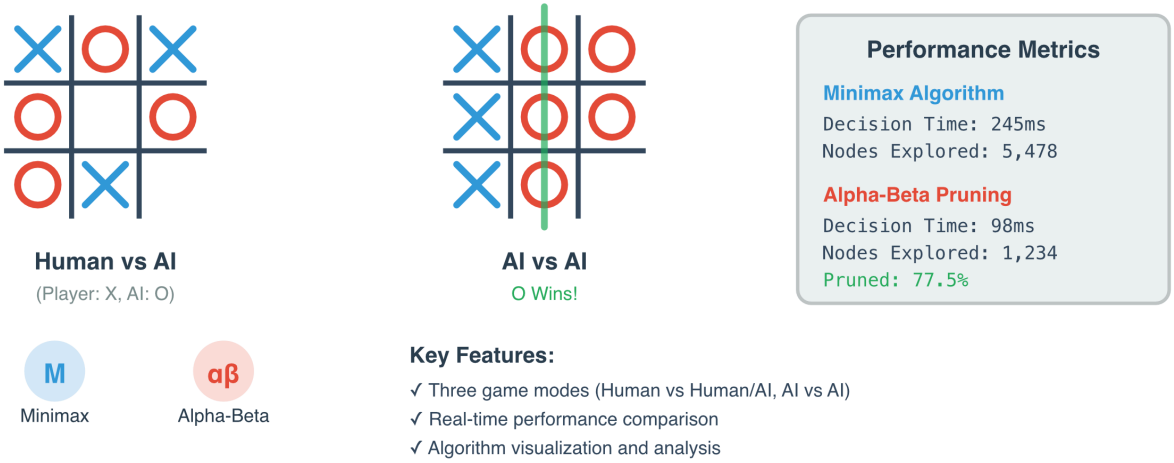
Topic: Tic Tac Toe Game with AI

Assignment Overview

This assignment requires you to develop a **Tic-Tac-Toe game** application (web-based or desktop software) with AI capabilities, comparing the performance of two classic search algorithms: **Minimax** and **Alpha-Beta Pruning**. The project should follow **GitHub** open-source project standards, including comprehensive documentation, easy-to-deploy code structure, and user-centered interaction design. Similarly, you may approach the task as a team manager by focusing on system design, and then use GenAI to handle specific coding tasks. You are expected to demonstrate your understanding of the overall work.



Game Background

Tic-Tac-Toe is a **two-player**, **zero-sum**, **deterministic**, **perfect-information**, **sequential game** played on a 3×3 board. Players take turns marking empty cells with their symbol ('X' or 'O'). The first player to align three symbols in a row (horizontally, vertically, or diagonally) wins. If the board fills up with no winner, the game ends in a draw. This game is one of the most typical examle used to introduce Minimax algorithms and adversarial search in AI.

Core Requirements

1. Your application must support the following **three** game modes:
    - Mode 1: Human vs Human
        - Two players take turns on the same device

        - Clearly indicate whose turn it is
    - Mode 2: Human vs AI
        - Human player competes against AI
        - AI uses either Minimax or Alpha-Beta Pruning algorithm
        - Allow users to select which algorithm to use
        - Allow users to choose whether to play first or second
    - Mode 3: AI vs AI (Auto-play)
        - Two AIs play against each other automatically
        - **Display step-by-step** game progression (visualize each move)
        - Show final game outcome
        - Allow selection of algorithms for both AIs (can be same or different)
2. You must implement both of the following algorithms:
    - Algorithm 1: Minimax Algorithm
        - Implement standard Minimax search tree
        - Evaluation function should consider:
            - Winning state: +10 (AI wins), -10 (opponent wins)
            - Draw state: 0
            - Search depth (optional, for early pruning)
    - Algorithm 2: Alpha-Beta Pruning
        - Implement Alpha-Beta pruning optimization on top of Minimax
        - Correctly implement pruning logic to reduce unnecessary node searches
3. You must implement visualization comparison of the following performance metrics:
    - **Decision Time**: Display time required for each AI decision (milliseconds or seconds)
    - **Nodes Explored**: Count total number of nodes searched by the algorithm
    - **Pruning Efficiency**: For Alpha-Beta, show percentage of pruned nodes
    - **Real-time Display**: Show current performance data while AI is thinking
4. Interface design must meet the following requirements:

- **Intuitive and Clear**: Users should understand how to operate without reading instructions
- **Visual Feedback**:
    - Highlight clickable cells
    - Animation effects (piece placement, winning line, etc., optional but recommended)
    - Clearly distinguish X and O symbols
- **Information Display**:
    - Current game state (whose turn, game over, etc.)
    - Currently selected algorithm
    - Performance data (as described above)
- **Control Buttons**:
    - Restart game
    - Switch game mode
    - Switch algorithm
    - Pause/Resume/Speed control for AI auto-play (optional)

Extra Credit Opportunities

You may freely add reasonable additional features to earn extra credit (maximum 20 points). Graders will assign points based on implementation quality and innovation. Extra credit **only applies to the Assignments portion** and does not affect other components of the course grade.

Submission Requirements

1. README.md or Report Document (PDF format)
    - Complete documentation as described above
    - If using PDF, still recommended to include README.md in code package
2. Complete Source Code

    - All code files, well-organized
    - Include all resource files (e.g., images, CSS, JS, etc.)
3. Executable File or Running Script
    - Web app: Directly runnable HTML file or startup script
    - Desktop app: Executable file or detailed compilation instructions
    - You should compile and package your solution properly. By default, the grader will not recompile your source code unless it is absolutely necessary, since the environment may be different.
4. Dependency List
    - `requirements.txt` (Python)
    - `package.json` (Node.js)
    - Or other appropriate dependency declaration files

Submission Format:

- Place all files in one folder
- Compress to `.zip` or `.tar.gz` format
- Upload to Canvas
    - Submissions are limited to one per group (at most 3 students). Please leave a comment to identify your team members. Once the grader evaluates your submission, they will update the score for all team members.