AIR UNIVERSITY

Department of Electrical and Computer Engineering

Lab # 05: Tic-Tac-Toe

Lab Instructor: Muhammad Awais

Tic-Tac-Toe in Artificial Intelligence

The development of a Tic Tac Toe game presents an excellent opportunity to delve into the realms of artificial intelligence (AI) and computer science principles, particularly constraint satisfaction problems (CSPs). In this lab task, we aim to create a single-player Tic Tac Toe game where the player competes against an AI opponent. Through this project, we will implement various AI techniques, including the Minimax algorithm, to enable the AI to make strategic decisions in the game. By leveraging these AI concepts, we seek to enhance the gameplay experience and provide a challenging opponent for players to compete against.

Tic-Tac-Toe

```
import math

Player_X = 'X'
Player_O = 'O'
EMPTY = ' '

def print_board(board):
    for row in board:
        print(" | ".join(row))
        print("-" * 9)
```

```
import math

Player_X = 'X'
Player_O = 'O'
EMPTY = ' '

def print_board(board):
    for row in board:
        print(" | ".join(row))
        print("-" * 9)


def check_winner(board, player):
    for i in range(3):
        if all(board[i][j] == player for j in range(3)) or all(board[j][i] == player for j i
            return True
    if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i i
        return True
    return False


def game_over(board):
  return check_winner(board, Player_X) or check_winner(board, Player_O) or \
  all(board[i][j] != EMPTY for i in range(3) for j in range(3))


def evaluate(board):
  if check_winner(board, Player_X):
    return 1
  elif check_winner(board, Player_O):
    return -1
  else:
    return 0
```

```
def minimax(board, depth, maximizing):
    if game_over(board) or depth == 0:
        return evaluate(board)

    if maximizing:
        max_eval = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == EMPTY:
                    board[i][j] = Player_X
                    eval = minimax(board, depth - 1, False)
                    board[i][j] = EMPTY
                    max_eval = max(max_eval, eval)
        return max_eval
    else:
        min_eval = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == EMPTY:
                    board[i][j] = Player_O
                    eval = minimax(board, depth - 1, True)
                    board[i][j] = EMPTY
                    min_eval = min(min_eval, eval)
        return min_eval


def find_best_move(board):
    best_eval = -math.inf
    best_move = None
    for i in range(3):
        for j in range(3):
            if board[i][j] == EMPTY:
                board[i][j] = Player_X
                eval = minimax(board, 10, False)  # depth can be adjusted
                board[i][j] = EMPTY
                if eval > best_eval:
                    best_eval = eval
                    best_move = (i, j)
    return best_move
```

```
    board = [[EMPTY] * 3 for _ in range(3)]
    print_board(board)
    while not game_over(board):
        x, y = map(int, input("Enter your move (row column): ").split())
        if board[x][y] != EMPTY:
            print("Invalid move. Try again.")
            continue
        board[x][y] = Player_O
        print_board(board)
        if game_over(board):
            break
        print("AI is thinking...")
        x, y = find_best_move(board)
        board[x][y] = Player_X
        print_board(board)
    if check_winner(board, Player_X):
        print("AI wins!")
    elif check_winner(board, Player_O):
        print("You win!")
    else:
        print("It's a draw!")
```

```
      |   |
     ---------
      |   |
     ---------
      |   |
     ---------
     Enter your move (row column): 1 1
      |   |
     ---------
      | O |
     ---------
      |   |
     ---------
     AI is thinking...
     X |   |
     ---------
      | O |
     ---------
      |   |
     ---------
     Enter your move (row column): 0 2
     X |   | O
     ---------
      | O |
     ---------
      |   |
     ---------
     AI is thinking...
     X |   | O
     ---------
      | O |
     ---------
```

```
X |   |
---------
Enter your move (row column): 1 0
X |   | O
---------
O | O |
---------
X |   |
---------
AI is thinking...
X |   | O
---------
O | O | X
---------
```

Conclusion

In conclusion, the development of the Tic Tac Toe game with AI has been a rewarding exercise in applying AI and computer science principles to create an engaging and challenging gaming experience. Through the implementation of the Minimax algorithm, we have successfully empowered the AI opponent to make intelligent moves based on strategic analysis of the game state. This project not only demonstrates the practical applications of AI in game development but also serves as a valuable learning experience in problem-solving, algorithm design, and programming. As we reflect on the journey of building this game, we recognize the potential for