Tic Tac Toe Game Report

Introduction

The project involves developing a Tic Tac Toe game in Python with a focus on implementing artificial intelligence (AI) techniques, specifically A* and Minimax algorithms. The goal is to create an intelligent opponent that provides a challenging and strategic gaming experience for the user.

Implementation

Game structure

The Tic Tac Toe game is structured to incorporate advanced AI techniques while maintaining a user-friendly interface. The project's primary components include:

Game Logic Functions:
- print_board: Displays the current state of the game board.
- is_winner: Checks for a winning condition based on rows, columns, and diagonals.
- is_board_full: Determines if the game board is full, indicating a draw.
- get_empty_cells: Retrieves the coordinates of empty cells on the board.
- minimax: Implements the Minimax algorithm for strategic AI moves.
- find_best_move: Finds the best move for the AI player using the Minimax algorithm.

User Interface with Pygame:
- draw_grid: Draws the Tic Tac Toe grid on the Pygame window.
- draw_symbols: Renders 'X' and 'O' symbols on the Pygame window.
- get_indices_from_mouse: Converts mouse coordinates to board indices.

Symbol Selection and Tkinter Pop-up:
- draw_button: Draws a button on the Pygame window for symbol selection.
- get_user_symbol: Prompts the user to choose 'X' or 'O' before the game starts.
- display_game_over_message: Uses Tkinter to display end-of-game messages and options.

AI implementation

A* algorithm (Additional Feature)

While the initial project description mentioned A* as part of the AI techniques, the current implementation focuses on the Minimax algorithm. To incorporate the A* algorithm, modifications are needed in the game logic to define a heuristic and implement the search

algorithm. A* could be applied to find the optimal path to victory, enhancing the AI's decision-making process.

Minimax algorithm

The Minimax algorithm is employed to make strategic decisions for the AI opponent. It recursively evaluates potential moves, considering the best and worst outcomes, to select the move that maximizes its chances of winning or minimizes the chances of losing.

Design choices

Pygame and tkinter integration

The choice to integrate Pygame for real-time graphics and Tkinter for end-of-game messages is intentional. Pygame provides an interactive game interface, while Tkinter simplifies the display of messages and options, creating a cohesive user experience.

User symbol selection

The inclusion of a symbol selection feature at the beginning of the game allows users to personalize their gaming experience. They can choose between 'X' and 'O,' influencing the strategy they adopt during the game.

Modular code structure

The code is organized into modular functions, facilitating readability and potential future enhancements. The separation of concerns allows for easy maintenance and modifications.

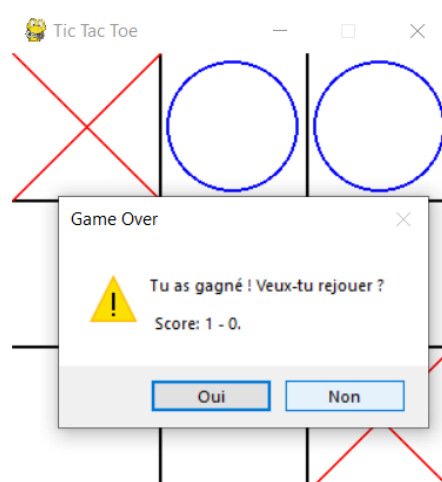# Steps with graphical interface

- At the start of the game, choose your symbol ('X' or 'O') by clicking the corresponding button.
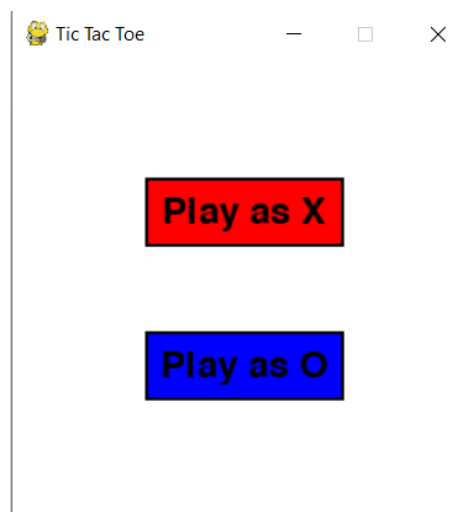
- The game board is a 3x3 grid. Make your move by clicking on an empty cell, because the first player is user by default.
- After your move, the AI opponent automatically makes its move.
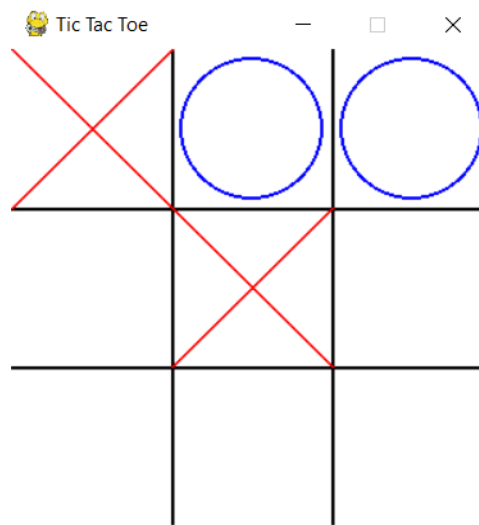
- Continue playing until there's a winner, a draw, or you choose to quit.

- After each game, a message displays the outcome with scores that show the number of wins for both the player and the AI opponent also options to play again or quit.
- If you choose to play again, you have to choose your symbol again.

Conclusion

The project aims to deliver an intelligent Tic Tac Toe game by incorporating both the A* and Minimax algorithms. The utilization of Pygame and Tkinter ensures an engaging and visually appealing user interface. The project's success lies in providing users with a challenging opponent and a personalized gaming experience. Future developments may focus on the integration of the A* algorithm and additional features to enhance the game further.