

**Project Title:**

**Multi-Player Strategy Chess: An AI-Driven 3-Player Chess Variant**

**Submitted By:**

Shiza Aijaz (22k-4683)

Asna Arshad (21k-2000)

**Course:**

AI-Lab

**Section:**

BCY-6B

**Instructor:**

Miss Mehak Mazhar

**Submission Date:**

09-May-2025

**1. Project Overview**

**● Project Topic:**  
Design and implement an unconventional board game—Multi-Player Strategy Chess—that extends traditional two-player chess into a three-player format. This variant uses a modified board layout, additional piece types, and complex rules while integrating advanced AI techniques.

**● Objective:**  
The main goal is to develop a strategic AI for a multi-player chess variant using a multi-agent extension of the Minimax algorithm (maxⁿ). The project aims to:

* Handle the increased complexity and branching factors of a three-player game.
* Refine evaluation heuristics that incorporate piece value, mobility, and board control.
* Create an immersive gaming experience with a graphical user interface (GUI) featuring smooth animations and sound effects.

**2. Game Description**

**● Original Game Background:**  
Traditional chess is a two-player strategy game where each player commands a set of pieces (King, Queen, Rook, Bishop, Knight, Pawn) on an 8×8 board. The objective is to checkmate the opponent's king. In our variant, the conventional game is extended to three players with asymmetric starting positions.

**● Innovations Introduced:**

* **Multi-Player Setup:**
  + Player 1 (Human) is assigned the bottom row with a standard back row formation.
  + Player 2 (AI) uses the top row with a standard back row.
  + Player 3 (AI) is arranged vertically along the left column with a modified set of pieces.
* **Additional Piece Types & Complex Rules:**
  + Implementation of five piece types: King, Queen, Rook, Bishop, and Knight.
  + Each piece moves following standard chess rules, with adaptations for board layout and balance.
  + Modified game rules include unique board configurations and elimination conditions (a player is removed when their king is captured).
* **Dynamic GUI Features:**
  + A graphical interface using Pygame displays the board, highlights valid moves, and shows smooth animations for piece movements.
  + Sound effects play on move execution to enhance the user experience.
* **AI Integration:**
  + Use of the maxⁿ algorithm (a multi-player variant of Minimax) with refined evaluation heuristics for decision making.
  + Evaluation incorporates base piece values, mobility (number of legal moves), and central control bonuses.

**3. AI Approach and Methodology**

**● AI Techniques to be Used:**

* **Maxⁿ Algorithm:**
  + A multi-player extension of the traditional minimax algorithm to evaluate moves for all players.
* **Optional Enhancements:**
  + Alpha-Beta Pruning for potential efficiency improvements.
  + Future exploration of reinforcement learning for self-play and strategy optimization.

**● Heuristic Design:**

* **Evaluation Function:**
  + **Piece Value:** Standard values (King, Queen, Rook, Bishop, Knight).
  + **Mobility Bonus:** Extra score based on the number of legal moves available to each piece.
  + **Central Control Bonus:** Added value for pieces occupying the central squares of the board.
* These combined factors help the AI estimate board states more effectively in a multi-agent setting.

**● Complexity Analysis:**

* The branching factor increases significantly with three players, making deep searches computationally intensive.
* The maxⁿ algorithm has exponential time complexity with search depth; iterative deepening and heuristic pruning might be required to ensure timely decisions.
* Balancing game rules and evaluation criteria presents additional challenges in AI decision-making.

**4. Game Rules and Mechanics**

**● Modified Rules:**

* **Board Layout:**
  + Player 1 and Player 2 have traditional back-row formations (bottom and top rows, respectively).
  + Player 3’s pieces are arranged vertically on the left column, with a reduced set to fit the board.
* **Piece Movement:**
  + Each piece follows standard chess movement rules (e.g., sliding for Rook/Bishop/Queen, L-shaped moves for Knight, one-step moves for King).
  + No castling or en passant moves in this initial implementation.
* **Capture and Elimination:**
  + A capture occurs when a piece moves into a square occupied by an opponent’s piece.
  + A player is eliminated when their King is captured.

**● Winning Conditions:**

* The game ends when only one king remains on the board.
* The last surviving player is declared the winner.

**● Turn Sequence:**

* The game proceeds in a round-robin fashion:
  + **Player 1:** Human-controlled via the GUI (selecting pieces and destinations with a mouse).
  + **Player 2 & 3:** AI-controlled, using the maxⁿ algorithm to determine moves.
* Each turn allows one move per player before passing to the next.

**5. Implementation Plan**

**● Programming Language:**

* Python

**● Libraries and Tools:**

* **Pygame:** For GUI development, rendering the board, and handling animations and sound effects.
* **Standard Python Libraries:** For game logic (copy, sys, etc.).
* **Optional Tools:**
  + NumPy for data handling.
  + TensorFlow or Scikit-learn for potential future integration of reinforcement learning.

**● Milestones and Timeline:**

* **Week 1-2:**
  + Finalize game design, board layout, and rule modifications.
* **Week 3-4:**
  + Develop the AI strategy using the maxⁿ algorithm.
  + Refine evaluation heuristics (piece value, mobility, central control).
* **Week 5-6:**
  + Code game mechanics, implement additional piece types, and incorporate advanced move rules.
* **Week 7:**
  + Integrate the GUI with animations and sound effects.
  + Test human–AI interactions and multi-player turn management.
* **Week 8:**
  + Final testing, debugging, and report preparation.

**6. References**

 **Pygame Documentation.**  
 Pygame Official Website. Available at: https://www.pygame.org/docs/

 **Chess Rules and Piece Movement.**  
 Wikipedia article on Chess. Available at: <https://en.wikipedia.org/wiki/Chess>

 **Minimax and Maxⁿ Algorithms.**  
 Wikipedia articles on Minimax and Multi-Player Extensions.

* Minimax: <https://en.wikipedia.org/wiki/Minimax>
* Maxⁿ: [https://en.wikipedia.org/wiki/Max^n\_algorithm](https://en.wikipedia.org/wiki/Max%5En_algorithm)

 **Evaluation Heuristics in Chess Programming.**  
 Chess Programming Wiki: Evaluation. Available at: https://www.chessprogramming.org/Evaluation

 **Reinforcement Learning: An Introduction.**  
 Sutton, R. S., & Barto, A. G. (2018).  
 Available at: http://incompleteideas.net/book/the-book-2nd.html