilema. Debe salvar a su rey del ataque cuántico o a la dama. Mueva cual mueva tiene un 50% de posibilidades de perder la otra pieza.

Tras el movimiento de las blancas, un movimiento de rey da un 50% de posibilidades de perder la dama, y otro 50% de salvarla y seguir jugando con ventaja. Mover la dama son un 50% de posibilidades de perder el rey y por tanto la partida, y otro 50% de seguir jugando con ventaja.

Podemos concluir de forma teórica y empírica que la evaluación puede asemejarse más a la siguiente fórmula que al método de la media aritmética simple:

$$F(P_0, AB) = \frac{F(P_A) + F(P_B) + F(P_A, H_B[]) + F(P_B, H_A[]) }{4}$$

En definitiva, la evaluación de los movimientos y de las posiciones cuánticas van de la mano. Este método recursivo se adentrará en posiciones P_A y P_B para calcular cuales son las mejores jugadas contra A y las mejores jugadas contra B. Después, evaluará jugar las jugadas resultantes de A en la posición B y viceversa.

La técnica para acercarnos un poco más a una evaluación óptima es hacer las siguientes preguntas: ¿Es la mejor opción que él tiene contra A, mala jugada contra B? ¿Y al revés? ¿Existe una jugada que pueda contrarrestar ambas jugadas?

9.2 Las infinitas soluciones de las posiciones cuánticas

Este ejemplo en particular nos ha parecido merecedor de estudio porque ejemplifica la infinidad de posibilidades que esta variante nos brinda, y es que todavía existe una forma de la que las negras podrían ver sus opciones de victoria ampliadas significativamente, y no arriesgar al 50/50.

Hemos hallado un interesante ejemplo en el que podemos darle la vuelta a la posición una vez más mediante un movimiento cuántico. Hay una jugada de dama que merece la pena valorar: De3, Df7.

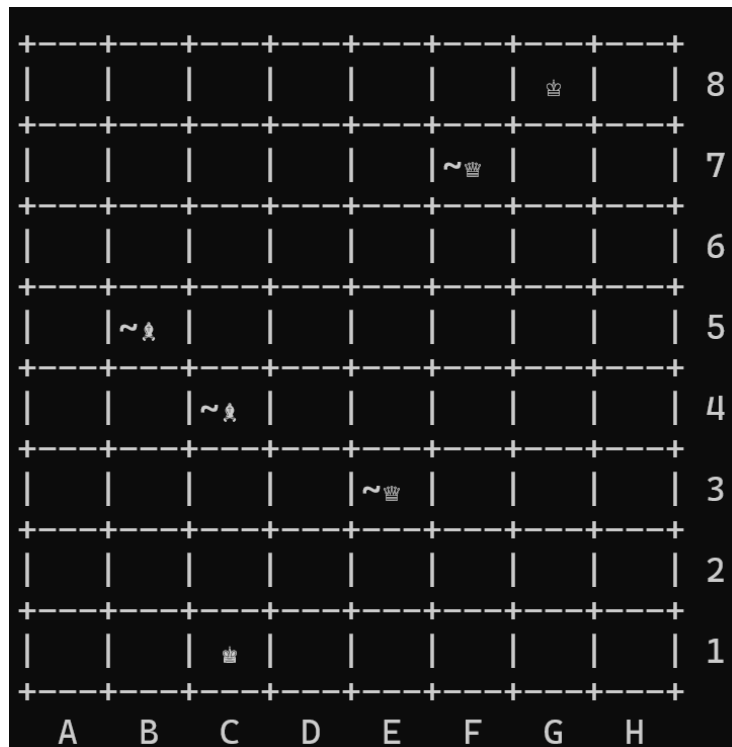


Figura 23: Problemas de la evaluación

Esta jugada mostrada en la figura 23 cambia un poco las posibilidades. El blanco sigue teniendo un 50% de posibilidades de que el alfil esté en b5 y entonces las negras continuarían con ventaja. Por otro lado, tienen un 50% de posibilidades de que se encuentre en c4 y entonces debe tomar una importante decisión. Si captura en f7 hay un 50% de posibilidades de que la dama esté en esa casilla y entonces igualar la posición, o podría estar en e3 y entonces el blanco perdería el rey. Por el otro lado, si captura en g8 tiene una posibilidad de capturar el rey enemigo y ganar o que la dama este en medio invalidando esa jugada y regalando el alfil. Ahora el blanco tiene un 25% de posibilidades de ganar y un 75% de continuar jugando en desventaja o perder.

La falta de tiempo entre otras cosas nos ha llevado a hacer simplemente este estudio teórico de la posibilidad de evaluación y no su implementación. Pero creemos que su desarrollo incrementaría notablemente los resultados generados por el sistema y la calidad de sus cálculos.

El gran problema del uso de Stockfish y que requerirá una inversión de tiempo para poder gestionar la casuística necesaria es cuando llega el final de la partida. Stockfish no devuelve una puntuación en el momento en el que ve venir un jaque mate. Si ese es el caso, Stockfish nos devolverá un valor para "Mate". Este valor son las jugadas restantes hasta que podamos hacer mate.

En el ajedrez cuántico, el jaque mate no existe, pero es ciertamente probable que, si tienes mate en pocas jugadas, es muy probable que tengas una oportunidad de comerte el rey rival. Es posible plantear heurísticas en las que se valoren las estadísticas de los diferentes eventos cuánticos.

La idea sería hacer algo muy similar a lo que hacemos con la penalización por jugadas ilegales dentro de la valoración de los movimientos durante nuestra quinta iteración. Una posible función para calcular un beneficio a partir de tener un mate cerca podría ser la siguiente:

$$F(P) = \frac{a}{N} * (-1^m)$$

Siendo $F(P)$ la función evaluación para la posición P, N el número de jugadas restantes para el mate, 'a' un número entero como penalización y 'm' el número de movimientos realizados en el juego. N será positivo si al jugador al que le toca es el que tiene la opción de dar el mate. Si al blanco le toca y se encuentra en disposición de dar jaque mate, N será positivo y m será par. Si al negro le toca, pero es el blanco quien puede dar jaque mate, N será negativo y m será impar. Su producto será positivo marcando la ventaja blanca. N se encuentra en el divisor de modo que, cuanto más cerca se encuentre el mate, mayor será la penalización para el rival.

Por supuesto hay muchos más casos a considerar, todavía queda mucho por investigar y mucho por mejorar. Los algoritmos aquí descritos solo son un paso previo a la posterior implementación en un futuro cercano de heurísticas más eficientes y acertadas para los juegos cuánticos.

10 MANUAL DE USUARIO

El presente apartado pretende servir como guía para el desarrollador que desee conocer el proyecto y comprobarlo, pero también para el usuario que sienta curiosidad y decida probar la jugabilidad. Por ello, su interfaz no requiere de conocimientos en el campo de la informática o de la física cuántica, y tan solo se recomienda conocer las normas del ajedrez clásico para poder hacer uso de la herramienta.

La interfaz está completamente en inglés, es necesario un mínimo nivel para la comprensión de la interfaz, aunque no es requisito exclusivo.

La interfaz se basa exclusivamente en una consola de comando, por lo cual, toda interacción con el sistema deberá ser redactada a través de la misma. Definimos una serie de ordenes que guiarán al sistema antes del inicio del juego. Entre estas, debemos seleccionar el oponente, el color, etc.

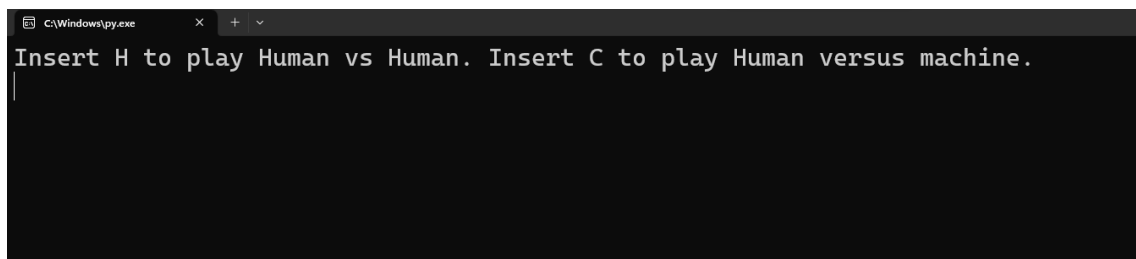


Figura 24: Pantalla de inicio. Selección de oponente

En una primera instancia, como podemos observar en la Figura 24, debemos seleccionar si queremos una partida contra la máquina o contra otro humano (o visto de otro modo, dos usuarios en local). Para ello insertaremos la “H” de “Human” o la “C” de “Computer”.

Si hemos elegido jugar contra Stockfish, entonces necesitaremos elegir un color. Si hemos elegido jugar contra otro humano en local, no será necesario ya que cada jugador insertará sus movimientos de forma manual a través de la consola cuando sea su turno.

Para seleccionar el color, el sistema nos pedirá que insertemos “W” de “White” para jugar con blancas o “B” para “Black”, jugar con negras.

Si hemos elegido negras, Stockfish jugará directamente su primer movimiento y nos encontraremos un tablero en el que las blancas han comenzado jugando. Una vez el

juego es iniciado, podemos hacer uso del comando “Help” para solicitar por texto todos los comandos disponibles y su función.

```

C:\Windows\py.exe
Insert H to play Human vs Human. Insert C to play Human versus machine.
C
Insert W to play white or B to play black
W
Type 'Help' + Intro for command details

+---+---+---+---+---+---+---+
|  ♖  |  ♜  |  ♞  |  ♝  |  ♞  |  ♜  |  ♖  | 8
+---+---+---+---+---+---+---+
|  ♙  |  ♙  |  ♙  |  ♙  |  ♙  |  ♙  |  ♙  | 7
+---+---+---+---+---+---+---+
|  |  |  |  |  |  |  | 6
+---+---+---+---+---+---+---+
|  |  |  |  |  |  |  | 5
+---+---+---+---+---+---+---+
|  |  |  |  |  |  |  | 4
+---+---+---+---+---+---+---+
|  |  |  |  |  |  |  | 3
+---+---+---+---+---+---+---+
|  ♚  |  ♚  |  ♚  |  ♚  |  ♚  |  ♚  |  ♚  | 2
+---+---+---+---+---+---+---+
|  ♟  |  ♟  |  ♟  |  ♟  |  ♟  |  ♟  |  ♟  | 1
+---+---+---+---+---+---+---+
  A  B  C  D  E  F  G  H
White to move
Insert a valid move:

```

Figura 25: Inicio del juego

```

C:\Windows\py.exe
+---+---+---+---+---+---+---+
|  |  |  |  |  |  |  | 5
+---+---+---+---+---+---+---+
|  |  |  |  |  |  |  | 4
+---+---+---+---+---+---+---+
|  |  |  |  |  |  |  | 3
+---+---+---+---+---+---+---+
|  ♚  |  ♚  |  ♚  |  ♚  |  ♚  |  ♚  |  ♚  | 2
+---+---+---+---+---+---+---+
|  ♟  |  ♟  |  ♟  |  ♟  |  ♟  |  ♟  |  ♟  | 1
+---+---+---+---+---+---+---+
  A  B  C  D  E  F  G  H
White to move
Insert a valid move:
help
To write a move, write the UCI notation for the movement:
I.e. To move any piece from the square b5 to the square d3 write 'b5d3'.
If you want to move it quantic, to d3 and e2 write 'b5d3, b5e2'

Write 'Rules' to display the set of rules for Quantum Chess
Write 'Over' anytime to finish the game, or 'Reset' to restart it

You can use Stockfish engine to help you move, rate your position, etc.
Typing 'Evaluate', you will receive an evaluation of the current position,
as a number that can be positive or negative
Higher number means white is winning, and lower (negative) numbers mean black is winning
Type 'Evaluate move', and after type your move, to receive a number telling how good the move is
Type 'Hint' to ask Stockfish what to move in the current position

```

Figura 26: Comando HELP

En la Figura 25 observamos la pantalla de juego. La interfaz de juego nos muestra en todo momento la posición actual y quién tiene posesión del turno de juego. Cualquier entrada de texto se considera proveniente de ese jugador. En la Figura 26 vemos el resultado de solicitar ayuda mediante el comando “HELP”.

Los comandos no son “case sensitive”, excepto por los movimientos. Se espera que los movimientos escritos en notación UCI usen letras minúsculas para las columnas o para las piezas que transformamos al coronar.

El comando “Rules”, mostrado en la Figura 27 nos describe por pantalla las normas del ajedrez cuántico y sus diferencias con el ajedrez clásico. Es vital destacar que la descripción de las normas para esta variante del juego se basa en las normas del juego original, y será necesario tener una mínima noción de las mismas para entenderlo.

```

White to move
Insert a valid move:
rules
Quantum Chess is a variant of the classical chess game adding some Quantum properties.
In the game, sometimes pieces can be two places at once, the known as quantum superposition.
This happens when you make a quantum move. Quantum moves are the result of two classical
moves at once, so we no longer know the precise position of this piece. Piece's position
is defined when an observation happens. In this case, when someone captures a quantum
piece or when a quantum piece captures another. Then we have 50% chance to be on each
different position. This allows quantum entanglement. If a quantum piece has a certain
probability to be at one place, means that there is a probability that it is not there and
I can pass through it with a slide move. In this case the piece may have done the movement
or not, it depends on the final position of the previous quantum piece. It is entangled.
To win, you must capture the enemy king.

+---+---+---+---+---+---+---+---+
| x | o | x | w | d | x | o | x | 8
+---+---+---+---+---+---+---+---+
| x | x | x | x | x | x | x | x | 7
+---+---+---+---+---+---+---+---+
| | | | | | | | | 6
+---+---+---+---+---+---+---+---+
| | | | | | | | | 5
+---+---+---+---+---+---+---+---+
| | | | | | | | | 4
+---+---+---+---+---+---+---+---+
| | | | | | | | | 3
+---+---+---+---+---+---+---+---+
| x | x | x | x | x | x | x | x | 2
+---+---+---+---+---+---+---+---+
| x | x | x | w | w | x | x | x | 1
+---+---+---+---+---+---+---+---+
  A  B  C  D  E  F  G  H

White to move
Insert a valid move:

```

Figura 27: Comando RULES

El sistema reconoce por un lado el comando “Evaluate” y por otro, “Evaluate move”

El primero nos evaluará la posición y nos otorgará una puntuación, un número entero de dos decimales, marcando la ventaja que posee el jugador blanco. Esto significa que, si recibimos un número positivo, es ventaja para el jugador blanco. Si el número en contraposición es negativo, la ventaja será del jugador negro. Así mismo, qué tan grande sea el número indica qué tan desbalanceada se encuentra la partida.

El segundo nos evaluará un movimiento, y siguiendo los criterios del comando anterior, nos evaluará que tan buena es la posición resultante de hacer determinado movimiento. Debemos insertar primero el comando “Evaluate move” y pulsar intro. Inmediatamente se nos pedirá que introduzcamos un movimiento, igual que cuando nos encontramos en posición de jugar.

El comando “Hint”, como vemos en la Figura 28, nos recomendará un conjunto de opciones a jugar. Nos entregará una serie de movimientos y la puntuación que ellos reciben tras ser evaluados. Como podemos observar en la Figura X, hay tres opciones

que se componen de movimientos clásicos, pues solo poseen un movimiento en notación UCI y la puntuación asignada. Por otro lado, hay un movimiento compuesto de dos movimientos clásicos y una puntuación.

```

White to move
Insert a valid move:
hint
[('c1g5', 0.38), ('c1g5, c1f4', 0.28), ('c1f4', 0.16), ('g1f3', 0.14)]

+---+---+---+---+---+---+---+---+
|  ♖  |  ♜  |  ♞  |  ♝  |  ♞  |  ♝  |  ♞  |  ♝  | 8
+---+---+---+---+---+---+---+---+
|  ♚  |  ♚  |  ♚  |  ♚  |  ♚  |  ♚  |  ♚  |  ♚  | 7
+---+---+---+---+---+---+---+---+
|  ♙  |  ♙  |  ♙  |  ♙  |  ♙  |  ♙  |  ♙  |  ♙  | 6
+---+---+---+---+---+---+---+---+
|  ♗  |  ♗  |  ♗  |  ♗  |  ♗  |  ♗  |  ♗  |  ♗  | 5
+---+---+---+---+---+---+---+---+
|  ♛  |  ♛  |  ♛  |  ♛  |  ♛  |  ♛  |  ♛  |  ♛  | 4
+---+---+---+---+---+---+---+---+
|  ♜  |  ♜  |  ♜  |  ♜  |  ♜  |  ♜  |  ♜  |  ♜  | 3
+---+---+---+---+---+---+---+---+
|  ♞  |  ♞  |  ♞  |  ♞  |  ♞  |  ♞  |  ♞  |  ♞  | 2
+---+---+---+---+---+---+---+---+
|  ♝  |  ♝  |  ♝  |  ♝  |  ♝  |  ♝  |  ♝  |  ♝  | 1
+---+---+---+---+---+---+---+---+
  A   B   C   D   E   F   G   H
White to move
Insert a valid move:
|

```

Figura 28: Comando HINT

El juego acabará en cuanto uno de los jugadores capture al rey rival o cuando se inserte uno de los comandos, “Reset” u “Over”. En caso de haber capturado el rey, la interfaz mostrará el jugador vencedor y propondrá la posibilidad de comenzar otra partida. En caso de usar el comando “Over”, veremos el juego acabar del mismo modo y nos hará la misma pregunta. En cualquier caso, podemos escribir “yes” para reiniciar o cualquier otra cadena de texto para cerrar la aplicación.

El ligero cambio respecto al comando “Reset” es que este no da la opción de cerrar la aplicación y reinicia directamente el juego. Ambos comandos, tanto “Reset” como “Over” te preguntan para confirmar que quieres acabar el juego actual. Para proceder, introducimos “yes”. Cualquier otra cadena de caracteres continúa el juego.

11 Bibliografía

- Akl, S. G. (2010). ON THE IMPORTANCE OF BEING QUANTUM. *Parallel Processing Letters*, 20(03), 275-286. <https://doi.org/10.1142/s0129626410000223>
- Akl, S. G. (2016). THE QUANTUM CHESS STORY. *Technical Report* No. 2016-629, <https://research.cs.queensu.ca/home/akl/techreports/story.pdf>.
- Cantwell, C. D. (2017). Quantum Chess: Making Quantum Phenomena Accessible. *Bulletin of the American Physical Society*.
http://absimage.aps.org/image/MAR17/MWS_MAR17-2016-000893.pdf
- Cantwell, C. D. (2019). Developing a Mathematical Framework and Design Methodology for Creating Quantum Games. *arXiv*.
<https://arxiv.org/pdf/1906.05836.pdf>
- Colaboradores de Wikipedia. (2021). *Árbol de búsqueda Monte Carlo*. Wikipedia, la enciclopedia libre.
https://es.wikipedia.org/wiki/%C3%81rbol_de_b%C3%BAsqueda_Monte_Carlo
(fecha de consulta: 7 de junio de 2023)
- Colaboradores de Wikipedia. (2021). *Interfaz Universal de Ajedrez*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Interfaz_Universal_de_Ajedrez
(fecha de consulta: 30 de mayo de 2023)
- Colaboradores de Wikipedia. (2022). *Símbolos de ajedrez en Unicode*. Wikipedia, la enciclopedia libre.
https://es.wikipedia.org/wiki/S%C3%ADmbolos_de_ajedrez_en_Unicode (fecha de consulta: 16 de junio de 2023)
- Colaboradores de Wikipedia. (2023). *Ajedrez cuántico*. Wikipedia, la enciclopedia libre.
https://es.wikipedia.org/wiki/Ajedrez_cu%C3%A1ntico (fecha de consulta: 13 de junio de 2023)
- Colaboradores de Wikipedia. (2023). *Deep Blue (computadora)*. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Deep_Blue_\(computadora\)](https://es.wikipedia.org/wiki/Deep_Blue_(computadora))
(fecha de consulta: 29 de junio de 2023)
- Colaboradores de Wikipedia. (2023). *Minimax*. Wikipedia, la enciclopedia libre.
<https://es.wikipedia.org/wiki/Minimax> (fecha de consulta: 7 de junio de 2023)

Colaboradores de Wikipedia. (2023). *Relación de indeterminación de Heisenberg*.

Wikipedia, la enciclopedia libre.

https://es.wikipedia.org/wiki/Relaci%C3%B3n_de_indeterminaci%C3%B3n_de_Heisenberg (fecha de consulta: 1 de junio de 2023)

IQIM Caltech. (2016, 27 enero). *Paul Rudd explores the Quantum Realm with Stephen Hawking* [Video]. YouTube. https://www.youtube.com/watch?v=Hi0BzqV_b44 (fecha de consulta: 5 de mayo de 2023)

python-chess. (lanzamiento 2020, 26 octubre). *python-chess*. PyPI.

<https://pypi.org/project/python-chess/> (fecha de consulta: 6 de junio de 2023)

QC Ware. (2020, 23 diciembre). Quantum Chess Tournament | Final | Q2B20 [Video]. YouTube. <https://www.youtube.com/watch?v=6wTtWLnEnwQ> (fecha de consulta: 29 de junio de 2023)

Quantum Realm Games. (s. f.). *Chess with a quantum twist*. Quantum chess.

Recuperado 30 de junio de 2023, de <https://quantumchess.net/> (fecha de consulta: 25 de junio de 2023)

QuantumFracture. (2021, 18 marzo). *Reto a una Gran Maestra al Ajedrez Cuántico*

[Video]. YouTube. <https://www.youtube.com/watch?v=b5eVgV6IPAw> (fecha de consulta: 12 de marzo de 2023)

stockfish. (lanzamiento 2022, 5 julio). *Stockfish*. PyPI. <https://pypi.org/project/stockfish/> (fecha de consulta: 6 de junio de 2023)

Wikipedia contributors. (2023). *Computer chess*. Wikipedia.

https://en.wikipedia.org/wiki/Computer_chess (fecha de consulta: 6 de junio de 2023)

Wikipedia contributors. (2023). *Stockfish (chess)*. Wikipedia.

[https://en.wikipedia.org/wiki/Stockfish_\(chess\)](https://en.wikipedia.org/wiki/Stockfish_(chess)) (fecha de consulta: 20 de abril de 2023)

Wikipedia contributors. (2023). *Werner Heisenberg*. Wikipedia.

https://en.wikipedia.org/wiki/Werner_Heisenberg (fecha de consulta: 1 de junio de 2023)

Wismath, A. (2012). Quantum Chess. Online Quantum Chess, de

https://research.cs.queensu.ca/home/aki/QUANTUM_CHESS/QuantumChessOnlineTheWeb.pdf
