Project Proposal: AI-Powered Grand Tic-Tac-Toe

Project Title: AI for Grand Tic-Tac-Toe

Submitted By: Zoha [22k-4644], Misha [22k-4179], Aliza [22k-4566]

Course: AI

Instructor: Alina Arshad, Almas Ayesha Ansari

Submission Date: 22-03-2025

---

1. Project Overview

Project Topic: This project focuses on developing an AI-powered Grand Tic-Tac-Toe game, where each box of the main Tic-Tac-Toe grid contains a smaller Tic-Tac-Toe board. The AI opponent will use the Minimax algorithm with Alpha-Beta Pruning for decision-making and potentially Reinforcement Learning for adaptive gameplay.

Objective: The goal is to create an intelligent AI capable of playing Grand Tic-Tac-Toe efficiently. The AI will analyze moves at both the micro (small grids) and macro (main grid) levels, making optimal decisions based on board states and strategic heuristics.

---

2. Game Description

Original Game Background: Tic-Tac-Toe is a classic 3x3 grid-based game where two players take turns marking spaces with “X” and “O.” The objective is to get three marks in a row (horizontally, vertically, or diagonally) before the opponent. Grand Tic-Tac-Toe extends this concept by embedding a smaller Tic-Tac-Toe board inside each of the nine spaces of a larger grid.

Innovations Introduced:

- **Nested Grid System:** Each of the nine cells in the main grid contains a separate 3x3 TicTac-Toe board.
- **Turn-Based Restrictions:** The placement in a small grid determines the next valid board for the opponent’s move.
- **AI Implementation:** The AI will consider both local (small board) and global (large board) strategies.

- **Game Modes:**
  - **Player vs. AI Mode:** The player competes against the AI opponent.
- **Graphical User Interface (GUI):** Implemented using Pygame for an interactive experience.

---

3. AI Approach and Methodology

AI Techniques to be Used:

- **Minimax Algorithm:** Evaluates all possible moves and selects the best one.
- **Alpha-Beta Pruning:** Optimizes Minimax by reducing the number of nodes evaluated.
- **Reinforcement Learning (Optional):** AI can learn strategies through gameplay.

Complexity Analysis:

- The nested grid structure significantly increases the game tree complexity compared to standard Tic-Tac-Toe.
- Minimax with Alpha-Beta Pruning will help optimize decision-making.
- Reinforcement Learning, if implemented, will introduce training complexity but improve adaptability.

---

4. Game Rules and Mechanics

Modified Rules:

- Players alternate turns, marking “X” or “O” in a small Tic-Tac-Toe grid.
- The placement in a small grid determines the next board where the opponent must play.
- Winning a small grid grants control of that space in the main grid.
- The game is won by securing three won grids in a row in the main board.

Winning Conditions:

- A player wins by securing three small-grid victories in a row (horizontally, vertically, or diagonally) on the main board.
- If all grids are filled without a main board winner, the game ends in a draw.

Turn Sequence:

- The first player chooses any grid to start.

- Subsequent moves are restricted to the grid corresponding to the last move's position in its small grid.
- If a chosen grid is already won or full, the player can choose any open grid.

---

5. Implementation Plan

Programming Language: Python

Libraries and Tools:

- **Pygame:** For the graphical user interface.
- **NumPy:** For handling board states and computations.
- **AI Libraries (Optional):** TensorFlow/PyTorch for reinforcement learning (if implemented).

Milestones and Timeline:

- **Week 1-2:** Game design and rule finalization.
- **Week 3-4:** Implementation of Minimax algorithm and heuristic evaluation.
- **Week 5-6:** Integration of GUI and game mechanics.
- **Week 7:** AI testing and difficulty level adjustments.
- **Week 8:** Final testing, optimizations, and report preparation.

---

6. References

- Tic-Tac-Toe AI Algorithms: https://en.wikipedia.org/wiki/Minimax
- Python Pygame Documentation: https://www.pygame.org/docs/