Tic-Tac-Toe AI Project

OVERVIEW

This project implements a Tic Tac Toe game with varying difficulty levels. The game involves the player competing against an AI with different game modes.

ALGORITHMS USED

- **Minimax (Hard Mode)**
  The minimax algorithm is a brute force technique used in hard mode. It explores all possible game states to determine the optimal move for the AI, ensuring that the AI never loses if a winning move exists. This exhaustive search guarantees an unbeatable opponent, making it challenging for the player to win.
- **Straightforward (Rule-Based) and Heuristic (Medium Modes)**
  In the medium modes, two distinct AI strategies are employed. The straightforward approach evaluates all possible moves, focusing on blocking the player's winning moves and prioritizing strategic positions, such as corners and center, to increase the chances of winning.
  On the other hand, the heuristic algorithm utilizes a rule-based approach combined with specific heuristics to make more informed decisions. In addition to what the rule-based algorithm does, heuristic algorithm inculcates some pre-defined preferred moves for the AI.
- **Randomized Algorithm (Easy Mode)**
  The easy mode employs a randomized algorithm for AI moves. This approach randomly selects available positions without considering advanced strategies or game states. It offers the player a much higher chance of winning.

JUSTIFICATIONS

Minimax for Hard Mode: The minimax algorithm guarantees an unbeatable AI by exhaustively searching all possible moves. It ensures the player can't win, providing a formidable challenge.

Straightforward and Heuristic for Medium Modes: These algorithms strike a balance between challenge and winnability for the player. They focus on strategic moves and blocking opponent wins but allow some chances for the player to win, providing an engaging experience.

Differences between Rule-Based and Heuristic Algorithms: The rule-based straightforward approach evaluates all possible moves, while the heuristic algorithm incorporates specific strategies and heuristics to make more efficient and informed moves, enhancing its performance.

Randomized Algorithm for Easy Mode: The randomized AI strategy makes moves without considering complex strategies, providing an easier game for the player. It introduces unpredictability, increasing the player's chances of winning.

CODE WALKTHROUGH

Board State and Player Moves

The display_board() function prints the current state of the board in the terminal. This function is useful for debugging and understanding the game's progress during development. The check_winner() function determines if a player has won by examining rows, columns, and diagonals.

The player_move() function handles the player's move in the UI by accepting the row and column indices of the button clicked. It updates the board and checks for a win or a tie after the player's move.

Gameplay Function

The play_game() function orchestrates the gameplay. It initializes the board, handles player moves, and triggers AI moves based on the selected difficulty. After each move, it checks for a win or a tie, displaying messages accordingly using messagebox. The function ensures a smooth flow of the game by updating the UI board buttons and responding to user clicks.

User Interface

The graphical user interface is built using Tkinter, representing the game board as a 3x3 grid of buttons. Each button click triggers the respective player's move, followed by the AI's response based on the selected difficulty level.

Note: Go through the ‘hints’ folder for steps to win the medium mode.