PROYECTO 1

LAURA TATIANA COICUE - 2276652-3743

LAURA SOFÍA PEÑALOZA - 2259485-3743

SANTIAGO REYES RODRIGUEZ - 2259738-3743

JESSICA FERNANDA VILLA - 2266301-3743

PROFESOR

JOSHUA DAVID TRIANA

UNIVERSIDAD DEL VALLE SEDE TULUÁ

FACULTAD DE INGENIERÍA

PROGRAMA ACADÉMICO DE INGENIERÍA DE SISTEMAS

INTELIGENCIA ARTIFICIAL

TULUÁ – VALLE DEL CAUCA

2024

Para abordar el problema del ajedrez espejo, se decidió utilizar Python como lenguaje de programación tanto para la parte lógica como para la visual. Esto se debe a que emplear otro lenguaje para la interfaz podría generar complicaciones en la integración o el consumo de la misma. Además, considerando el tiempo limitado disponible, esta decisión favoreció la eficiencia del desarrollo. La estructura general del proyecto incluye dos carpetas principales que almacenan los recursos multimedia: una para los sonidos y otra para las imágenes.

En cuanto a la lógica del juego, el proyecto está organizado en varias clases que se encargan de todo el funcionamiento.

La clase juego controla las reglas y el flujo general del juego. Es responsable de inicializar y gestionar una partida de ajedrez variante Alice en Pygame. Al instanciarse, configura la ventana del juego, ajusta las dimensiones según el tamaño de las casillas y define las áreas del tablero, encabezado y zona de piezas capturadas. Además, carga configuraciones externas a través de un menú, lo que incluye el tamaño del tablero y la canción de fondo, que se reproduce en bucle utilizando el módulo de sonido de Pygame.

El juego carga las imágenes correspondientes a cada pieza (peón, caballo, alfil, torre, dama y rey) para ambos colores, escalándolas al tamaño adecuado. Maneja posibles errores al cargar dichas imágenes, mostrando un mensaje en consola si ocurre algún fallo. También inicializa sonidos, como el efecto de mover una ficha, y establece el turno inicial en blancos. Si el jugador elige jugar con negras, la inteligencia artificial realiza el primer movimiento usando el algoritmo MinMax.

Para la visualización, la clase dibujar_tablero renderiza dos tableros de ajedrez (base de la variante Alice), alternando colores claros y oscuros en las casillas. Además, resalta las posiciones de origen y destino del último movimiento, utilizando rectángulos de colores específicos. Se dibujan índices de letras y números alrededor de ambos tableros, un encabezado con el título "Alice Chess" y el nombre del equipo creador, "By LLJS Team". La interfaz también muestra las piezas capturadas, divididas por color, y la indicación de turno ("Mueven blancas" o "Mueven negras"). Un botón de menú se posiciona en la esquina superior derecha, permitiendo accesos adicionales. Finalmente, la clase Juego administra listas para las piezas capturadas, mantiene un registro del último movimiento y verifica si el jugador juega contra la IA. En ese caso, se calcula el mejor movimiento mediante el algoritmo implementado, actualizando el tablero y reproduciendo el sonido correspondiente al movimiento realizado.

Luego está la clase menú que gestiona la interfaz inicial y las opciones del usuario. El código implementa un sistema de menús interactivos para el ajedrez basado. Utiliza la biblioteca Pygame para el desarrollo de la interfaz gráfica y el manejo de eventos. El menú principal presenta opciones como "Jugar" y "Salir", con efectos visuales y sonoros al interactuar con los botones. Si el jugador selecciona "Jugar", se inicia un submenú donde se elige entre piezas blancas o negras. Esta selección se realiza mediante rectángulos interactivos que muestran imágenes escaladas de las piezas y cambian de color al pasar el cursor por encima (efecto *hover*). El menú también incluye efectos de sonido al interactuar con los botones, utilizando archivos de audio predefinidos. Además, se contempla un menú de configuración que permite ajustar parámetros como la resolución, seleccionar diferentes pistas de música y ajustar el volumen mediante una barra deslizante interactiva. El manejo de eventos es clave en todo el sistema, ya que se monitorizan acciones como clics del ratón, movimiento del cursor y cierres de ventana. Al seleccionar una opción, el juego ejecuta la lógica correspondiente, como iniciar la partida con las piezas elegidas. La interfaz combina gráficos, sonidos y control del flujo del programa, creando una experiencia de usuario intuitiva y dinámica.

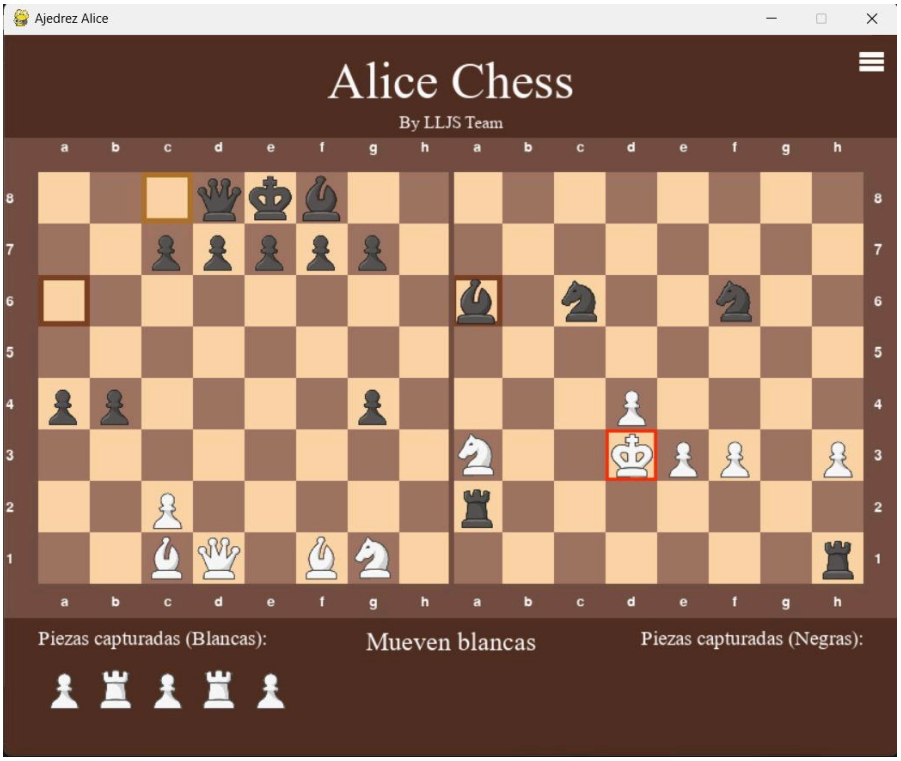
También está la clase min_Max que implementa el algoritmo de inteligencia artificial. Se utilizó el algoritmo Minimax con poda alfa-beta, la clase MinMax representa la IA y contiene métodos para evaluar el tablero, generar movimientos y decidir la mejor jugada, la función heurística evaluar_tablero asigna puntuaciones a las piezas en función de su valor (por ejemplo, el peón vale 100 y la dama 900) y considera bonificaciones posicionales, como peones avanzados o piezas menores ubicadas en el centro del tablero. El método principal, minimax, aplica el algoritmo recursivo Minimax para explorar las posibles jugadas hasta cierta profundidad (definida en la inicialización). Utiliza poda alfa-beta para reducir el número de nodos evaluados, mejorando así la eficiencia. La IA maximiza su puntuación mientras minimiza las del oponente, alternando entre "maximizador" y "minimizador" en cada nivel de la búsqueda. El método obtener_mejor_movimiento selecciona la jugada óptima usando minimax. Si es el primer movimiento, aplica una estrategia inicial, priorizando mover peones centrales. Para movimientos normales, explora todas las jugadas posibles y elige aquella con la mejor evaluación. La función obtener_todos_movimientos recopila todos los movimientos válidos para las piezas del tablero, iterando por cada casilla en ambos tableros y llamando al método movimientos_pieza para obtener las posiciones a las que puede moverse cada pieza.

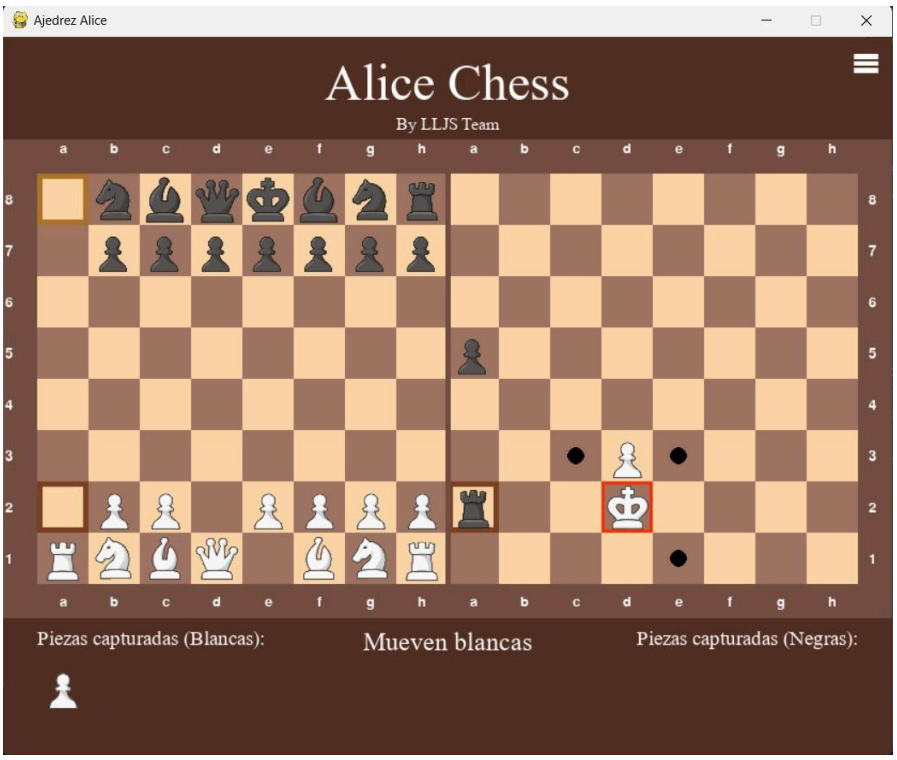
Además, se tiene la clase tablero que maneja la representación y lógica del tablero de ajedrez. El código proporcionado simula un tablero de ajedrez y maneja las reglas de los movimientos de las piezas, incluyendo condiciones especiales como el enroque y las capturas. Utiliza una combinación de clases y enumeraciones para representar las piezas del ajedrez, sus colores y las posibles direcciones de movimiento en el tablero. La clase Tablero es fundamental ya que se encarga de gestionar toda la lógica de los movimientos, la inicialización del tablero, y la verificación de situaciones especiales como el jaque o el jaque mate.

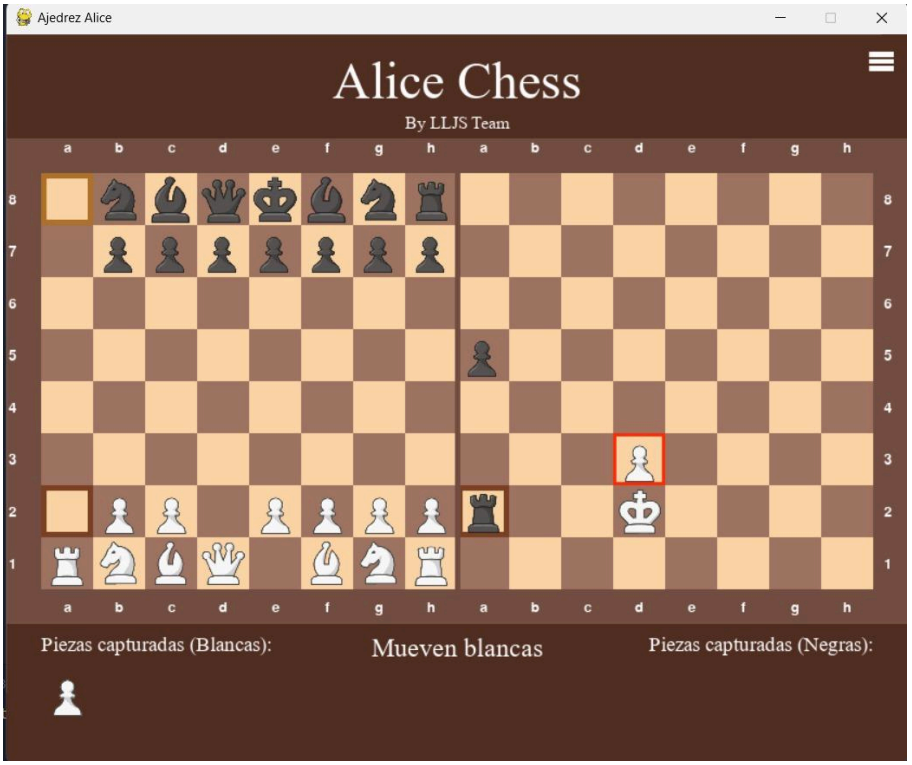
El método realizar_movimiento() es clave para ejecutar un movimiento en el tablero, verificando las reglas básicas de movimiento para cada pieza, así como las reglas específicas para el enroque y las capturas de piezas. Este método actualiza las posiciones de las piezas y mantiene un historial de movimientos para permitir retrocesos si es necesario. Además, maneja el cambio de turnos entre jugadores. inicializar_tablero() se encarga de colocar las piezas en su posición inicial en un tablero de ajedrez, asegurando que las piezas de cada jugador estén en las posiciones correctas correspondientes al inicio de un juego. copiar_tablero() crea una copia exacta del tablero actual, lo que permite a los jugadores o a las funciones de la aplicación comprobar diferentes estados del juego sin alterarlo. obtener_pieza() permite verificar la pieza en una posición específica del tablero, lo cual es útil para validar posibles movimientos de las piezas en función de las reglas del ajedrez. movimientos_pieza() calcula todos los movimientos posibles para una pieza dada en una posición específica del tablero, tomando en cuenta las reglas de movimiento de esa pieza y asegurándose de no permitir movimientos inválidos o fuera de las reglas del juego. Finalmente, esta_en_jaque() verifica si una posición específica está bajo ataque, es decir, si el rey de un jugador está en posición de ser capturado en su próximo turno. Esta función utiliza el concepto de "ataque directo" para determinar si la pieza que ataca puede capturar al rey directamente.

Por último, la clase main actúa como el punto de entrada del programa.

Este enfoque estructurado del código permite una implementación robusta y precisa de las reglas de ajedrez en una aplicación, permitiendo a los usuarios jugar de manera legal y estratégica en un entorno digital.


Ajedrez Alice
Alice Chess
By LLJS Team
Piezas capturadas (Blancas):
Mueven blancas
Piezas capturadas (Negras):


Ajedrez Alice
Alice Chess
By LLJS Team
Piezas capturadas (Blancas):
Mueven blancas
Piezas capturadas (Negras):



Por medio de las pruebas, lo que se buscó fue la visualización correcta de los Jaque y Jaque Mate en los casos en que fuera necesario, esto, para asegurar que el juego se llevará a cabo de la mejor forma.

De igual forma, en las pruebas se buscó siempre poder atacar al rey del equipo contrario, caso que resultó casi imposible, debido a que cada intento por llegar, era parado por alguna de las fichas del oponente, resaltando que la forma en que está jugando la IA es la mejor.