

**Final Project Report**

**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:**

11-May-2025

**Multi-Player Strategy Chess: Final Project Report**

**1. Abstract**

This project introduces Multi-Player Strategy Chess, an unconventional extension of classic two-player chess into a three-player format. The game features a uniquely modified board, asymmetric player positions, and strategic AI agents. Advanced decision-making is powered by the maxⁿ algorithm, enabling effective multi-agent gameplay. A dynamic graphical user interface (GUI) is built using Pygame, incorporating visual animations and sound effects to deliver an immersive experience.

**2. Introduction**

Chess is a timeless game of strategy and intellect. Traditionally played between two players, this project reimagines the game with a third player, introducing new complexities and tactical dimensions. The main objective is to develop an AI capable of participating in this environment using multi-agent decision-making logic and provide a smooth, engaging GUI for human interaction.

**3. Game Design and Rules**

**3.1 Modified Board and Setup**

* A custom square board layout accommodates three players.
* **Player 1 (Human):** Bottom row with standard back row.
* **Player 2 (AI):** Top row with standard back row.
* **Player 3 (AI):** Left column, vertically aligned pieces.

**3.2 Game Rules**

* **Piece Types:** King, Queen, Rook, Bishop, Knight (no pawns).
* Movements follow traditional chess logic adapted to the new layout.
* No castling or en passant in this variant.
* Capture leads to removal of the opponent’s piece.
* A player is eliminated when their king is captured.
* The game ends when only one king remains—that player wins.
* Turns proceed in a round-robin sequence: Player 1 → Player 2 → Player 3.

**4. AI Strategy**

**4.1 Algorithm**

* **Maxⁿ Algorithm:** A multi-player version of Minimax where each player aims to maximize their own utility.
* Handles increasing branching factors due to the third player.

**4.2 Evaluation Heuristics**

* **Piece Value:** Standard point values assigned.
* **Mobility:** Bonus points based on the number of legal moves.
* **Central Control:** Pieces occupying central board areas receive additional value.

**4.3 Optional Enhancements (Future Scope)**

* **Alpha-Beta Pruning** for efficiency.
* **Reinforcement Learning** (e.g., Q-learning/self-play) to improve long-term strategies.

**5. Implementation Details**

**5.1 Programming Language**

* **Python 3**

**5.2 Tools and Libraries**

* **Pygame:** GUI rendering, animations, and sound effects.
* **Standard Python Libraries:** such as copy, sys.
* **Optional:** NumPy for board representation.

**5.3 GUI Features**

* Board rendered with interactive squares.
* Mouse-click selection for human player moves.
* Valid moves highlighted on selection.
* Smooth animations for piece movement.
* Sound feedback on moves and captures.

**6. Testing and Results**

**6.1 Functional Testing**

* Game states tested for valid turn execution.
* AI tested against each other and human player.
* GUI behavior validated for different moves, captures, and elimination conditions.

**6.2 AI Performance**

* The AI was able to make competitive decisions and handle king elimination scenarios.
* Maxⁿ successfully predicted multi-player interactions with acceptable performance at depths of 2–3.
* Performance decreases with higher depths due to exponential complexity.

**7. Challenges and Solutions**

| **Challenge** | **Solution** |
| --- | --- |
| Adapting movement rules for vertical player | Implemented board transformation logic to reuse movement code |
| High branching factor for 3 players | Applied depth limits and optimized heuristics |
| GUI synchronization of turns and animations | Built a turn-based controller with time delay buffer |

**8. Future Work**

* Implement Alpha-Beta Pruning for deeper searches.
* Introduce pawn-like pieces and rules like castling.
* Add online multiplayer support.
* Integrate reinforcement learning to allow AI to evolve through self-play.

**9. Conclusion**

The *Multi-Player Strategy Chess* project successfully extended traditional chess into a challenging three-player format. Through the use of the maxⁿ algorithm and well-crafted heuristics, the AI provided competitive gameplay in a novel asymmetric environment. The project not only achieved its initial objectives but laid the groundwork for future exploration in multi-agent board game AI.

**10. References**

1. Pygame Documentation: <https://www.pygame.org/docs/>
2. Chess Rules: <https://en.wikipedia.org/wiki/Chess>
3. Maxⁿ Algorithm: <https://en.wikipedia.org/wiki/Max%5En_algorithm>
4. Minimax: <https://en.wikipedia.org/wiki/Minimax>
5. Chess Evaluation Techniques: <https://www.chessprogramming.org/Evaluation>
6. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. <http://incompleteideas.net/book/the-book-2nd.html>