Tic Tac Toe AI Game

Code Overview:

1) Import Libraries:

-The code begins by importing essential libraries such as copy, sys, pygame, and numpy. pygame is employed for creating the game window and handling user input.

2) Constants:

-Several constants, including WIDTH, HEIGHT, ROWS, COLS, etc., are defined in an external constants module.

3) Board Class:

-This class models the Tic Tac Toe game board, keeping track of the state of each square.

-Methods include checking the game's final state, marking squares, verifying if a square is empty, obtaining empty squares, and more.

4) AI Class:

-Represents the AI player in the game.

-Provides methods for making random moves (rnd) and utilizing the minimax algorithm for strategic moves (minimax).

-The eval method determines the main evaluation of the move to be made.

5) Game Class:

-Manages the overall game state, including the board and players.

-Offers methods for drawing the game board, making moves, changing game modes, checking game completion, and resetting the game.

6) Main Function:

-Sets up the game and enters the main loop.

-Handles pygame events like quitting, restarting, changing game mode, and choosing AI levels.

-Manages player moves and AI moves based on the chosen game mode.

Choices Made:

1) Board Representation:

-The game board is represented using a numpy array (self.squares).

-0 represents an empty square, 1 represents a cross, and 2 represents a circle.

2) Graphics and UI:

-Utilizes pygame for creating a graphical user interface.

-Draws the game board, lines, crosses, and circles using basic shapes and colors.

3) AI Strategies:

-The AI class implements two strategies: random move (rnd) and minimax algorithm (minimax).

-The AI level (0 or 1) determines which strategy to use.

4) Game Modes:

-Supports two game modes: Player vs. Player (pvp) and Player vs. AI (ai).

-The game mode can be changed during runtime by pressing the 'g' key.

5) Restart and AI Level Change:

-Pressing the 'r' key restarts the game.

-Pressing keys '0' or '1' changes the AI level.

MinMax Algorithm:

The minmax algorithm is a decision-making strategy used in two-player games like Tic Tac Toe. It explores all possible moves by both players, assigning a score to each move. The AI player then selects the move that maximizes its chances of winning and minimizes the opponent's chances.

minimax(self, board, maximizing) Method:

Inputs:

-**board:** The current game board.

-**maximizing:** A boolean flag indicating whether it's the turn of the maximizing player (True for AI's turn, False for the opponent's turn).

Outputs:

-Returns a tuple (eval, move) where:

-**eval:** The evaluation score of the current state of the board.

-**move:** The optimal move (row, col) for the AI.

Algorithm:

-The method recursively explores possible moves, considering both maximizing (AI) and minimizing (opponent) scenarios.

-The base cases check if the game is won by either player or if it's a draw.

-In the maximizing case, the algorithm chooses the move with the highest evaluation score.

-In the minimizing case, it selects the move with the lowest evaluation score.

The use of the minimax algorithm allows the AI to make strategic decisions and choose moves that lead to the best possible outcome. The depth of the search tree (i.e., how many moves ahead the algorithm looks) can be controlled by adjusting the level parameter in the AI class.