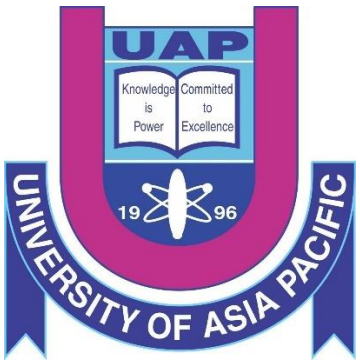


University of Asia Pacific

Department of CSE

Course Title: Artificial Intelligence Lab

Course Code: CSE 404

Project No: 02

Project Topic: Tic Tac Toe Game with Minimax and Alpha-Beta Pruning Algorithm

Submitted To:	Submitted By:
▪ Dr. Nasima Begum Associate Professor Department of CSE	▪ Injamul Haque ID: 21101001
	▪ Nawrin Rahman Antu ID: 21101027
	▪ Nahid Hasan ID: 21101029

Problem Description:

The task is to design and implement the Tic Tac Toe game as an adversarial search problem, featuring an intelligent AI player powered by the Minimax Algorithm with Alpha-Beta Pruning. The program will allow gameplay in two modes: human vs. computer and computer vs. computer. After each move, the current state of the game board will be displayed on the console, ensuring the player can follow the progress.

Tools and Technologies:

- Python (Programming Language)
- VS Code IDE (Development Environment)
- MS Word (For Documentation)

Tic Tac Toe Overview:

Tic Tac Toe is a classic two-player strategy game played on a 3x3 grid. Players alternate turns, placing their symbols ("X" or "O") in empty spaces. The objective is to align three of their symbols either horizontally, vertically, or diagonally to win. If all spaces on the board are filled and no player achieves a winning pattern, the game concludes in a draw.

Minimax Algorithm with Alpha-Beta Pruning:

Minimax Algorithm:

The Minimax Algorithm is widely used in game theory for decision-making. It involves evaluating every possible move to determine the optimal strategy for the AI. The algorithm assumes that both players act rationally, with one trying to maximize their chances of winning and the other minimizing the opponent's opportunities.

Alpha-Beta Pruning:

Alpha-Beta Pruning is an enhancement of the Minimax Algorithm, aimed at improving its computational efficiency. It eliminates unnecessary branches in the game tree that do not influence the final decision.

- **Alpha:** The best score that the maximizing player can guarantee at a given level or higher.
- **Beta:** The best score that the minimizing player can guarantee at a given level or lower.

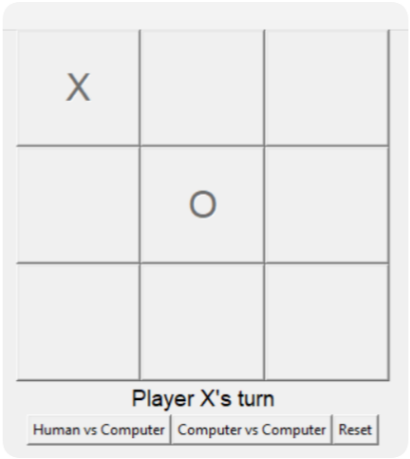
By effectively pruning irrelevant branches, the algorithm avoids unnecessary calculations, allowing deeper exploration of the game tree within the same computational budget.

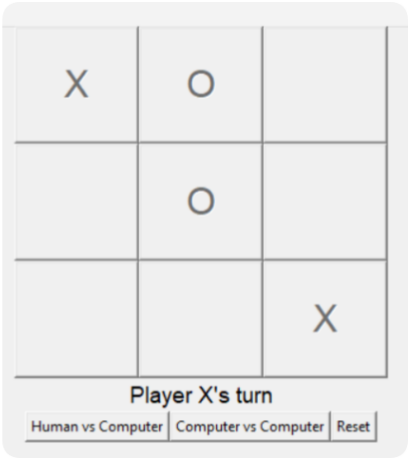
Pseudocode:

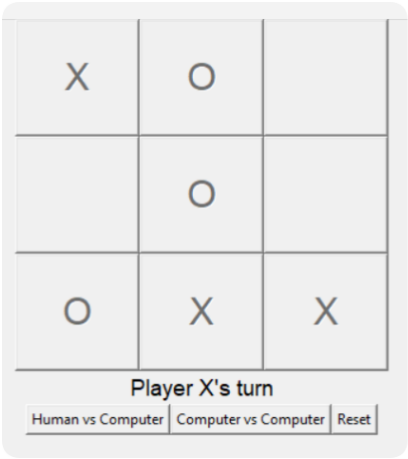
```
function AlphaBeta(node, depth, alpha, beta, isMaximizingPlayer):
  if depth is 0 or node is a terminal node:
    return the heuristic value of the node

  if isMaximizingPlayer:
    value = -infinity
    for each child of node:
      value = max(value, AlphaBeta(child, depth - 1, alpha, beta, False))
      alpha = max(alpha, value)
      if alpha >= beta:
        break
    return value

  else:
    value = +infinity
    for each child of node:
      value = min(value, AlphaBeta(child, depth - 1, alpha, beta, True))
      beta = min(beta, value)
      if beta <= alpha:
        break
    return value
```

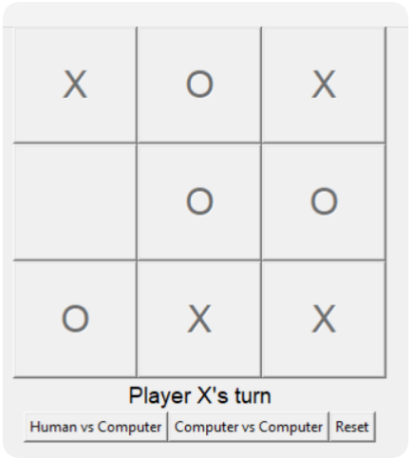
Sample Input/Output:

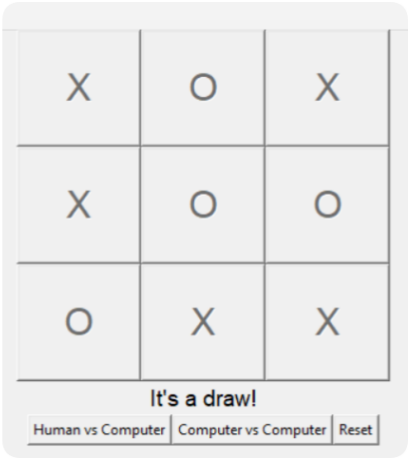
Human vs Computer:



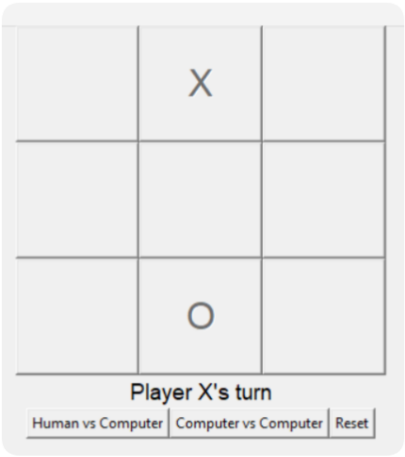


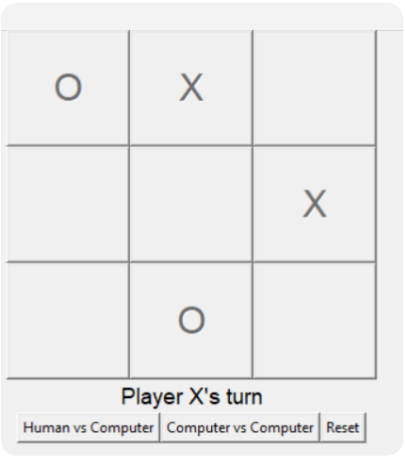


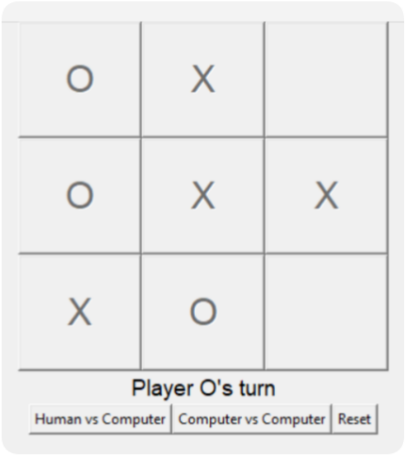


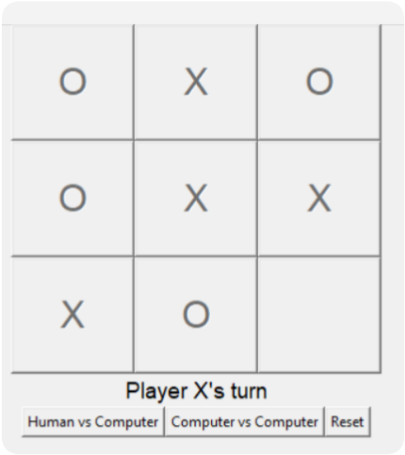


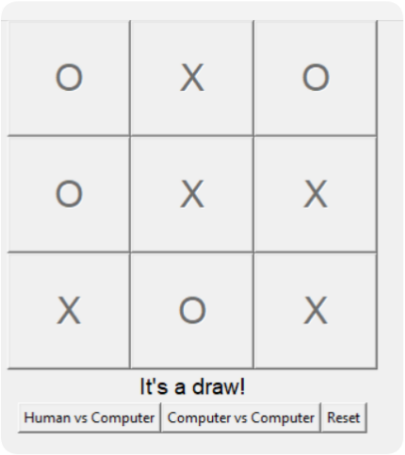
Computer vs Computer:











Challenges:

Creating a Tic Tac Toe game using the Minimax Algorithm and Alpha-Beta Pruning comes with several challenges. These algorithms can be tricky to implement, requiring a good understanding of game logic and decision-making. Optimizing the AI to make it challenging but fair is also a tough task. Making the game easy to play with a clear and simple design is essential. Dealing with issues like invalid inputs or unexpected scenarios can add complexity. Writing clean, organized code is important for debugging and improving the game in the future.

Conclusion:

Building a Tic Tac Toe game with the Minimax Algorithm and Alpha-Beta Pruning is a great way to learn about AI and game development. It introduces important concepts like decision-making, optimization, and creating smart opponents. Alpha-Beta Pruning helps make the AI more efficient, saving time while still playing well. This project improves problem-solving, coding, and debugging skills and lays the foundation for working on more advanced AI games and applications. It's both a fun and educational experience.