Project Report

# Title:

AI for Grand Tic-Tac-Toe

# Submitted By:

Zoha [22k-4644]  
Misha [22k-4179]  
Aliza [22k-4566]

# Course:

Artificial Intelligence (AI)

# Instructors:

Alina Arshad  
Almas Ayesha Ansari

# Submission Date:

22-03-2025

# 1. Introduction

This project presents an AI-powered variant of the classic game, Tic-Tac-Toe, called Grand Tic-Tac-Toe. Unlike the standard version, it involves a nested grid system where each main grid cell contains another 3x3 board. The project implements AI decision-making using the Minimax algorithm with Alpha-Beta Pruning, providing a challenging experience for the human player.

# 2. Objectives

- Design a scalable Grand Tic-Tac-Toe game.

- Develop an AI opponent capable of optimal gameplay.

- Integrate a graphical user interface (GUI) for interactivity.

- Experiment with different difficulty levels using search-depth control.

# 3. Game Description

Grand Tic-Tac-Toe is a sophisticated extension of the classic 3x3 Tic-Tac-Toe game. In this variant, the game board consists of a 3x3 grid where each cell itself contains another 3x3 board, resulting in a total of 81 playable cells. Players must win individual small boards to gain control over the corresponding cell in the main board. The strategic twist is that each move determines which small board the opponent must play in next. To win the overall game, a player must secure a line of three won small boards (horizontally, vertically, or diagonally) on the main board. This nested structure introduces deeper levels of strategy and planning, making the gameplay significantly more complex and engaging.

# 4. Features Implemented

- GUI: Built using Tkinter with dynamic button states, color-coded status, and active board highlighting.

- Game Modes: Human vs AI

- Player Choice: Player selects either 'X' or 'O' at the start.

- Move Restrictions: Based on the last move’s cell index.

- Win & Draw Detection: Both for small and main boards.

- Restart Option: Resets the game state cleanly.

- AI Logic:

- Minimax Algorithm with optional depth limit (3).

- Alpha-Beta Pruning improves efficiency.

- Evaluation based on current board heuristics.

# 5. AI Methodology

**5.1 Minimax Algorithm**

- Simulates future moves.

- Scores game states to maximize AI's chance of winning.

**5.2 Alpha-Beta Pruning**

- Reduces the number of nodes evaluated.

- Improves performance significantly.

**5.3 Heuristics**

- Win = +10 (AI), -10 (Player)

- Draw = 0

- Limited-depth search prevents combinatorial explosion.

# 6. Tools and Technologies

- Programming Language: Python

- Libraries Used:

- Tkinter: GUI development

- math, copy: Logic support

# 7. Challenges Faced

- Managing dynamic board states across 81 cells.

- Implementing AI decision-making efficiently in a nested game structure.

- Ensuring valid move restrictions and smooth GUI updates.

# 8. Future Improvements

- Add Reinforcement Learning (Q-learning) for adaptive gameplay.

- Introduce a 2-player local mode.

- Offer difficulty levels by tuning search depth or randomness.

- Add undo functionality.

# 9. Conclusion

This project demonstrates how classical AI techniques like Minimax and Alpha-Beta Pruning can be applied to complex games like Grand Tic-Tac-Toe. The result is an intelligent, interactive game that challenges players both visually and logically.

# 10. References

- Minimax Algorithm: https://en.wikipedia.org/wiki/Minimax

- Python Tkinter Documentation: https://docs.python.org/3/library/tk.html