جامعة مصر للمعلوماتية
EGYPT UNIVERSITY OF INFORMATICS

Egypt University of Informatics
Computer and Information Systems
Artificial Intelligence Course

Introduction to Artificial Intelligence Project

Submitted by:

Badr Mohamed (22-101178)

Omar khaled (22-101132)

Youssef yasser (22-101089)

Youssef elmnshawi (22-101241)

Supervised by:

Dr. Mohamed Taher El Refaei

Eng. Tukka Mohammed

5/1/2025

Introduction:

Artificial Intelligence (AI) is often used to solve complex problems, including games like chess. Chess is a great way to test AI because it requires strategic thinking.
In this project, we created a Python-based chess game with an AI agent using the Minimax algorithm and Alpha-Beta pruning. The AI evaluates the board, predicts outcomes, and chooses the best moves to play intelligently and efficiently.
This project showcases how AI can be used to build smart game players.

Features:

1. Chessboard Display

- Draws an 8x8 chessboard with alternating white and green squares.

---

2. Piece Rendering

- Loads and displays images for all chess pieces (pawns, rooks, knights).

---

3. Drag-and-Drop Moves

- Players can click and drag pieces to move them on the board.

---

4. Turn-Based Play

- Alternates turns between the player (white) and the AI (black).

---

5. Chess Rules Enforcement

- Validates moves based on standard chess rules (pawns move forward, knights move in L-shapes).

---

6. Capture Logic

- Captured pieces are removed from the board and displayed in a side panel.

---

7. Simple AI Opponent

- AI makes moves based on piece values, king safety, and board control.

---

8. Check Detection

- Alerts players when a king is in check.

---

9. Checkmate Detection

- Detects when a player has no legal moves to escape check, ending the game.

---

10. Game Over Screen

- Displays a message when checkmate occurs and offers a restart option.

---

11. Restart Functionality

- Resets the board to the starting position and clears captured pieces.

Technical Implementation:

1. Pygame Setup

- Uses pygame for graphics and input.
- Creates a window (1000x600 pixels) and defines colors for the board and side panel.

---

2. Chessboard

- Represented as an 8x8 list (matrix).
- Each cell stores a piece ('wP' for white pawn) or . for empty.

---

3. Piece Images

- Loads images for pieces (wP.png, bK.png) from an images folder.
- Stores images in a dictionary (PIECE_IMAGES) for easy access.

---

4. Drag-and-Drop

- Tracks mouse clicks to pick up pieces and releases to drop them.
- Temporarily removes the piece from the board while dragging.

---

5. Move Validation

- A rules() function checks if a move is valid based on the piece type and chess rules.
- Prevents illegal moves (capturing your own pieces).

---

6. AI Opponent

- Evaluates moves using piece values, king safety, and board control.
- Simulates moves and chooses the best one.

---

7. Check and Checkmate

- is_king_in_check(): Checks if a king is under attack.
- is_checkmate(): Checks if a player has no legal moves to escape check.

---

8. Game Over

- Displays a checkmate message and offers a restart option.
- restart() function resets the board and clears captured pieces.

---

9. Captured Pieces

- Stores captured pieces in lists (captured_white and captured_black).
- Displays them in the side panel.

---

10. Visuals

- Uses a custom font for text (falls back to default if unavailable).
- Adds a semi-transparent overlay for the game-over screen.

---

11. Modular Code

- Functions handle specific tasks:
    - draw_board(): Draws the chessboard.
    - draw_pieces(): Draws the pieces.

- evaluate_position(): Evaluates the board for the AI.

---

12. Main Loop

- Handles events (mouse clicks, quitting).
- Updates the screen and alternates turns between player and AI.

Technical Challenges:

1. Move Validation

- **Challenge**: Ensuring all pieces move according to chess rules (e.g., pawns move forward, knights jump in L-shapes).
- **Solution**: Write a rules() function that checks valid moves for each piece type.

---

2. Drag-and-Drop Mechanics

- **Challenge**: Smoothly picking up, dragging, and dropping pieces without glitches.
- **Solution**: Track mouse events (MOUSEBUTTONDOWN and MOUSEBUTTONUP) and temporarily remove the piece from the board while dragging.

---

3. AI Implementation

- **Challenge**: Creating an AI that makes reasonable moves.
- **Solution**: Use a simple evaluation function (piece values, king safety) to choose the best move.

---

4. Check and Checkmate Detection

- **Challenge**: Detecting when a king is in check or checkmate.
- **Solution**: Write functions (is_king_in_check() and is_checkmate()) to simulate moves and check for threats.

---

5. Game State Management

- **Challenge**: Keeping track of the board, captured pieces, and whose turn it is.
- **Solution**: Use global variables (board, captured_white, current_turn) and update them after each move.

---

6. Visual Rendering

- **Challenge**: Drawing the board, pieces, and side panel correctly.
- **Solution**: Use Pygame's blit() function to render images and shapes at the right positions.

---

7. Performance Optimization

- **Challenge**: Ensuring the game runs smoothly, especially during AI calculations.
- **Solution**: Limit AI search depth and optimize move evaluation logic.

---

8. Edge Cases

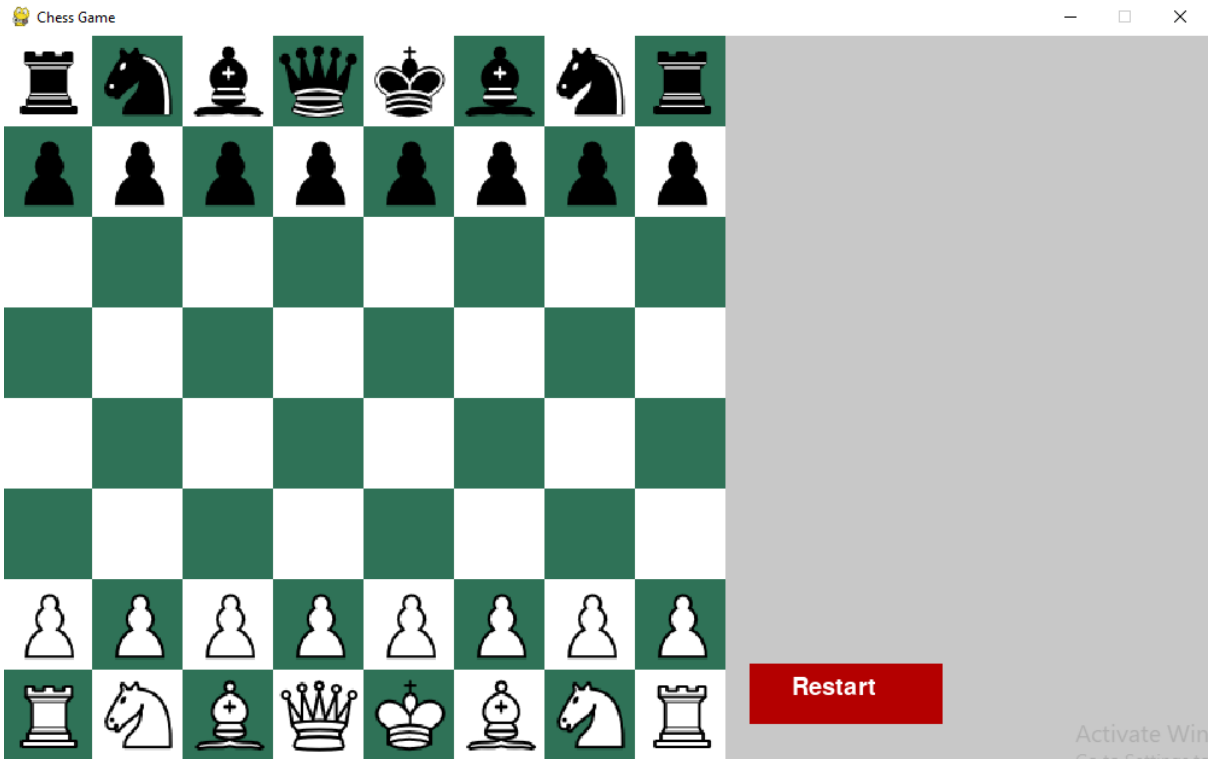
- **Challenge**: Handling edge cases like pawn promotion, castling, and en passant.
- **Solution**: Implement additional rules and logic for these special moves (currently missing in the code).

---

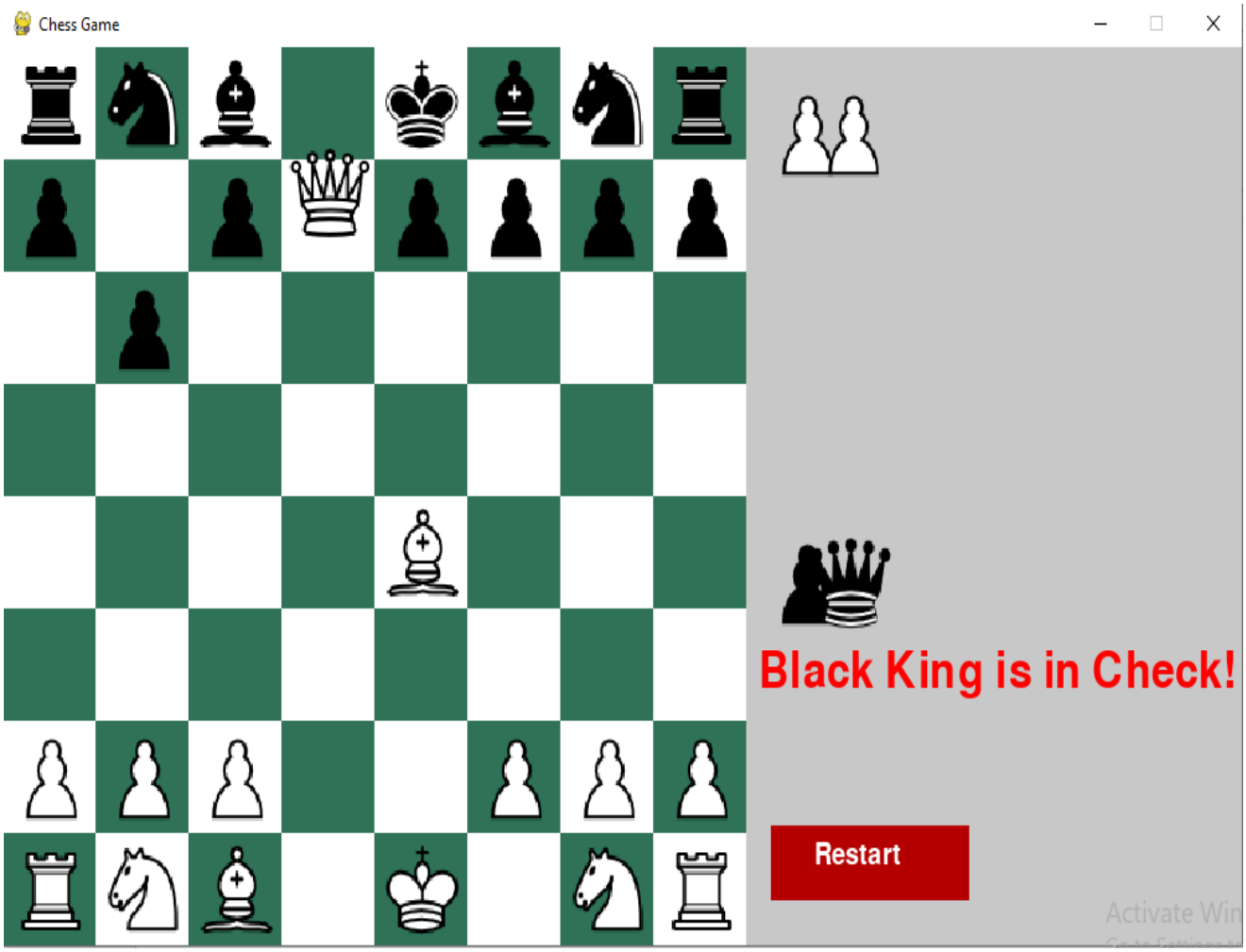
9. User Interaction

- **Challenge**: Making the game intuitive and responsive for players.
- **Solution**: Add visual feedback (highlighting valid moves) and ensure smooth drag-and-drop functionality.

Visuals:



Here we start the game..



If the Black king in Check ,the program alert me

When make Checkmate ,we have option to restart the game

Conclusion:

In summary, this chess project successfully combines the classic game of chess with Python programming. Using Pygame, the game offers an interactive and visually appealing experience. The AI opponent, though simple, provides a challenging experience by prioritizing strategic moves and king safety.

The project highlights key programming concepts like event handling, graphical rendering, and decision-making logic. It also serves as a great starting point for learning game development and AI implementation. Future improvements could include more advanced AI, multiplayer support, or additional chess variants. Overall, this project is a fun and educational way to explore programming and game design.