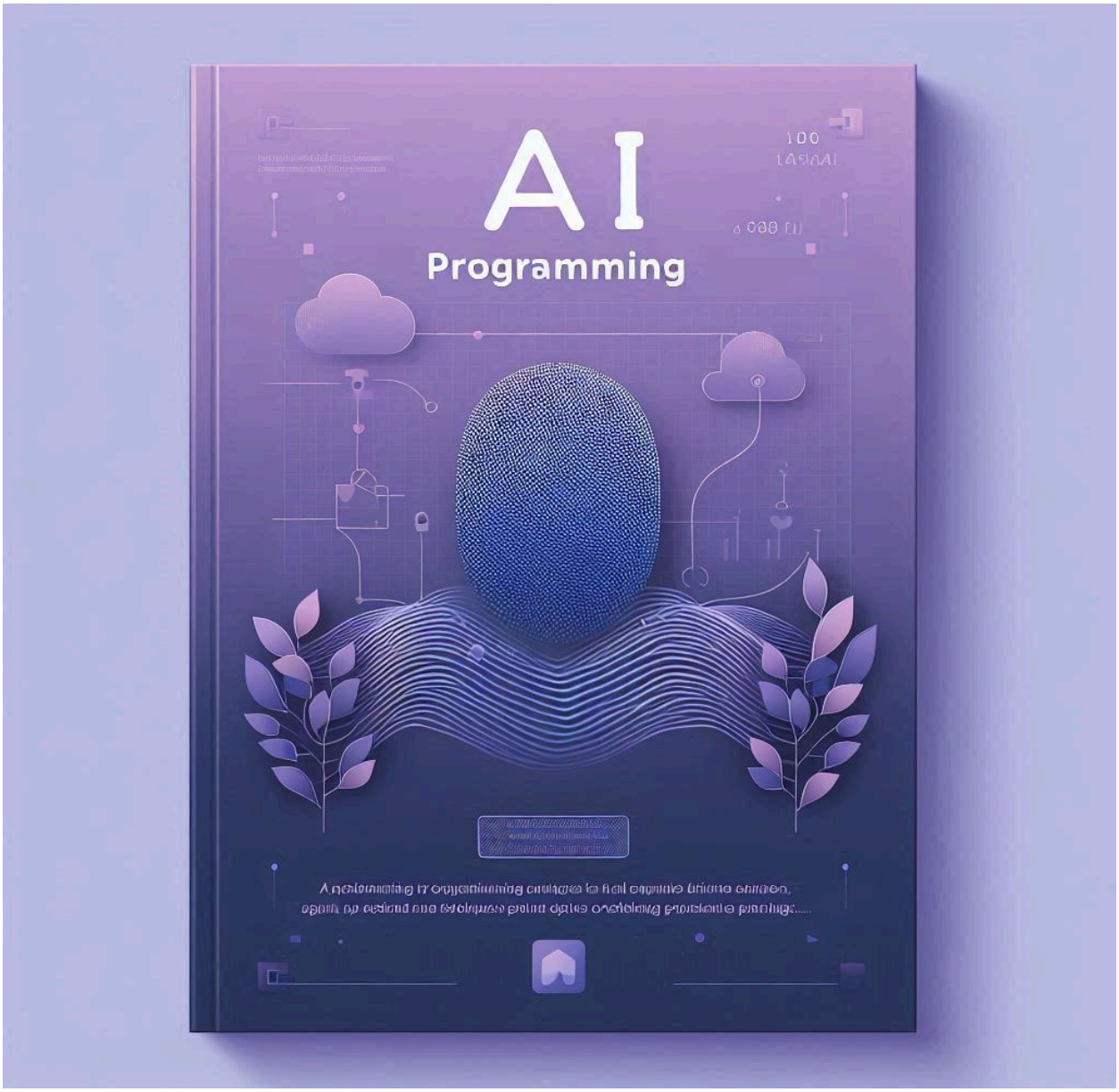
Programación en Java



Autora: Laura Fornós Ramírez

Curs: Especialització IA i Big Data

MP3. Programació d' intel·ligència artificial

Profesor: Francis Real Vázquez

Laura Fornós Ramírez
Curs Especialització IA i Big Data

Algoritmo minimax - Ratón y gatos

M3

Objetivo

En esta práctica, se desarrollará un juego basado en el algoritmo Minimax, aplicando la búsqueda de estados en juegos. El objetivo es programar el juego del "Ratón y los Gatos" sobre un tablero de ajedrez, utilizando una interfaz gráfica básica, gestionando turnos, movimientos, detección de ganadores y la implementación del algoritmo Minimax con heurísticas adecuadas para que la IA controle a los gatos.

Descripción del Juego

- Se juega sobre un tablero de ajedrez de 8x8 casillas.
- Solo se usan las casillas blancas para el movimiento de las piezas.
- Hay cuatro gatos que inician en la fila 1 (inferior del tablero).
- Hay un ratón que inicia en la fila 8 (superior del tablero).
- El ratón comienza el juego.

Reglas de Movimiento

- **Ratón:**
    - Se mueve en diagonal una casilla a la vez.
    - Puede moverse hacia adelante (hacia la fila inferior) o hacia atrás (hacia la fila superior).
    - No puede moverse a una casilla ocupada por un gato.
- **Gatos:**
    - Cada turno, el jugador de los gatos selecciona uno de los cuatro gatos.
    - El gato seleccionado puede moverse en diagonal una casilla hacia adelante (fila superior).
    - No pueden moverse hacia atrás ni ocupar una casilla ocupada por otro gato o ratón.

Laura Fornós Ramírez
Curs Especialització IA i Big Data

Condiciones de Victoria

- **Gana el ratón si:**
    - Llega a la fila 1 (inferior del tablero).
    - Todos los gatos están en filas superiores a la suya.
- **Ganan los gatos si:**
    - El ratón no tiene ninguna casilla disponible para moverse en su turno.

Requisitos de Implementación

- **Interfaz Gráfica Mínima:**
    - Representación visual del tablero y las piezas.
    - Posibilidad de seleccionar piezas y realizar movimientos.
- **Gestión de Turnos y Movimientos:**
    - Alternancia entre turnos de ratón y gatos y Validación de movimientos legales.
- **Detección de Condiciones de Victoria:**
    - Implementación de la lógica que determine cuándo gana el ratón o los gatos.
- **Implementación del Algoritmo Minimax:**
    - Los gatos deben ser controlados por una IA basada en Minimax.
    - Se debe definir una heurística adecuada para que la IA tome decisiones inteligentes.
- **Modo de Juego Humano vs. IA:** Un jugador humano debe poder jugar como el ratón contra la IA de los gatos.

Entrega y Evaluación

- Código bien estructurado y documentado.
- Funcionamiento correcto del juego y del algoritmo Minimax.
- Implementación de una interfaz gráfica funcional y calidad de la heurística utilizada en la IA.

Laura Fornós Ramírez
Curs Especialització IA i Big Data

Decisiones tomadas

Primero, he planteado las **clases principales** que quiero en mi código, las cuales fueron r**atón, gato y tablero**, lo mínimo para poder entender que necesito y como quiero que se vea.

Después, procedí a plantear cómo sería **el tablero** y las imágenes que querría para mi ratón y mis cuatro gatos, así que la primera clase que hice fue la de Tablero, y luego gato y ratón. El programa no funcionaba pero al menos con **jframe** podía ver cómo había quedado mi juego.

Luego, empecé a plantear el movimiento para poder ver que primero necesitaba la **posición inicial** de cada uno de los elementos, el **ratón** en la **fila 1**, **columna 5** (como habías decidido antes), y los **gatos** en la **fila 8**.

A continuación, me puse con las clases **jugadores** (obtenemos la posición de las piezas) y **turnos** (la lógica del juego), donde en esta última, he pasado la mayoría del tiempo, ajustando primero los parámetros iniciales, como **inicializa**r el tablero y los **turnos de ratón y gatos,** después planteando el **movimiento de las piezas**, las condiciones de **victoria** y por último la parte más complicada que ha sido la implementación del **algoritmo minimax** y la **heurística**.

En el **algoritmo minimax** he usado la distancia **Manhattan,** porque es la más fácil de entender y de plasmar en código, para poder calcular en base a la posición en coordenadas de los gatos y el ratón, el movimiento más óptimo. También he usado poda Alpha-Beta, donde Alpha recoge el mejor valor para MAX y Beta recoge el mejor valor para MIN. Esta técnica descarta ramas que no llevan a un buen resultado final y mejora la velocidad de procesamiento.

Después, probé diferentes **profundidade**s para encontrar la óptima. Con **4, 6 y 8**, los resultados eran rápidos, pero la IA tomaba decisiones pobres. En cambio, con **10 o 12,** los gatos jugaban mejor, pero el tiempo de ejecución era demasiado alto, afectando la fluidez del juego. Finalmente, decidí que una profundidad de **9** era el mejor **equilibrio** entre rendimiento y calidad de juegos.

La **otra heurística** ha consistido en evaluar el tablero de manera que favorece las posiciones de los gatos que **acorralan** al ratón y sobretodo la **cooperación** de los cuatro gatos. Para otorgar una puntuación, se **puntúa** si el ratón tiene pocos movimientos válidos, incentivando a los gatos a limitar sus opciones. Si el ratón está cerca de ganar o llegar a la fila 7, se **penaliza** fuertemente. Además, se **otorga**n puntos a los gatos según su proximidad al ratón: capturas inmediatas o bloqueos cercanos suman más puntos, mientras que posiciones más alejadas **restan**.

Por último, simplemente con la clase **Jugar**, creamos un nuevo tablero y damos comienzo a nuestra partida.

Laura Fornós Ramírez
Curs Especialització IA i Big Data

Resumen de Clases

- **Tablero:** Clase que gestiona la apariencia del tablero, en cuanto a casillas en blanco y negro, y los iconos de el raton y los gatos, que han sido imagenes cargadas en nuestro proyecto, aparte se actualiza el tablero despues de los movimientos para la buena representacion de la parte grafica del tablero.
- **Gato:** Se procesan los movimientos válidos de los Gato en diagonal, y se guardan en un array los posibles movimientos, se calcula el nuevo movimiento que hará y si es valido hacerlo.
- **Ratón:** Se procesa el movimiento del ratón y si podemos movernos a las casillas adyacentes y si el movimiento que queremos hacer se puede hacer, actualiza la posición del ratón.
- **Movimiento:** Se procesan las posiciones de inicio y final, guardando las filas y columnas de origen y destino.
- **Jugadores:** Métodos para obtener y establecer la posición del jugador. También un método abstracto puedeMoverse que debe ser implementado en las clases hijas para definir la lógica de movimiento de cada tipo de jugador (probablemente un gato o un ratón).
- **Jugar:** Inicializamos un nuevo tablero, con el que empezamos a jugar
- **Turnos:** La magia del programa, donde se procesa el juego, a continuación se explica en profundidad ya que todo el planteamiento del juego está aquí.

Laura Fornós Ramírez
Curs Especialització IA i Big Data

Estructura del Código de la Clase Turnos

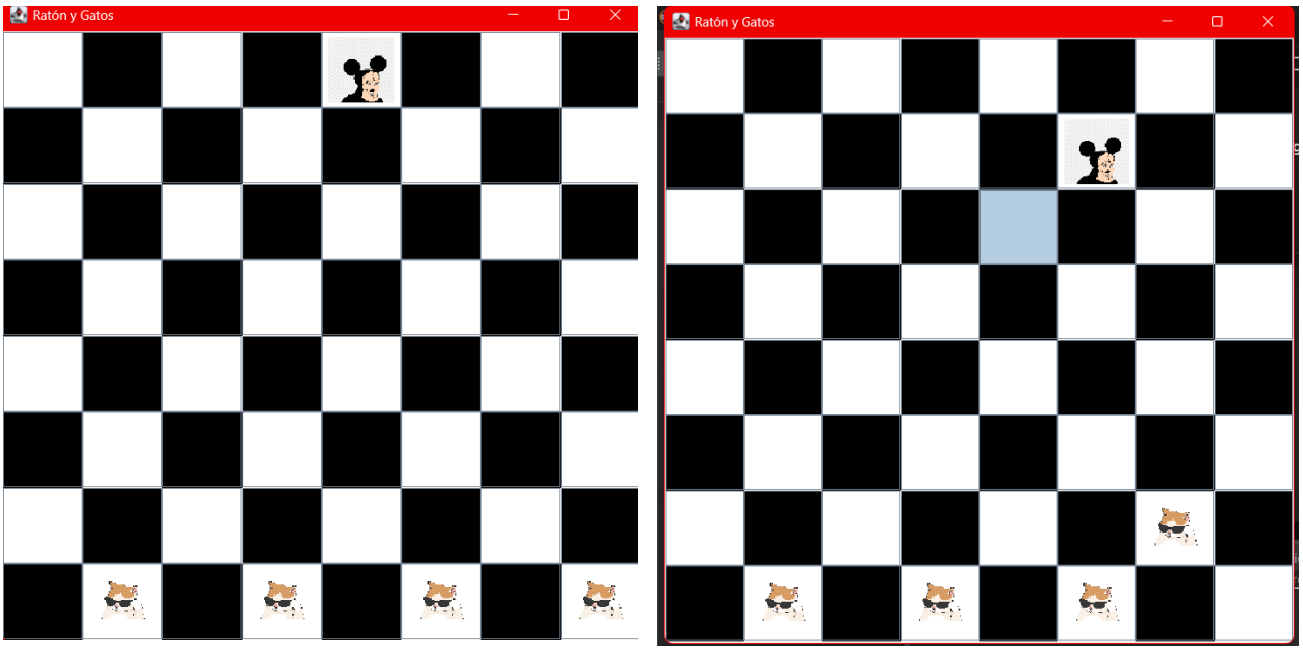
La clase Turnos gestiona el flujo del juego y es donde podemos encontrar cómo se procesan las clases anteriores y la IA de los gatos.

1. **Inicialización del Tablero**:
    a. En el constructor Turnos, inicializo el tablero colocando los gatos en la fila 7 y el ratón en la fila 0, columna 4. Establezco que es el turno del ratón.
2. **Movimiento y Validación**:
    a. Utilizo el método esMovimientoValido para verificar si un movimiento es válido, asegurándome de que el destino esté dentro del tablero, en una casilla vacía y sea una casilla válida.
    b. El método moverPieza mueve una pieza de una casilla a otra y actualiza la posición del ratón si es necesario.
3. **Manejo de Turnos**:
    a. El método manejarClick gestiona el clic del usuario para mover el ratón. Si el movimiento es válido y el ratón se mueve, verifico si el ratón ha ganado; si no, cambio el turno a la IA (gatos).
4. **Turno de la IA**:
    a. El método turnoIA utiliza el algoritmo minimax con poda alpha-beta (minimaxAlphaBeta) para determinar el mejor movimiento de los gatos, lo ejecuto y verifico condiciones de victoria.
5. **Evaluación del Tablero**:
    a. El método evaluarTablero evalúa la posición del tablero usando la distancia **Manhattan** y otros factores, asignando una **puntuación heurística** que guía la toma de decisiones.
6. **Minimax con Poda Alpha-Beta**:
    a. El método minimaxAlphaBeta recorre el árbol de decisiones hasta una profundidad dada, evaluando posiciones y utilizando poda alpha-beta para optimizar la búsqueda. Genera y ordena los movimientos posibles de gatos y ratón, eligiendo el mejor movimiento basado en la evaluación del tablero.
7. **Condiciones de Victoria**:
    a. Los métodos esVictoriaGatos y esVictoriaRaton determinan si los gatos o el ratón han ganado, respectivamente, y muestran el mensaje correspondiente en un cuadro de diálogo.
8. **Obtener Movimientos**:
    a. Con los métodos obtenerMovimientosGatos y obtenerMovimientosRaton, ordeno los movimientos de manera estratégica en Manhattan, optimizando así las decisiones del algoritmo minimax.

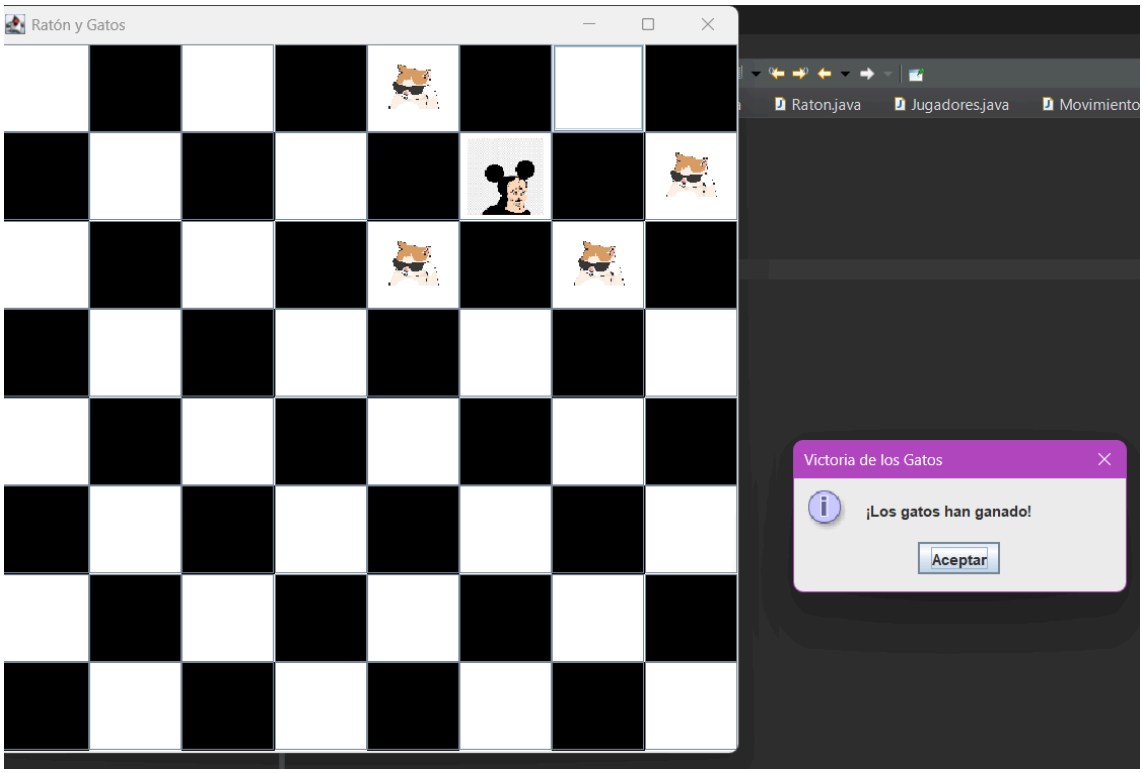
Laura Fornós Ramírez
Curs Especialització IA i Big Data

Capturas

Tablero inicial y selección de casilla clicando donde queremos que se mueva el ratón



Ganan Gatos



Laura Fornós Ramírez
Curs Especialització IA i Big Data

Gana Ratón (adelantó a los cuatro gatos)