Project Title: Connect 4 AI: An Adversarial Search Implementation

Group Members:
Bryan Maus 885677641
Jericho Salonga 885551705
Logan Clampitt 884973835

Problem Statement

Connect 4 is a two-player connection game where players drop colored discs into a 6-row, 7-column vertical grid. The objective is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. The problem we will solve is to design, implement, and evaluate an artificial intelligence agent that can play Connect 4 at a highly competitive level against a human player.

Approach

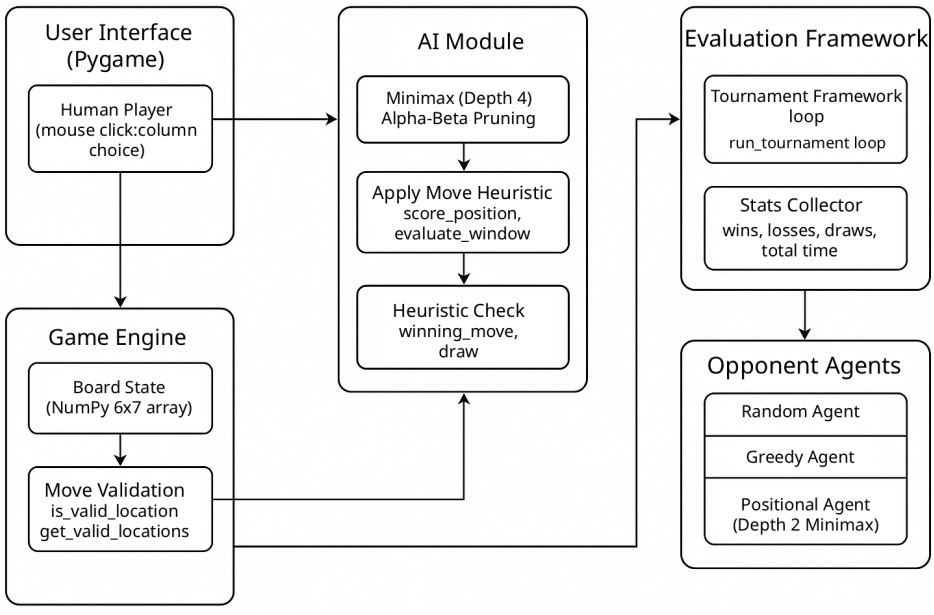
Our AI agent is based on **adversarial search** techniques covered in CPSC 481.

- **Minimax Algorithm:** The core decision-making logic uses the minimax algorithm, which evaluates future game states under the assumption that both players act optimally. The AI selects moves that maximize its minimum guaranteed outcome, making it effective for deterministic zero-sum games like Connect 4.
- **Alpha-Beta Pruning:** To improve efficiency and allow deeper lookahead, alpha-beta pruning is incorporated into the minimax search. This technique eliminates branches of the game tree that cannot influence the final decision, reducing computational cost without affecting correctness.
- **Heuristic Evaluation:** Since the full Connect 4 game tree is too large to explore exhaustively, a heuristic evaluation function is used when the depth limit is reached. Our custom heuristic contribution prioritizes a Central Control Strategy. We tuned specific weights (+3 for Center, +5 for 3-in-a-row) to create an aggressive agent that prefers controlling the board geometry over passive defense.
- **Adversarial Reasoning:** The AI explicitly models the opponent as an intelligent adversary whose goal is to minimize the AI's success. Rather than reacting only to the current board state, the AI reasons about how the opponent will respond to each possible move.
- **Game Tree Search:** The Connect 4 board is represented as a **game tree**, where nodes correspond to board states and edges represent legal moves. The AI explores this tree up to a fixed depth, enabling lookahead-based decision making instead of greedy play.

Software Description

The system is implemented in Python using NumPy for efficient board representation and Pygame for visualization. The architecture is modular, separating the game engine, AI logic, and evaluation framework.

- **System Components**
  - **Game Engine:** Manages board state, validates moves, drops pieces, and checks for win/loss/draw conditions.
  - **AI Module:** Implements minimax with alpha-beta pruning and heuristic evaluation function.
  - **Opponent Agents:** Includes baseline agents (Random, Greedy, and Positional) used for evaluation.
  - **Tournament Framework:** Automates large-scale testing, records win/loss statistics, and measures execution time.
  - **User Interface:** Displays the game board and allows human interaction via mouse input.
- **Data Flow:**
  1. The current board state is passed to the active agent.
  2. The agent selects a column based on its strategy.
  3. The game engine updates the board and checks for terminal conditions.
  4. The updated board is rendered in the GUI or passed to the next agent during evaluation.



Evaluation

We evaluate the AI using the following criteria:

- **Performance:** Win/loss/draw rates against baseline agents.
- **Efficiency:** Total runtime across tournament simulations.
- **Correctness:** Tactical accuracy in known winning or blocking scenarios.

Each match up consists of **10,000 games**, with the starting player randomized to prevent first-turn bias. The main AI used a minimax depth of four, while the positional agent used a depth of two. All games were executed without human input to ensure consistent and reproducible results.

Results

Win/Loss Performance

Opponent Agent	AI Wins	Opponent Wins	Draws	Win Rate
Random Agent	10,000	0	0	100.0%
Greedy Agent	9,969	30	1	99.7%
Positional Agent	10,000	0	0	100.0%

Runtime Performance

Opponent Agent	Total Time (seconds)
Random Agent	2814.10
Greedy Agent	4331.81
Positional Agent	3210.94

Conclusion

The implementation of an AI agent for Connect 4 game using Minimax algorithm with alpha-beta pruning allows the AI to create a competitive AI that can challenge human opponents. By integrating a heuristic evaluation function, the AI is able to assess the board strategically, identifying a strong winnable board or a threatening board. This approach ensures that the AI can make well-informed decisions and reduce the likelihood of losing.

The AI's performance is evaluated based on its win rate against baseline agents, its efficiency in processing moves, and the correctness of its decisions in known board states. This provides a comprehensive view of how well the AI performs in a competitive context.

Overall, this project demonstrates a successful application of adversarial search techniques to make an engaging and competitive AI opponent that has room for future improvements.

Future Work

- **Multiple Difficulty Levels:** Implement different AI difficulty levels. The easier levels could involve random moves or a shallow depth minimax. We could also use a different algorithm for the easier level while the harder levels will use the current minimax function we currently have.
- **Improve game UI/UX -** Implement menu for restarting and choosing difficulty levels. Adds sound effects or background audio that can be adjusted in the menu. Make the app compatible with mobile devices.

References:

Russell, S., & Norvig, P. *Artificial Intelligence: A Modern Approach*
CPSC 481 Lecture Notes on Adversarial Games
NumPy Documentation
Pygame Documentation
Online Connect 4 Minimax Tutorials (for conceptual reference)