****

**A Report on**

**Tic Tac Toe Game with AI Player**

**Course name: Artificial Intelligence**

**Course code: CSE366**

**Semester: Fall 2023**

**Section: 04**

**Submitted by**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Nisarga Mridha | 2020-2-60-010 |
| Sofia Noor Rafa | 2020-1-60-226 |
| Nurzahan Akter Munni | 2020-2-60-176 |
| Md. Adib Khan | 2020-2-60-080 |

**Under the Guidance of**

Amit Mandal

Lecturer

Department of Computer Science and Engineering

East West University

**Submission Date: 28/12/23**

Table of Contents

[Title: Tic Tac Toe Game with AI Player 3](#_Toc154681480)

[Introduction: 3](#_Toc154681481)

[Objectives: 3](#_Toc154681482)

[Methodology: 3](#_Toc154681483)

[Minimax Algorithm: 3](#_Toc154681484)

[Back Tacking: 3](#_Toc154681485)

[Implementation: 4](#_Toc154681486)

[Implementation of the First and Second Page: 4](#_Toc154681487)

[Implementation of the Main Game: 5](#_Toc154681488)

[Output: 8](#_Toc154681489)

[Results: 9](#_Toc154681490)

[Conclusion: 10](#_Toc154681491)

# Title: Tic Tac Toe Game with AI Player

Introduction: Tic-tac-toe is a simple and popular two-player game that is played on a 3x3 grid. The players take turns marking the cells with their symbols, either X or O, and the goal is to form a horizontal, vertical, or diagonal line of three of their own symbols. In this project, we will implement a tic-tac-toe game in Python that can support different types of players, including human players, random computer players, and smart computer players that use the minimax algorithm to find the optimal move. The minimax algorithm is a recursive method that evaluates the possible outcomes of each move and chooses the one that maximises the player’s chances of winning or minimises the opponent’s chances of winning. The algorithm assumes that both players play optimally and can look ahead to the end of the game. We also create a graphical user interface (GUI) for the game using pygame. You can play it in PvP mode or Ai vs You. If you want to play with Ai then you will get 2 difficulty levels to choose. One is easy and another one is hard. You can also decide who will make the first move. After the game is over it will show the result which player won or is it a draw.

# Objectives:

* Being the first player to line up three of their own symbols (X or O) either horizontally, vertically, or diagonally on the 3x3 grid is the primary objective.
* At the same time, players must strategically prevent their opponents from forming three lines.
* The objective is to predict your opponent's moves and strategically place your symbols.
* The game ends if one player manages to form a line of three, or if the entire grid is filled without a clear winner, resulting in a draw.
* The game encourages players to utilise basic strategic thinking and foresight to plan their moves and anticipate their opponent's actions.
* Tic Tac Toe is designed for fast and fun gameplay and can be played by players of all ages. The simple rules allow for repeated rounds, promoting a sense of competition and fun.

# Methodology:

Minimax Algorithm: Minimax algorithm is a recursive algorithm which is used in decision-making and game theory, especially in AI games. It provides optimal moves for the player, assuming that the opponent is also playing optimally. The algorithm performs a depth-first search (DFS) which means it will explore the complete game tree as deep as possible, all the way down to the leaf nodes.

Back Tacking: It is an approach to problem-solving where we explore all possible solutions by making choices, and if a choice leads to a dead end, we backtrack and try a different choice. Here we have used backtracking in the minimax algorithm to find out the best move for the AI.

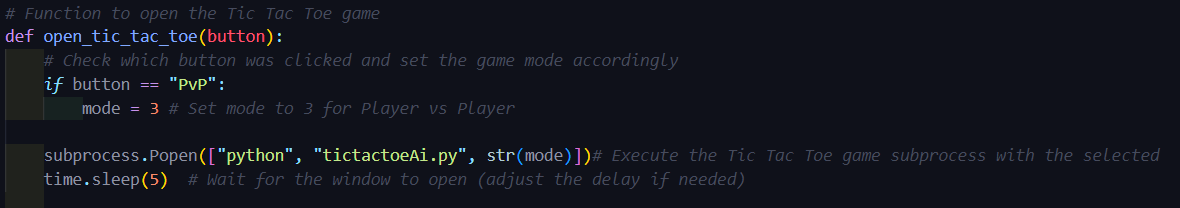
|  |  |
| --- | --- |
|  |  |

*figure 1: Minimax algorithm working process*

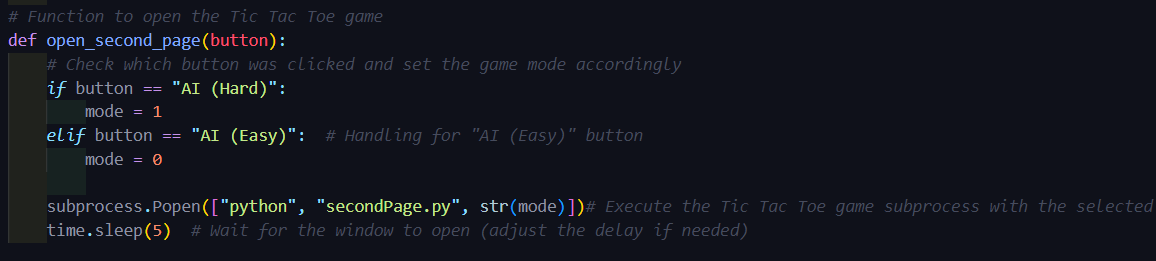
# Implementation:

## Implementation of the First and Second Page:

Multiple functions were used in the *firstPage.py.* Three options are shown on the first page. These options are for choosing the game modes. Three game modes were used in our project. They are PVP, AI (Easy), and AI (Hard). The PVP game mode can be accessed by the *open\_tic\_tac\_toe(button):* function and the AI game mode can be accessed by the *open\_second\_page(button):* function. If the player wants to play PvP mode then he can click the button and play the PvP mode. Again, if he wanted to play against AI then he had two options, AI(Easy) for easy mode and AI (Hard) for hard mode. If he clicks any of them it will redirect to the second page.

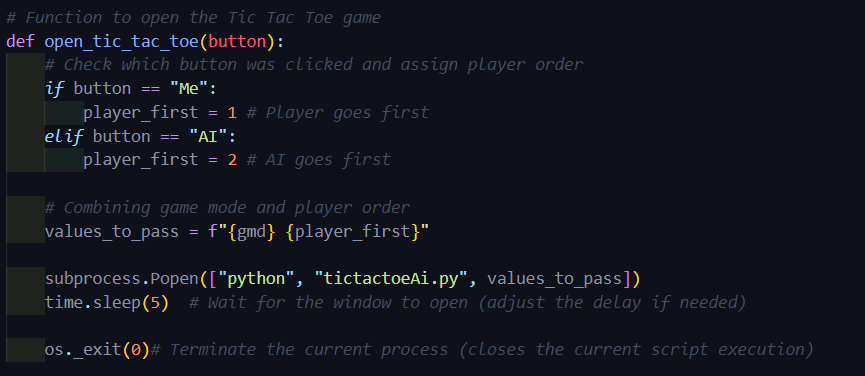


*figure 2. Functions for PvP game modes*



*figure 3. Functions for AI game modes*

The second page is a similar image to the first page. If the user chooses to play with AI on the first page, the user can decide whether he or the AI player will make the first move. Here, this feature is caused due to the function named *open\_tic\_tac\_toe(button):.* The rest of the functions are similar to the functions in the firstPage.py. This function is mainly storing values into a string then passing the string to our main tictactoeAi.py page. Where we are handling the situation based on these values. In the string we are storing the game mode value and which player will give the first move.

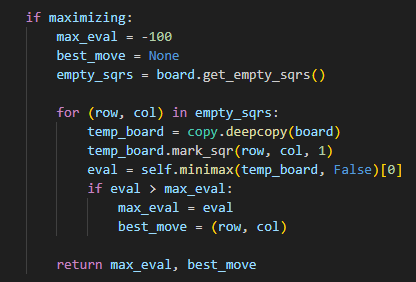


*figure 4: Functions for sending game mode and first player values*

Besides, other functions like *draw\_button(text, pos):, draw\_title():,* and *check\_button\_click(mouse\_pos):* are being used for displaying the rectangular shape button, showing the page title, and input the button clicks based on the mouse coordinates respectively.

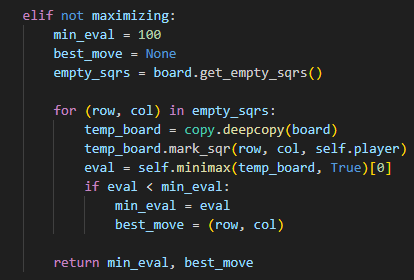
## Implementation of the Main Game:

If the user chooses to play with the AI player, the user has to choose either the easy mode or the hard mode. If the player chooses the easy mode to play, then the user will get two options. He can choose who will give the first move. Then AI player will choose its turn randomly based on the availability of the squares on the 3 × 3 board. In contrast, for the hard mode, the minimax algorithm is used here for giving the moves. The AI player recursively creates all possible moves based on the user’s move to find the best possible move. If the function is in a maximising turn, for each move the *minimax(self, board, maximising):* function copies the main board (3 × 3 board) into a variable called *temp\_board*,marks the square, and recursively calls the *minimax(self, board, maximising):* with maximising set to *False*. It then updates the *max\_eval* and *best\_move if it* finds a better move.



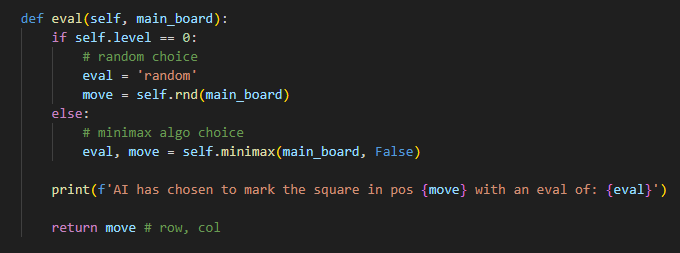
*figure 4. Maximising Turn*

If the function is in a minimising turn (looking for the best move for player 2), it follows the similar process mentioned above but with *min\_eval*.



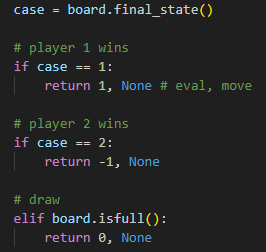
*figure 5. Minimising Turn*

Basically, the main function in the game is the *minimax(self, board, maximising):* function. In fewer words, the function recursively checks all depths and then by evaluating the scores, the algorithm returns the best move.



*figure 6. Evaluation of each move*

We have shown the evaluation result by observing the *final state*. If the board is full it means there are no squares left. In this case, the function will return 0, which means the game is drawn. The function will return 1 and -1 if the user and the AI player win the game respectively.



*figure 7. Three Cases of Final State*

Besides, there are two other classes, Board and Game. These classes are created for designing the interface.

# Output:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Results:

1. Tic Tac Toe Game: We have created a tic tac toe game where two players can take turns to make moves. It can also detect a win or draw and provide feedback based on that.

2. AI Player: The AI player can make a move using the minimax algorithm. The AI player evaluates the state of the board recursively and then based on the board state, the AI player chooses the best move.

3. Game Interface: We have created the game using pygame library. The user can choose whether he wants to play with a real player or an AI player. The user can also choose whether he wants to make the first move or not. After the game, the interface shows a message about whether the game is drawn or not.

4. Difficulty Levels: We have developed two difficulty levels. They are Easy and Hard. For the easy mode, the AI player makes a move randomly. For the hard mode, the AI player evaluates the board recursively and selects the best move among all possible moves.

5. User Experience: We have added keyboard shortcuts to enhance the user experience. Also in the final state, a graphical line cuts the crosses or zeros to make the game more engaging to the players.

6. Testing and Validation: We have tested all the buttons to check whether they work properly or not. All the functions implemented in our code work fine. We have also tried different combinations in the hard game mode to test if the minimax algorithm works perfectly or not. The minimax algorithm never failed to choose the best move among all possible moves by evaluating the board.

# Conclusion:

In summary, our tic-tac-toe game brings simple joy to players of all ages. In the classic 3x3 grid, players take turns placing her X's and O's, aiming to get three in a row. The game is easy to understand and encourages strategic thinking and friendly competition. Whether you're playing against friends or taking on the AI, the experience is accessible and fun.

The user-friendly interface enhances your gaming experience and allows for fast and enjoyable rounds. It's a timeless pursuit of fun and friendly rivalry. We hope players feel the excitement with every click and enjoy the simplicity and charm of this classic game.

.