KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

<u>Course Code</u>: CSE 4110

<u>Course Title</u>: Artificial Intelligence Laboratory

<u>Project Title</u>: “Match The Line” AI Game

<u>Submitted to:</u>

Md. Shahidul Salim

Lecturer

Department of Computer Science and Engineering

Khulna University of Engineering & Technology, Khulna

Most. Kaniz Fatema Isha

Lecturer

Department of Computer Science and Engineering

Khulna University of Engineering & Technology, Khulna

<u>Submitted by:</u>

Name: Sagar Dutta

Roll: 1907085

Name: Wasif Zahin

Roll: 1907112

Department of Computer Science and Engineering

Khulna University of Engineering & Technology, Khulna

Introduction:

"Match The Line" is a simple yet strategic two-player game involving movement of pieces on a 3x3 grid. The objective is to align three pieces of the same color in a straight line, either horizontally, vertically, or diagonally but not in the initial starting line of the each player's accordingly. This game combines elements of traditional board games with modern AI techniques, providing both human and AI players a chance to test their strategic skills.

Features:

1. **AI Opponent**:
   - ➢ Play against an AI opponent powered by the minimax algorithm and fuzzy logic for enhanced strategic decision-making.
2. **Graphical User Interface**:
   - ➢ Utilizes the Pygame library to create an interactive and visually appealing game board.
   - ➢ Easy-to-use interface with clear visual indicators for valid moves and selected pieces.
3. **Turn-based Gameplay**:
   - ➢ Alternates turns between the two players, ensuring fair play.
4. **Move Validation**:
   - ➢ Highlights valid moves for the selected piece, preventing illegal moves.
5. **Win Detection**:
   - ➢ Automatically detects and announces the winner when three pieces of the same color are aligned in a straight line.

Game Rules:

1. **Objective**:

   ➢ The objective of the game is to align three pieces of your color in a straight line horizontally, vertically, or diagonally.

2. **Setup**:

   ➢ The game board is a 3x3 grid.
   ➢ Each player has three pieces: Player has red pieces and the AI has pink pieces.
   ➢ The initial setup places the red pieces on the top row and the pink pieces on the bottom row.

3. **Turns**:

   ➢ Players take turns to move one of their pieces.
   ➢ Red goes first, followed by pink.

4. **Valid Moves**:

   ➢ A piece can move to an adjacent empty spot either horizontally, vertically, or diagonally.
   ➢ A piece cannot jump over other pieces.

5. **Movement Restrictions**:

   ➢ Pieces can only move to adjacent spots.
   ➢ Special rules apply to movements involving the central positions to maintain game balance.

6. **Winning the Game**:

   - A player wins by aligning three pieces of their color in a straight line horizontally, vertically, or diagonally.
   - The game immediately ends when a winning condition is met, and the winning player is announced.

7. **AI Gameplay**:

   - When playing against the AI, the AI will make moves based on the minimax algorithm with additional fuzzy logic for strategic decision-making.
   - The AI aims to maximize its chances of winning while minimizing the player's chances.

<u>How to Play</u>:

1. **Start the Game**:

   - Launch the game to see the 3x3 grid with initial piece positions.

2. **Make a Move**:

   - Click on a piece to select it.
   - The game will highlight valid moves for the selected piece on blue color.
   - Click on a highlighted spot to move the piece.

3. **Alternate Turns**:

   - After making a move, it will be the opponent's turn.
   - Repeat the steps to make moves until a player wins or the game ends in a draw.

4. **Winning and Restarting**:

   - The game announces the winner when three pieces are aligned.
   - Restart the game to play again.

Pseudocode:

1. **Minimax:**

```
function minimax(position, depth, max_player, game):
if depth == 0 or position has a winner:
   return evaluate(position) - depth, position

if max_player:
   maxEval = negative infinity
   best_move = None
   for each move in get_all_moves(position, current_player, game):
      evaluation = minimax(move, depth - 1, False, game)[0]
      if evaluation > maxEval:
         maxEval = evaluation
         best_move = move
   return maxEval, best_move
else:
   minEval = positive infinity
   best_move = None
   for each move in get_all_moves(position, current_player, game):
      evaluation = minimax(move, depth - 1, True, game)[0]
      if evaluation < minEval:
         minEval = evaluation
         best_move = move
   return minEval, best_move
```

2. **Fuzzy Logic:**

```
function fuzzy_evaluation(board):

    initialize fuzzy variables: occupancy, proximity, score
    define fuzzy membership functions for occupancy and proximity
    define fuzzy rules for score based on occupancy and proximity

    create fuzzy control system with defined rules
    create fuzzy control system simulation with the control system

    generate random values for occupancy and proximity within defined ranges

    set input values for occupancy and proximity in the control system simulation
    compute the output score based on the inputs

    return the computed score
```

Contribution:

SL.	Sagar Dutta (1907085)	Wasif Zahin (1907112)
1	Game topic selection	Game Roadmap
2	Design the board	Design the pieces
3	Implement Minimax	Implement Fuzzy Logic
4	Debug	Debug

Discussion:

“Match The Line” project was developed through combining fundamental aspects of conventional board games and modern AI techniques, particularly the minimax algorithm and fuzzy logic. By being implemented in Python using Pygame, it had a visually dynamic that incorporated an interactive interface which made the game even more fun to play. The major issues faced were circular imports resulting from relationships between different elements of the game and harmonizing between AI decision-making algorithms. However, this was overcome by this project through proper management of modules and repeated testing results into a playable game.

Conclusion:

In conclusion, the project offers dynamic and strategic game play as it merges AI strategies with classic mechanics. Python and Pygame were used in this project to ensure a user-friendly interface and strong gameplay. If the game is updated later on, certain things could be improved such as AI’s behavior, performance optimization or enhanced multiplayer support and increased difficulty levels. In summary, by developing games using Python developers demonstrate how adaptable AI algorithms can make traditional gaming better.