Project Documentation

ONE PLAYER 3D TIC TAC TOE GAME WITH 64 SLOTS

ITCS 6150 - Intelligent Systems: Fall 2023

- **BY DR. ZBIGNIEW W RAS**

Submission Details - Group 23
Ashka Ashani – 801318317
Yogen Ghodke – 801312647
Mahesh Hasbi – 801315534
Hima Priya Geridipudi – 801308047

Table of Contents

Introduction

The project aims to implement a 3D Tic Tac Toe game in Python with a single player against an AI opponent equipped with three distinct difficulty levels: easy, difficult, and insane. Unlike the traditional 2D Tic Tac Toe, this game will have a 3D playing field with 64 slots arranged in a 4x4x4 grid. The player and the AI will take turns placing their markers in an attempt to form a line of four markers in any direction within the 3D space.

Project Description

The inspiration for this project comes from the implementation of a 2D Tic Tac Toe game, as demonstrated in the provided link [https://myplugins.net/coding-tic-tac-toe-gui/]. However, this project introduces an additional dimension to the game, making it more challenging and engaging.

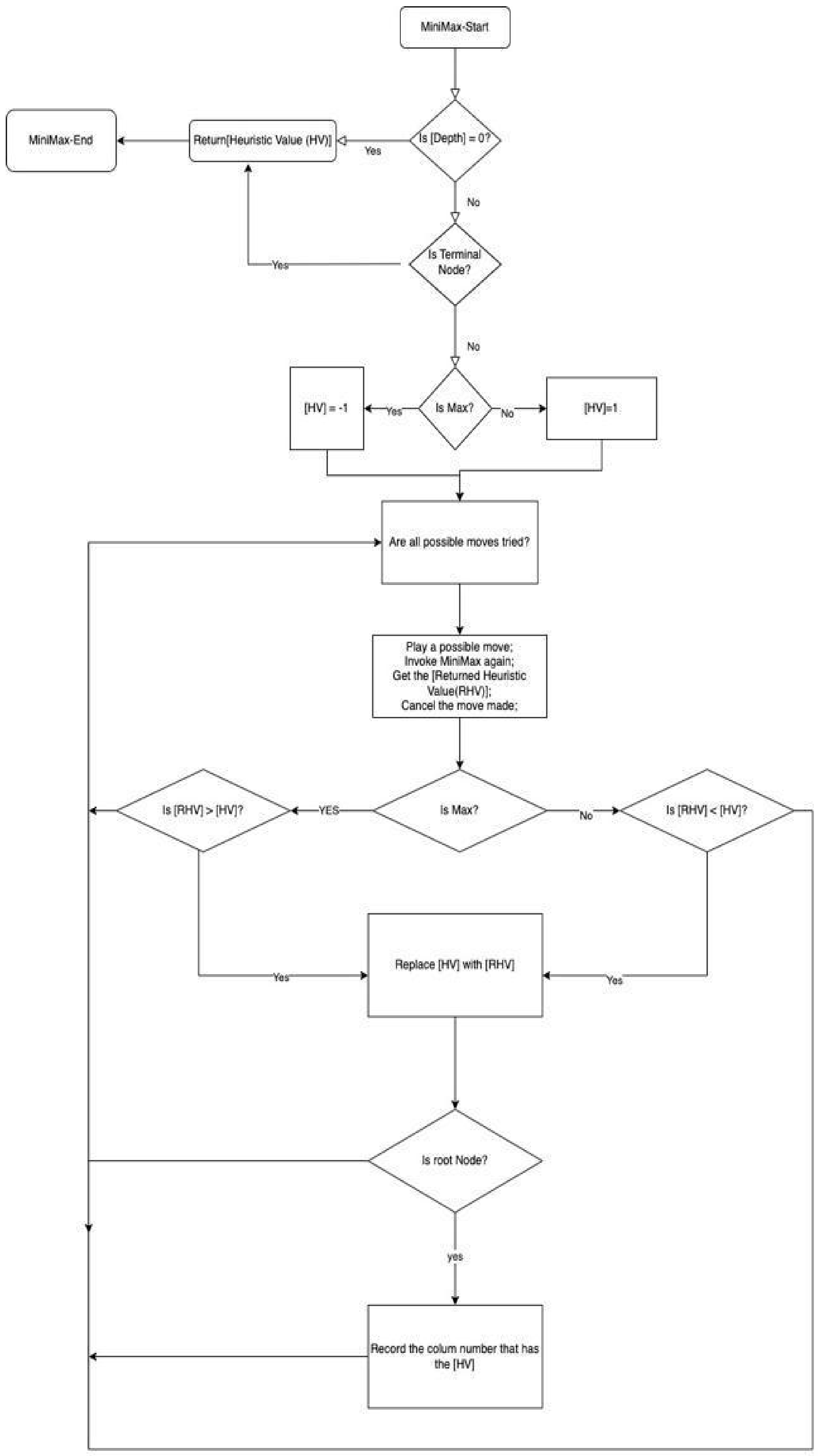
To make the game competitive and entertaining, the implementation will incorporate Mini-Max and Alpha-Beta pruning procedures for the AI opponent. The Mini-Max algorithm is a decision-making algorithm used in two-player games, such as Tic Tac Toe, to determine the best move for a player, assuming the opponent also makes optimal moves. The Alpha-Beta pruning technique enhances the efficiency of the Mini-Max algorithm by eliminating unnecessary branches in the search tree, reducing the number of nodes evaluated.

The project will involve designing a graphical user interface (GUI) for a seamless and user-friendly gaming experience. The GUI will display the 3D game board and allow the player to interact with the game by making moves. The AI opponent will use the Mini-Max and Alpha-Beta procedures to make intelligent moves, providing a challenging opponent for the player.

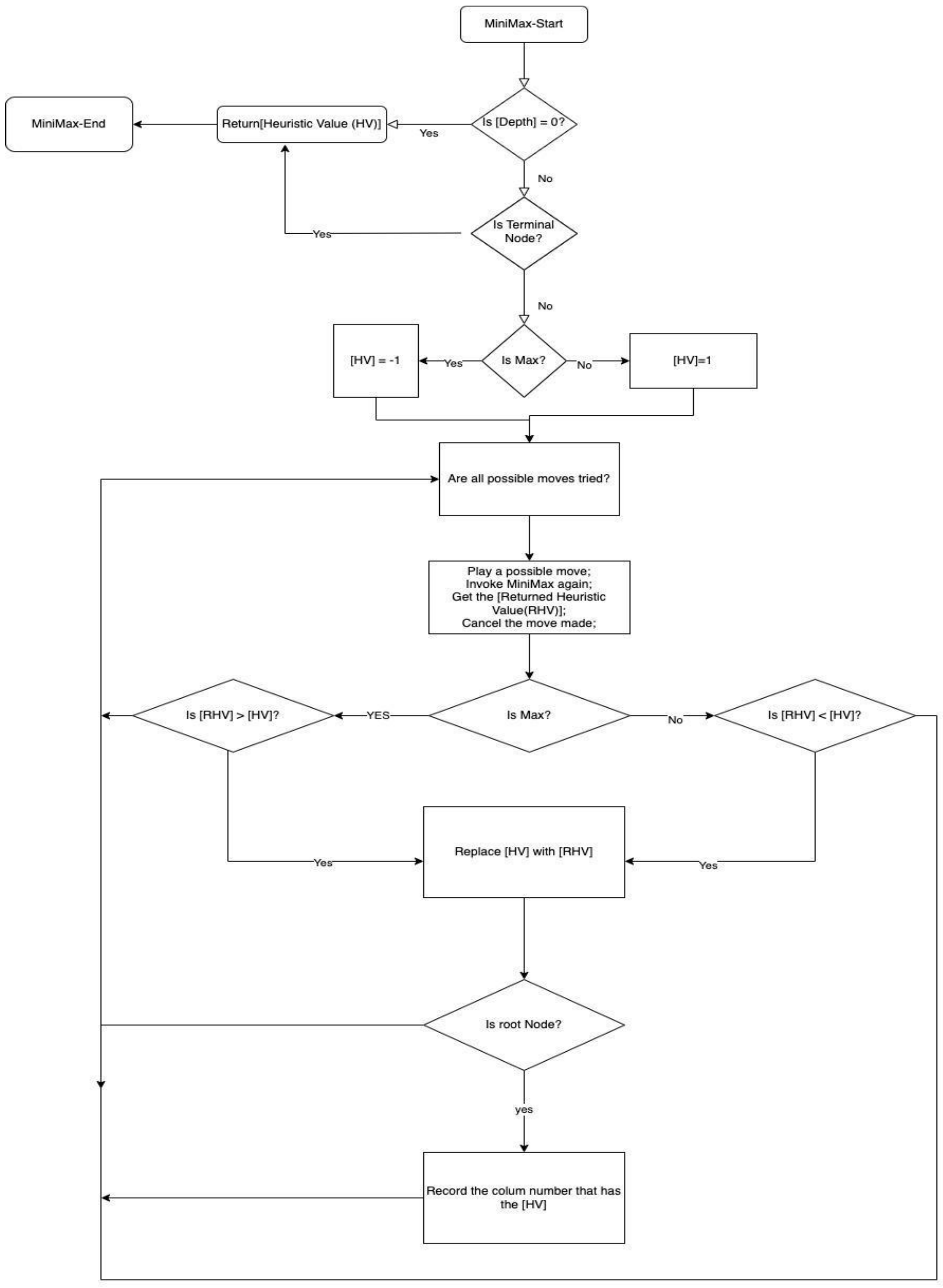
The implementation will follow the principles outlined in the provided resources [https://www.geeksforgeeks.org/implementation-of-tic-tac-toe-game/] for understanding and incorporating the Mini-Max and Alpha-Beta procedures into the 3D Tic Tac Toe game.

Overall, this project combines elements of game development, artificial intelligence, and graphical user interface design to create an entertaining and intellectually stimulating gaming experience for a single player. The end result will be a fully functional 3D Tic Tac Toe game with a responsive AI opponent, offering an enjoyable and challenging gaming experience.

System Flow Diagram

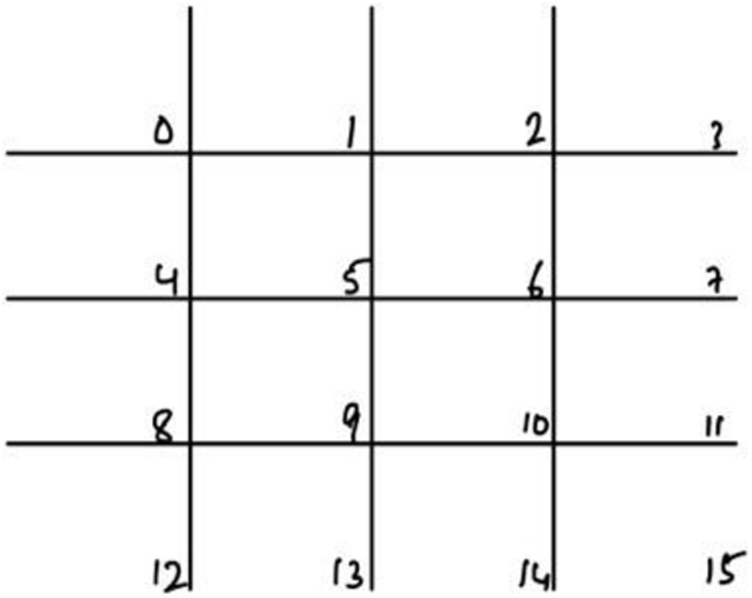


Data Flow Diagram

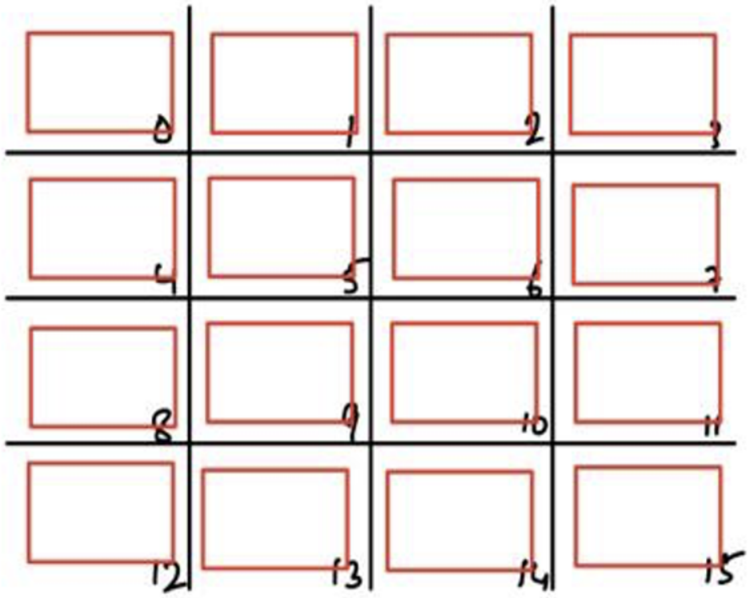

MiniMax-Start
Is [Depth] = 0?
Yes
Return[Heuristic Value (HV)]
MiniMax-End
No
Is Terminal Node?
Yes
No
Is Max?
Yes
[HV] = -1
No
[HV]=1
Are all possible moves tried?
Play a possible move; Invoke MiniMax again; Get the [Returned Heuristic Value(RHV)]; Cancel the move made;
Is Max?
YES
Is [RHV] > [HV]?
No
Is [RHV] < [HV]?
Yes
Replace [HV] with [RHV]
Yes
Is root Node?
yes
Record the colum number that has the [HV]

Input Design

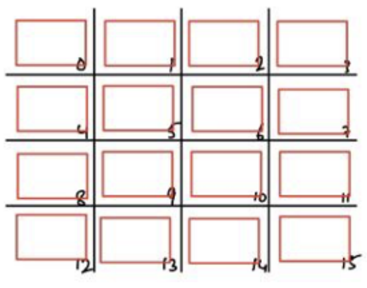
- Initial design with one board

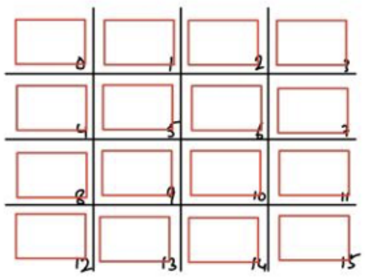


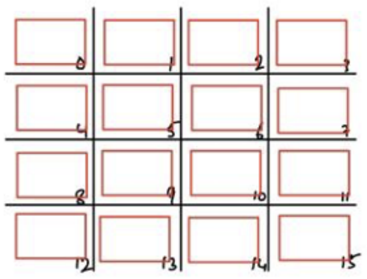
- Initial design with buttons on only board 1

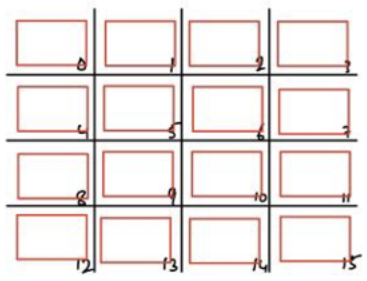


- we made multiple boards which now has 64 spaces.



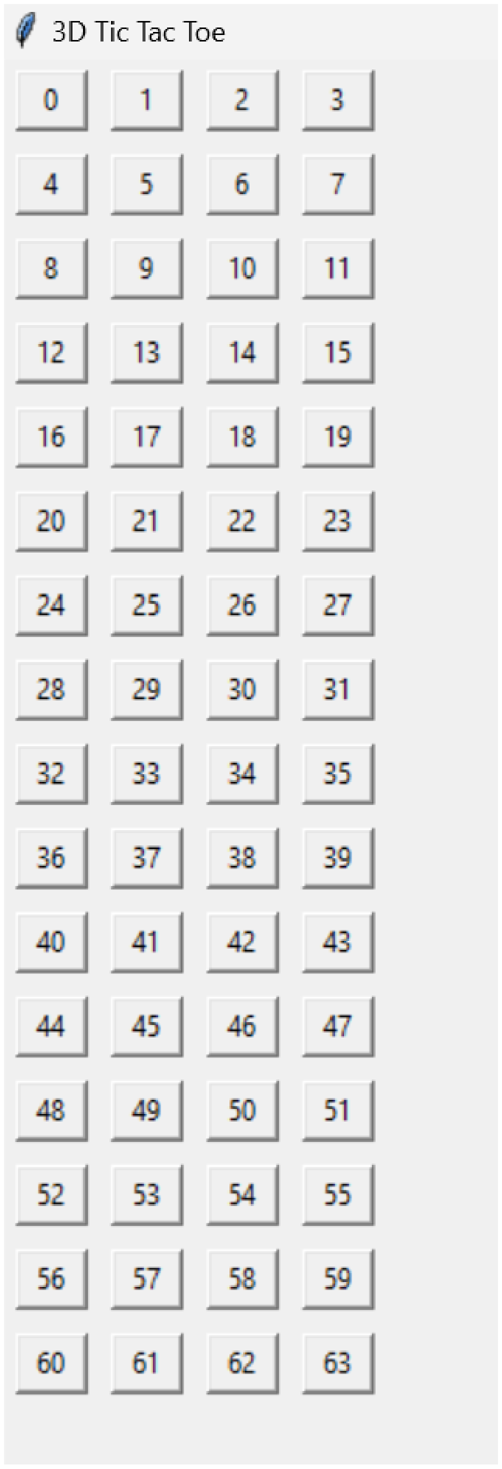


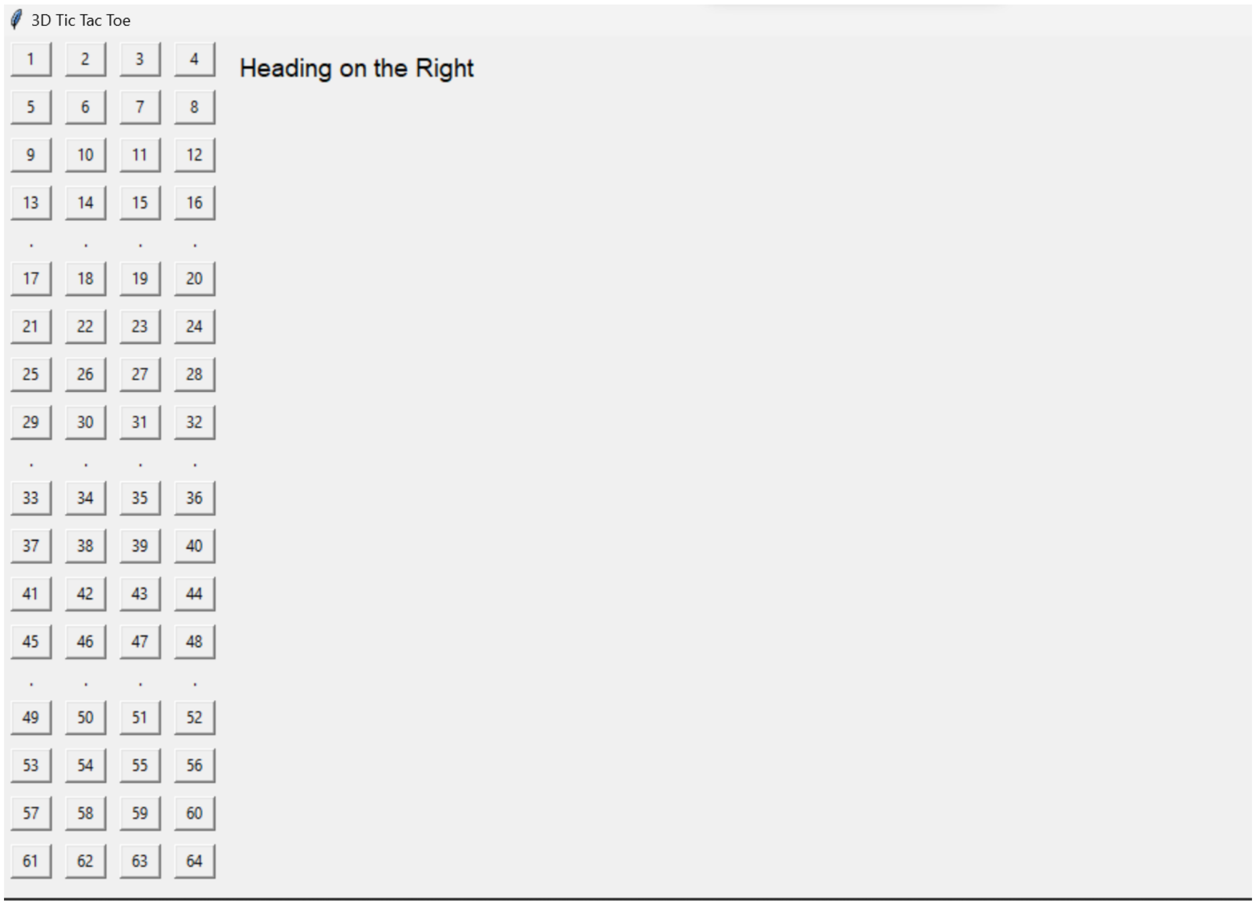




Output Design

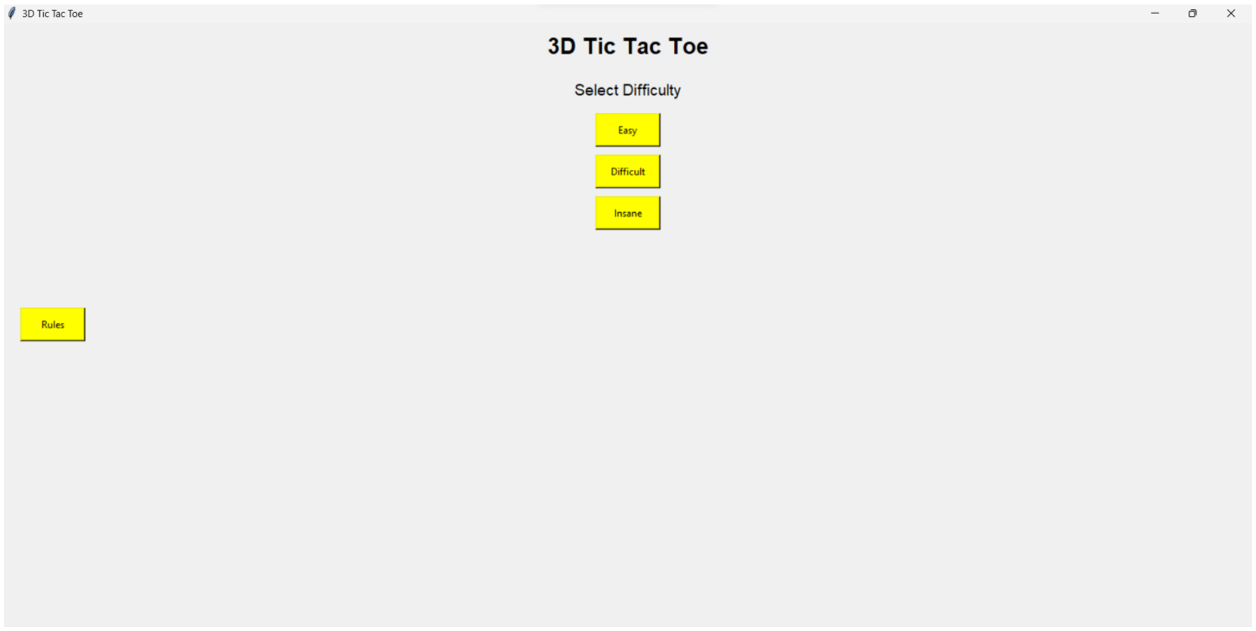
Output design for Tic Tac Toe with 64 slots:





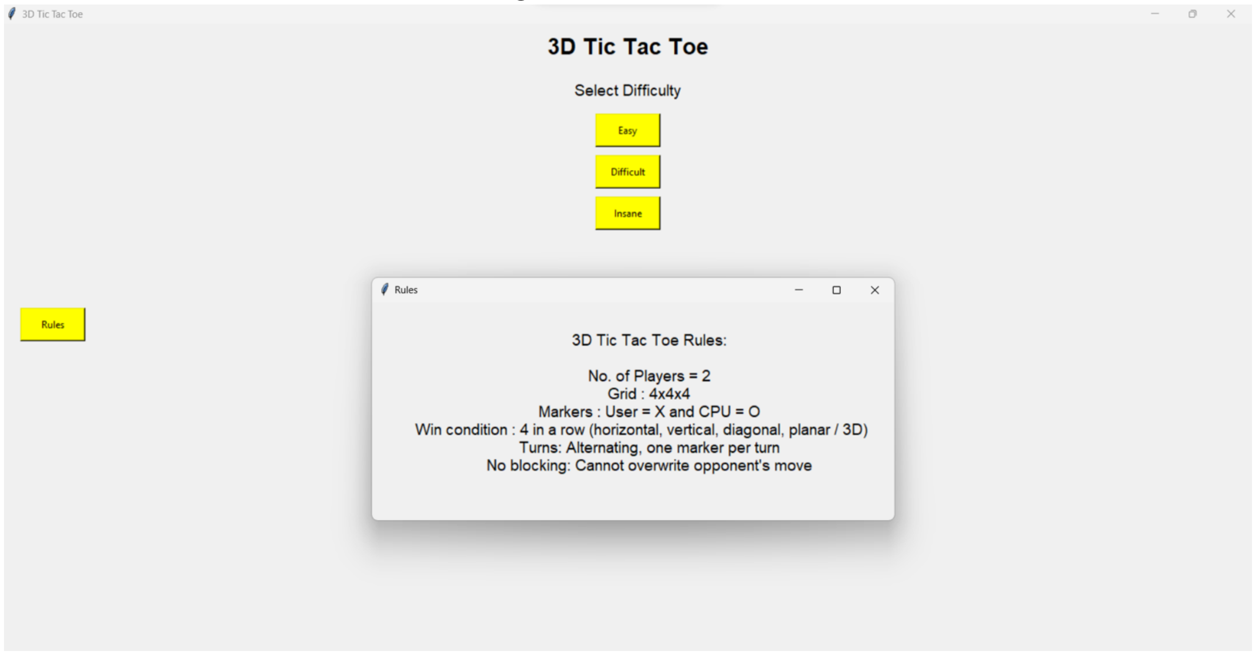
<u>TIC-TAC-TOE Graphical User Interface</u>

Step 1: Home Screen



Step 2: Game rules

Press the rules button to learn about the game's rules.

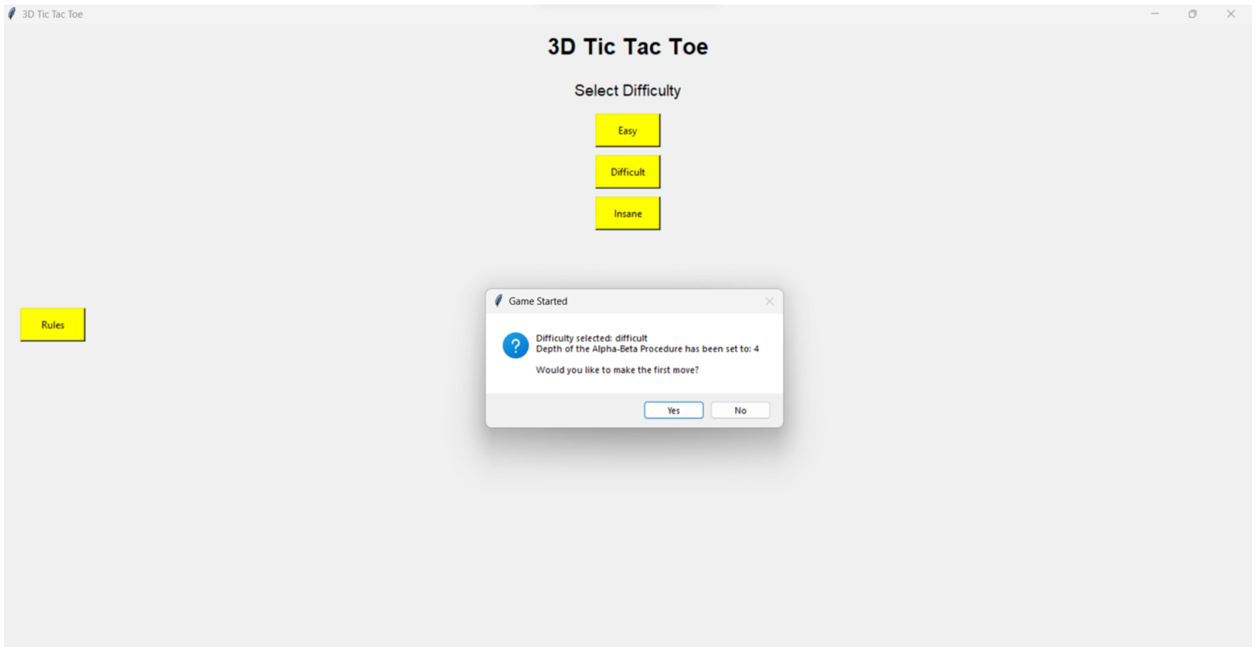


3D Tic Tac Toe Rules:

No. of Players = 2
Grid : 4x4x4
Markers : User = X and CPU = O
Win condition : 4 in a row (horizontal, vertical, diagonal, planar / 3D)
Turns: Alternating, one marker per turn
No blocking: Cannot overwrite opponent's move

Step 3: Choose Difficulty Level

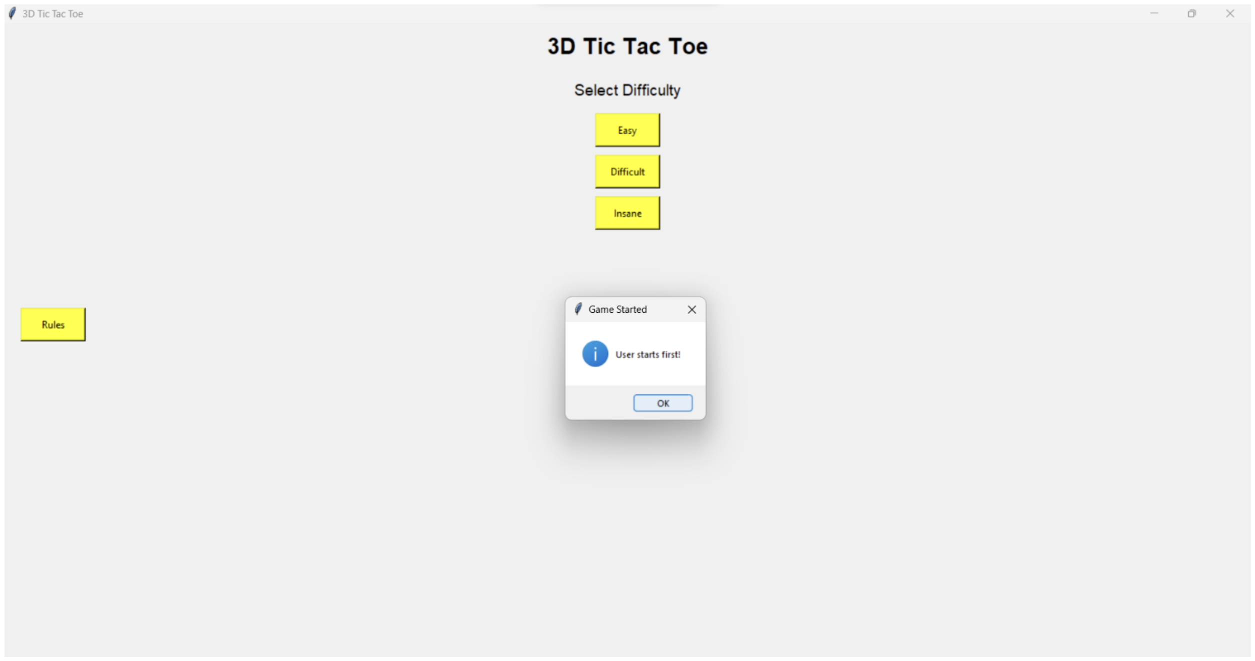
Choose the game difficulty by clicking on the difficulty levels.



Difficulty selected: difficult
Depth of the Alpha-Beta Procedure has been set to: 4

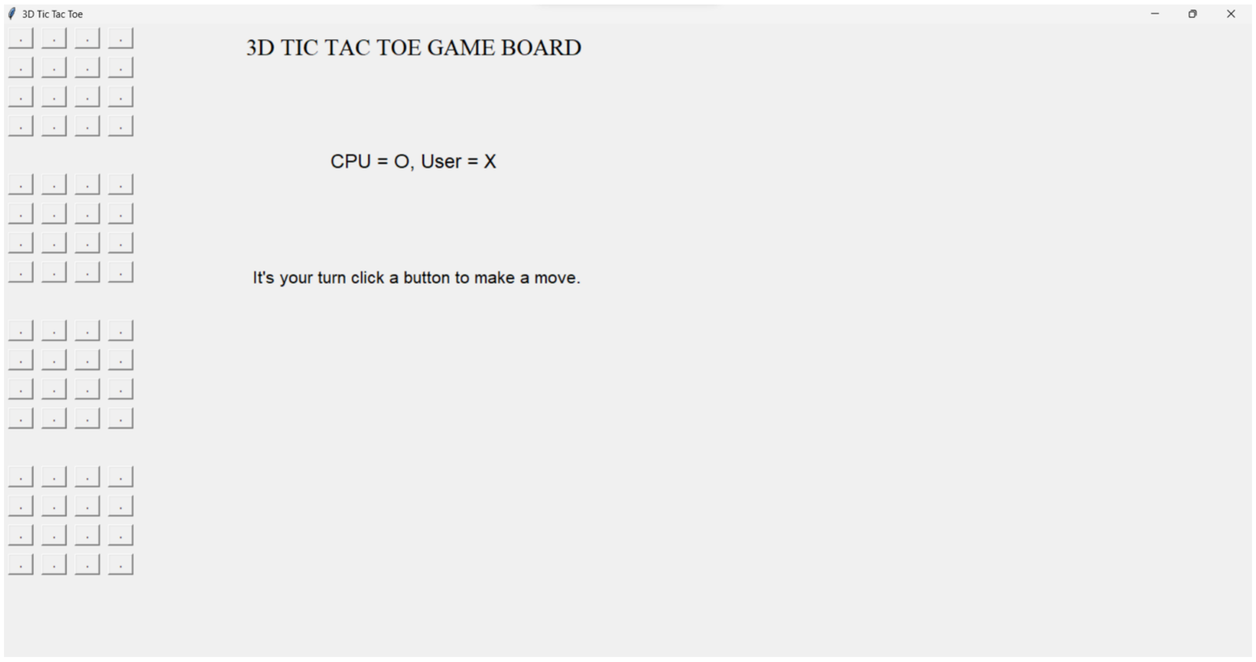
Would you like to make the first move?

Step 4: Choose either Yes or No to decide whether to start first or let the computer start first.



User starts first!

Step 5: Play the game by clicking on the buttons on the left.

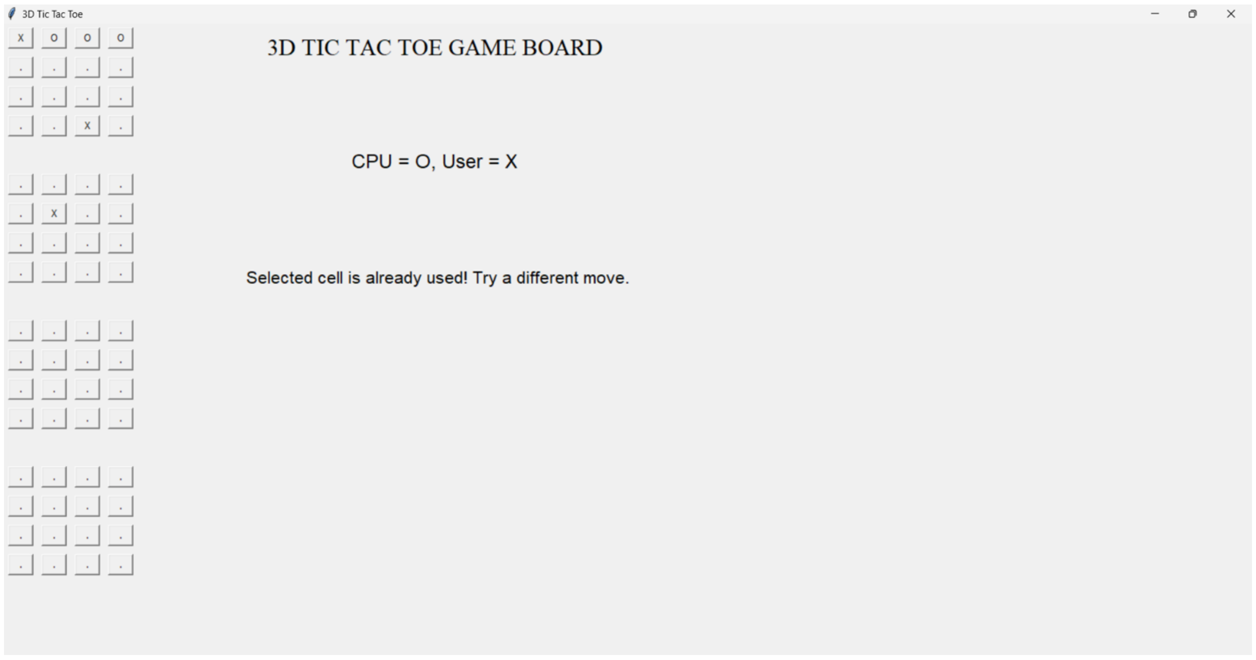


3D TIC TAC TOE GAME BOARD

CPU = O, User = X

It's your turn click a button to make a move.

Error Thrown when you try to overwrite the opponent's move:

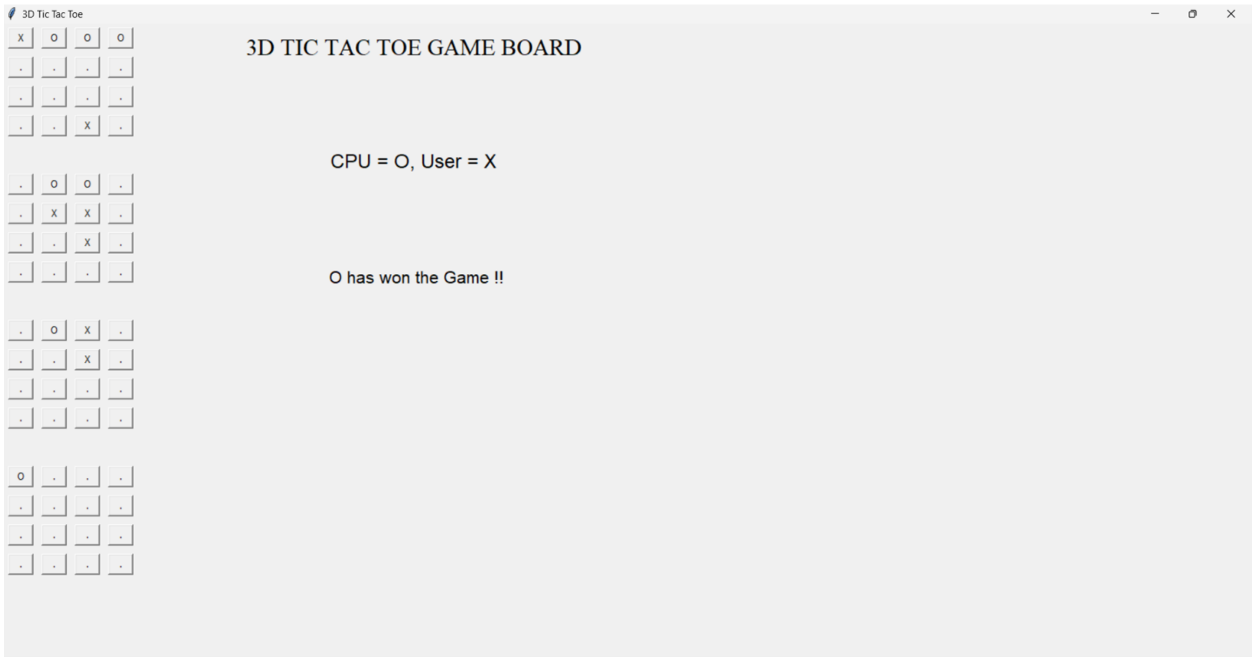


3D TIC TAC TOE GAME BOARD

CPU = O, User = X

Selected cell is already used! Try a different move.

Winner Declaration: AI WON



Features

3D Tic Tac Toe Experience

- **Immersive Gameplay:** Explore a 4x4x4 board, transcending traditional Tic Tac Toe boundaries.
- **Strategic Depth**: With 76 winning lines, including vertical, horizontal, and diagonal combinations, the game introduces complexity beyond the conventional 3x3 grid.

AI Opponents

- **Monte Carlo Tree Search (MCTS):** Face off against an MCTS-powered agent, utilizing random playouts and exploration-exploitation strategies.
- **Minimax Algorithm:** Challenge a Minimax agent with alpha-beta pruning, providing optimal moves by recursively exploring the game tree.

Difficulty Levels

- **Easy, Difficult, Insane**: Tailor the challenge to your preference with three difficulty levels, adjusting the depth of alpha-beta procedures for varied gameplay experiences.

Advanced Algorithms

- **Alpha-Beta Pruning:** Enhance efficiency in decision-making with alpha-beta pruning, reducing unnecessary node evaluations.

- **Forward Pruning:** Optimize performance by selectively exploring moves, especially at higher depths, for a more focused and efficient search.

Implementation Details

Game Logic

- Game Board Representation: Utilize a 4x4x4 matrix to represent the game board, with each cell accommodating player symbols.
- Winning Conditions: Implement the 76 winning lines, considering layers, rows, columns, and diagonals across the 3D space.

Monte Carlo Tree Search (MCTS)

- Selection: Use the UCB1 metric for node selection, exploring potential child nodes until reaching a leaf node.
- Expansion: Dynamically create child nodes based on available moves, simulating further gameplay.
- Simulation: Conduct random playouts from selected nodes to gather information for decision-making.
- Backpropagation: Update node statistics based on simulation outcomes, refining the tree for future selections.
- Best Move Evaluation: Define the "best" move as the one minimizing the opponent's win ratio.

Game Board

Board Representation

The game board is a 4x4x4 matrix, creating a three-dimensional space for an engaging 3D Tic Tac Toe experience. Each cell within this matrix serves as a potential placement point for player symbols.

Winning Lines

The board features a total of 76 winning lines, adding a layer of complexity to the classic Tic Tac Toe concept. These winning lines comprise combinations in individual layers, rows, columns, and diagonals spanning the three-dimensional grid.

Winning Line Distribution

- **Individual Layers (10 per layer):** Each of the four layers contributes 10 winning lines, covering rows, columns, and diagonals within that specific layer.
- **Inter-Layer Combinations (36 total):** Imagining all possible winning combinations in the third dimension, considering lines that contain a single piece in each of the layers. These contribute an additional 36 winning lines.

Player Interaction
Players take turns placing their symbols ('X' or 'O') in empty cells on the board. The objective is to achieve four of their symbols in a row, either vertically, horizontally, or diagonally, to secure victory.

Visual Representation
The game board, while internally represented as a 4x4x4 matrix, is visually presented in a user-friendly format, enabling players to make strategic moves within the 3D space. The graphical representation enhances the gaming experience, allowing for intuitive and immersive gameplay.

Alpha-Beta Procedure

- Minimax Algorithm: Implement the minimax algorithm for optimal decision-making.
- Alpha-Beta Pruning: Cut off sub-trees based on alpha-beta values, reducing computational complexity.
- Move Sorting: Enhance efficiency by sorting moves with a heuristic function, facilitating better alpha-beta pruning.

Difficulty Levels

Easy
- Depth: 2 levels
- Description: In the easy difficulty level, the alpha-beta procedure explores the game tree up to a depth of 2. This provides a quick and straightforward gaming experience, suitable for those looking for a casual or introductory challenge.

Difficult
- Depth: 4 levels
- Description: The difficult difficulty level ramps up the challenge by exploring the game tree up to a depth of 4 using the alpha-beta procedure. Players can expect a more strategic and competitive gameplay experience, requiring thoughtful moves and planning.

Insane
- Depth: 6 levels
- Description: For the bravest players seeking the ultimate challenge, the insane difficulty level pushes the boundaries with a depth of 6 in the alpha-beta procedure. This level of depth demands advanced strategic thinking and offers a highly competitive environment.

How to Play

Player Symbols

- X: Represents the human player.
- O: Represents the AI player.

Game Flow

1. Start: The game begins with player selecting the difficulty level of the game.
2. Move Selection: Choose either Yes or No to decide whether to start first or let the computer start first and start playing the game.
3. Objective: The goal is to achieve four symbols in a row, either vertically, horizontally, or diagonally, to win the game.
4. Difficulty Selection: Players choose the desired difficulty level: easy, difficult, or insane.
5. Gameplay: The alpha-beta procedure, based on the selected difficulty, determines the computer's moves for challenging gameplay.
6. End: The game concludes when a player achieves a winning configuration, and scores are updated accordingly.

Installation

To install the game, follow these steps:

1. Clone Repository:
   ```
   git clone https://github.com/yogen-ghodke-113/IS_23.git
   cd IS_23
   ```
2. Install numpy using pip
   ```
   pip install numpy
   ```
3. Run the Game:
   ```
   python main.py
   ```
4. Play the game using GUI.

Usage

Command Line

- Launch the game by running python main.py in the command line.
- Follow on-screen prompts to choose player, set difficulty levels, and make moves.

Graphical User Interface (GUI)

- Utilize the user-friendly GUI for an interactive gaming experience.
- Click on cells to place player symbols and navigate through the game.

Contributors

This project has been made possible through the collaborative efforts of the following contributors:

- Ashka Ashani
- Yogen Ghodke
- Mahesh Hasbi
- Hima Priya Geridipudi

Acknowledgments

We extend our gratitude to the following sources for their valuable insights, references, and inspirations:

1. Wolfram Demonstrations - 3D Tic Tac Toe
2. Math Is Fun - Foursight 3D Tic Tac Toe - This resource provided valuable insights into the gameplay and strategies of 3D Tic Tac Toe.
3. GeeksforGeeks - Implementation of Tic Tac Toe Game - The implementation details and coding concepts from GeeksforGeeks have been a helpful reference in our project development.
4. Wikipedia - 3D Tic Tac Toe
5. Wikipedia - Alpha–beta pruning

These resources have played a crucial role in shaping our understanding and implementation of the project.

Future Improvements

As we look forward, several enhancements and additions can be considered for future iterations of the project:

- Enhanced AI Strategies: Explore more advanced strategies and algorithms for both Monte Carlo Tree Search and Minimax to further improve AI gameplay.
- Graphical User Interface (GUI) Improvements: Implement a more intuitive and visually appealing GUI for a seamless user experience.
- Multiplayer Functionality: Introduce multiplayer capabilities, allowing users to play against each other over a network.
- Customizable Settings: Provide users with options to customize game settings, such as board size and win conditions.
- Machine Learning Integration: Investigate the integration of machine learning models for adaptive AI that can learn and improve over time.

These potential improvements aim to elevate the gaming experience and add versatility to the existing project structure.