Samuel Oseguera

CPSC-25-10103

May 6, 2025

Coding: TicTacToe with Trees responses

Reflection:
I learned how recursion is used in decision making algorithms like the minimax algorithm present in our code. The gameState is constantly evaluated with simulations of potential future moves which change how the AI plays in response to user moves. It really gave me a glimpse into how AI can simulate different scenarios and choose its best moves. The AI did pretty good during the rounds I played against it. It managed to cut off many of my plays and didn't really do any absurd moves to put it in a losing position. There were multiple draws as well, especially when I started in the middle position (4). It did run through the same placement process during those draw rounds, so it felt as though the limitations were just a standard set of moves. However, adding the two modifications shifted its standard plays from those limitations. With more time I would aim to add visuals to the game. I feel that would make a simple terminal game like TicTacToe stand out more. Perhaps the ability to keep count of scores during a play session would be a good addition as well.

Question responses (also in the README):
1. What is the purpose of the minimax function?
The purpose of the minimax function is to evaluate all moves to determine the AIs best next move. This evaluation is based on simulations of future game states. It allows the standard "limitations" to be present in the original copy of code from the assignment.

2. How does the GameState class represent the tree?
The GameState class represents the tree as a single game state, and recursion builds the game tree by exploring possible moves. The exploring of possible moves is done by recursively applying the makeMove method.

3. When does the recursion stop in the minimax algorithm?

The recursion will stop in the minimax algorithm when a winner is found or if the board becomes full.